Linux Professional Institute

LPIC-1

Verzió 5.0

Magyar

101

Table of Contents

Linux Professional Institute

Témakör 101: Rendszerarchitektúra

101.1 A hardverbeállítások meghatározása és konfigurálása

Hivatkozás az LPI célkitűzésre

LPIC-1 v5, Exam 101, Objective 101.1

Súlyozás

2

Kulcsfontosságú ismeretek

- Integrált perifériák engedélyezése és letiltása.
- Különbség a különböző típusú tömegtároló eszközök között.
- Az eszközök hardveres erőforrásainak meghatározása.
- Eszközök és segédprogramok a különböző hardverinformációk listázására (pl. lsusb, lspci stb.).
- Eszközök és segédprogramok az USB-eszközök manipulálásához.
- A sysfs, udev és dbus fogalmi megértése.

A használt fájlok, kifejezések és segédprogramok listája

- `/sys/`
- `/proc/`
- `/dev/`
- `modprobe`
- `lsmod`
- `lspci`
- `lsusb`

101.1 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	101 Rendszerarchitektúra
Fejezet:	101.1 A hardverbeállítások meghatározása és konfigurálása
Lecke:	1/1

Bevezetés

Az elektronikus számítástechnika kezdete óta az üzleti és személyi számítógépek gyártói különféle hardvereket építettek be a gépeikbe, ezeket az elemeket viszont az operációs rendszernek támogatnia kell. Ez az operációs rendszer fejlesztője szempontjából túlterhelő lehet, hacsak az iparág nem hoz létre szabványokat az utasításkészletekre és az eszközkommunikációra vonatkozóan. Hasonlóan az operációs rendszer által egy alkalmazás számára biztosított szabványosított absztrakciós réteghez, ezek a szabványok megkönnyítik egy olyan operációs rendszer megírását és karbantartását, amely nem kötődik egy adott hardvermodellhez. Az integrált mögöttes hardver összetettsége azonban néha kiegészítő beállításokat igényel azzal kapcsolatban, hogy az erőforrásokat hogyan kell az operációs rendszer számára hozzáférhetővé tenni, hogy az telepíthető legyen és megfelelően működhessen.

E beállítások némelyike telepített operációs rendszer nélkül is elvégezhető. A legtöbb gépen van egy konfigurációs segédprogram, amely a gép bekapcsolásakor futtatható. A 2000-es évek közepéig a konfigurációs segédprogramot a BIOS-ban (*Basic Input/Output System*), az x86-os alaplapokon található alapvető konfigurációs rutinokat tartalmazó firmware-szabványban valósították meg. A 2000-es évek első évtizedének végétől az x86-os architektúrán alapuló gépeken

a BIOS-t egy új, UEFI (*Unified Extensible Firmware Interface*) nevű implementációval kezdték helyettesíteni, amely kifinomultabb funkciókkal rendelkezik az azonosítás, a tesztelés, a konfiguráció és a firmware frissítése terén. A változás ellenére nem ritka, hogy a konfigurációs segédprogramot továbbra is BIOS-nak nevezik, mivel mindkét implementáció ugyanazt az alapvető célt szolgálja.

NOTE A BIOS és az UEFI közötti hasonlóságok és különbségek további részleteiről egy későbbi leckében lesz szó.

Az eszköz aktiválása

A rendszerkonfigurációs segédprogram a számítógép bekapcsolásakor egy adott billentyű lenyomása után jelenik meg. Az, hogy melyik billentyűt kell megnyomni, gyártónként változik, de általában a `Del` vagy az egyik funkcióbillentyű, például a `F2` vagy a `F12`. A használandó billentyűkombináció gyakran megjelenik a bekapcsolási képernyőn.

A BIOS beállítási segédprogramban lehetőség van az integrált perifériák engedélyezésére és letiltására, az alapvető hibavédelem aktiválására és a hardverbeállítások, például az IRQ (megszakításkérés) és a DMA (közvetlen memóriaelérés) módosítására. A modern gépeken ritkán van szükség ezeknek a beállításoknak a megváltoztatására, de szükség lehet a speciális problémák megoldásához szükséges módosításokra. Vannak például olyan RAM-technológiák, amelyek az alapértelmezett értékeknél gyorsabb adatátviteli sebességgel kompatibilisek, ezért ajánlott a gyártó által megadott értékekre módosítani. Egyes CPU-k olyan funkciókat kínálnak, amelyek az adott telepítéshez nem feltétlenül szükségesek és kikapcsolhatók. A letiltott funkciók csökkentik az energiafogyasztást, és növelhetik a rendszer védelmét, mivel az ismert hibákat tartalmazó CPU-funkciók is letilthatók.

Ha a gépben több tároló van, fontos meghatározni, hogy melyik rendelkezik a megfelelő bootloaderrel, és melyik legyen az első bejegyzés az eszköz indítási sorrendjében. Előfordulhat, hogy az operációs rendszer nem töltődik be, ha a BIOS bootellenőrzési listájában a helytelen eszköz áll az első helyen.

Eszközellenőrzés Linuxban

Miután az eszközök megfelelően azonosítva lettek, az operációs rendszer feladata, hogy hozzárendelje azokat a szoftverkomponenseket, amelyekre szükségük van. Ha egy hardverfunkció nem az elvártaknak megfelelően működik, fontos azonosítani, hogy pontosan hol jelentkezik a probléma. Ha az operációs rendszer nem érzékel egy hardverelemet, akkor nagy valószínűséggel az alkatrész — vagy a port, amelyhez csatlakozik — hibás. Ha a hardverrész helyesen érzékelődik, de mégsem működik megfelelően, akkor az operációs rendszer oldalán

lehet probléma. Ezért a hardverrel kapcsolatos problémák kezelésének egyik első lépése annak ellenőrzése, hogy az operációs rendszer megfelelően érzékeli-e az eszközt. A hardveres erőforrások azonosításának két alapvető módja van a Linux rendszerben: speciális parancsok használata vagy specifikus fájlrendszereken belül meghatározott fájlok beolvasása.

Az ellenőrzés parancsai

Linux rendszerben a csatlakoztatott eszközök azonosítására szolgáló két alapvető parancs a következő:

`lspci`

A PCI (*Peripheral Component Interconnect*) buszhoz jelenleg csatlakoztatott összes eszközt megjelenítése. A PCI-eszközök lehetnek az alaplaphoz csatlakoztatott alkatrészek, mint például egy lemezvezérlő, vagy egy PCI-slotba illesztett bővítőkártyák, mint például egy külső videokártya.

`lsusb`

A számítógéphez jelenleg csatlakoztatott USB (*Universal Serial Bus*) eszközök listája. Bár szinte minden elképzelhető célra léteznek USB-eszközök, az USB-interfész nagyrészt beviteli eszközök — billentyűzetek, mutatós eszközök — és cserélhető adathordozók csatlakoztatására szolgál.

Az `lspci` és `lsusb` parancsok kimenete az operációs rendszer által azonosított összes PCI és USB eszköz listája. Előfordulhat azonban, hogy az eszköz még nem teljesen működőképes, mivel minden hardvernek szüksége van egy szoftverkomponensre a megfelelő eszköz vezérléséhez. Ezt a szoftverkomponenst *kernelmodulnak* nevezik, és lehet a hivatalos Linux kernel része, vagy kerülhet más forrásból hozzáadásra.

A hardvereszközökhöz kapcsolódó Linux kernel modulokat más operációs rendszerekhez hasonlóan *drivernek* is nevezik. A Linux drivereket azonban nem mindig az eszköz gyártója szállítja. Míg egyes gyártók saját bináris illesztőprogramokat biztosítanak, amelyeket külön kell telepíteni, sok illesztőprogramot független fejlesztők írnak. Történelmileg például a Windows alatt működő eszközöknek nem biztos, hogy van megfelelő kernelmoduljuk Linuxhoz. Manapság a Linux-alapú operációs rendszerek erős hardvertámogatással rendelkeznek, és a legtöbb eszköz könnyedén működésre bírható.

A hardverhez közvetlenül kapcsolódó parancsok gyakran root jogosultságokat igényelnek a végrehajtáshoz, vagy csak korlátozott információkat jelenítenek meg, ha normál felhasználó hajtja végre őket, ezért előfordulhat, hogy root felhasználóként kell bejelentkezni, vagy a parancsot `sudo` paranccsal kell végrehajtani.

Az `lspci` parancs következő kimenete például néhány eszközt jelenít meg:

```
$ lspci
01:00.0 VGA compatible controller: NVIDIA Corporation GM107 [GeForce GTX 750 Ti] (rev a2)
04:02.0 Network controller: Ralink corp. RT2561/RT61 802.11g PCI
04:04.0 Multimedia audio controller: VIA Technologies Inc. ICE1712 [Envy24] PCI Multi-
Channel I/O Controller (rev 02)
04:0b.0 FireWire (IEEE 1394): LSI Corporation FW322/323 [TrueFire] 1394a Controller (rev 70)
```

Az ilyen parancsok kimenete több tucat sornyi is lehet, ezért az előző és a következő példa csak az érdekesebb részeket tartalmazza. Az egyes sorok elején található hexadecimális számok a megfelelő PCI-eszköz egyedi címeit jelentik. Az `lspci` parancs további részleteket mutat egy adott eszközről, ha annak címét az `-s` és a `-v` kapcsolóval együtt adjuk meg:

```
$ lspci -s 04:02.0 -v
04:02.0 Network controller: Ralink corp. RT2561/RT61 802.11g PCI
    Subsystem: Linksys WMP54G v4.1
    Flags: bus master, slow devsel, latency 32, IRQ 21
    Memory at e3100000 (32-bit, non-prefetchable) [size=32K]
    Capabilities: [40] Power Management version 2
    kernel driver in use: rt61pci
```

A kimenet most már sokkal több részletet mutat a `04:02.0` címen lévő eszközről. Ez egy hálózati vezérlő, amelynek belső neve `Ralink corp. RT2561/RT61 802.11g PCI`. A `Subsystem` az eszköz márkájához és modelljéhez — `Linksys WMP54G v4.1` — kapcsolódik, és diagnosztikai célok esetén lehet hasznos.

A kernelmodul azonosítható a `kernel driver in use` sorban, amely az `rt61pci` modult mutatja. Az összes összegyűjtött információból helyesen feltételezhető, hogy:

1. Az eszköz azonosításra került.
2. A megfelelő kernelmodul betöltődött.
3. Az eszköz készen áll a használatra.

Egy másik lehetőség annak ellenőrzésére, hogy melyik kernelmodul van használatban a megadott eszközhöz, az `lspci` újabb verzióiban elérhető `-k` kapcsoló:

```
$ lspci -s 01:00.0 -k
01:00.0 VGA compatible controller: NVIDIA Corporation GM107 [GeForce GTX 750 Ti] (rev a2)
    kernel driver in use: nvidia
```

```
    kernel modules: nouveau, nvidia_drm, nvidia
```

A kiválasztott eszköz, egy NVIDIA GPU kártya esetében az `lspci` megmondja, hogy a használt modul neve `nvidia`, a `kernel driver in use: nvidia` sorban, és az összes hozzá tartozó kernelmodul a `kernel modules: nouveau, nvidia_drm, nvidia` sorban található.

Az `lsusb` parancs hasonló az `lspci`-hez, de kizárólag az USB-információkat listázza ki:

```
$ lsusb
Bus 001 Device 029: ID 1781:0c9f Multiple Vendors USBtiny
Bus 001 Device 028: ID 093a:2521 Pixart Imaging, Inc. Optical Mouse
Bus 001 Device 020: ID 1131:1001 Integrated System Solution Corp. KY-BT100 Bluetooth Adapter
Bus 001 Device 011: ID 04f2:0402 Chicony Electronics Co., Ltd Genius LuxeMate i200 Keyboard
Bus 001 Device 007: ID 0424:7800 Standard Microsystems Corp.
Bus 001 Device 003: ID 0424:2514 Standard Microsystems Corp. USB 2.0 Hub
Bus 001 Device 002: ID 0424:2514 Standard Microsystems Corp. USB 2.0 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Az `lsusb` parancs megmutatja a rendelkezésre álló USB-csatornákat és a hozzájuk csatlakoztatott eszközöket. Az `lspci` parancshoz hasonlóan a `-v` opció részletesebb kimenetet jelenít meg. Egy adott eszköz kiválasztható, ha megadjuk az azonosítóját a `-d` kapcsolónál:

```
$ lsusb -v -d 1781:0c9f
Bus 001 Device 029: ID 1781:0c9f Multiple Vendors USBtiny
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               1.01
  bDeviceClass          255 Vendor Specific Class
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0         8
  idVendor           0x1781 Multiple Vendors
  idProduct          0x0c9f USBtiny
  bcdDevice            1.04
  iManufacturer           0
  iProduct                2 USBtiny
  iSerial                 0
  bNumConfigurations      1
```

A `-t` opcióval az `lsusb` parancs hierarchikus fa formájában mutatja az aktuális USB-eszköz

hozzárendeléseket:

```
$ lsusb -t
/:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=dwc_otg/1p, 480M
    |__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/4p, 480M
        |__ Port 1: Dev 3, If 0, Class=Hub, Driver=hub/3p, 480M
            |__ Port 2: Dev 11, If 1, Class=Human Interface Device, Driver=usbhid, 1.5M
            |__ Port 2: Dev 11, If 0, Class=Human Interface Device, Driver=usbhid, 1.5M
            |__ Port 3: Dev 20, If 0, Class=Wireless, Driver=btusb, 12M
            |__ Port 3: Dev 20, If 1, Class=Wireless, Driver=btusb, 12M
            |__ Port 3: Dev 20, If 2, Class=Application Specific Interface, Driver=, 12M
            |__ Port 1: Dev 7, If 0, Class=Vendor Specific Class, Driver=lan78xx, 480M
        |__ Port 2: Dev 28, If 0, Class=Human Interface Device, Driver=usbhid, 1.5M
        |__ Port 3: Dev 29, If 0, Class=Vendor Specific Class, Driver=, 1.5M
```

Lehetséges, hogy nem minden eszközhöz tartoznak megfelelő modulok. Bizonyos eszközökkel a kommunikációt az alkalmazás közvetlenül, modulok közvetítése nélkül is kezelheti. Ennek ellenére az `lsusb -t` kimenete sok információt tartalmaz. Ha létezik megfelelő modul, akkor annak neve az eszközhöz tartozó sor végén jelenik meg, mint a `Driver=btusb` sorban. Az eszköz `Class` az általános kategóriát azonosítja, mint például `Human Interface Device`, `Wireless`, `Vendor Specific Class`, `Mass Storage`, stb. Annak ellenőrzéséhez, hogy melyik eszköz használja az előző felsorolásban szereplő `btusb` modult, az `lsusb` parancs `-s` kapcsolójához meg kell adni mind a `Bus`, mind a `Dev` számát:

```
$ lsusb -s 01:20
Bus 001 Device 020: ID 1131:1001 Integrated System Solution Corp. KY-BT100 Bluetooth Adapter
```

Gyakori, hogy egy szabványos Linux rendszerben bármikor nagyszámú betöltött kernelmodult találunk. A velük való interakció előnyösebb módja a `kmod` csomag által biztosított parancsok használata, amely egy eszközkészlet a Linux kernelmodulokkal kapcsolatos általános feladatok kezelésére, mint például a modulok beszúrása, eltávolítása, listázása, tulajdonságok ellenőrzése, függőségek és aliasok feloldása. Az `lsmod` parancs például megmutatja az összes jelenleg betöltött modult:

```
$ lsmod
Module                  Size  Used by
kvm_intel             138528  0
kvm                   421021  1 kvm_intel
iTCO_wdt               13480  0
iTCO_vendor_support    13419  1 iTCO_wdt
```

```
snd_usb_audio         149112  2
snd_hda_codec_realtek  51465  1
snd_ice1712            75006  3
snd_hda_intel          44075  7
arc4                   12608  2
snd_cs8427             13978  1 snd_ice1712
snd_i2c                13828  2 snd_ice1712,snd_cs8427
snd_ice17xx_ak4xxx     13128  1 snd_ice1712
snd_ak4xxx_adda        18487  2 snd_ice1712,snd_ice17xx_ak4xxx
microcode              23527  0
snd_usbmidi_lib        24845  1 snd_usb_audio
gspca_pac7302          17481  0
gspca_main             36226  1 gspca_pac7302
videodev              132348  2 gspca_main,gspca_pac7302
rt61pci                32326  0
rt2x00pci              13083  1 rt61pci
media                  20840  1 videodev
rt2x00mmio             13322  1 rt61pci
hid_dr                 12776  0
snd_mpu401_uart        13992  1 snd_ice1712
rt2x00lib              67108  3 rt61pci,rt2x00pci,rt2x00mmio
snd_rawmidi            29394  2 snd_usbmidi_lib,snd_mpu401_uart
```

Az `lsmod` parancs kimenete három oszlopra van osztva:

`Module`

Modulnév.

`Size`

A modul által lefoglalt RAM mérete, bájtban.

`Used by`

Ettől a modultól függő modulok.

Néhány modulnak más modulokra van szüksége a megfelelő működéshez, például az audioeszközök moduljainak esetében:

```
$ lsmod | fgrep -i snd_hda_intel
snd_hda_intel          42658  5
snd_hda_codec         155748  3 snd_hda_codec_hdmi,snd_hda_codec_via,snd_hda_intel
snd_pcm                81999  3 snd_hda_codec_hdmi,snd_hda_codec,snd_hda_intel
snd_page_alloc         13852  2 snd_pcm,snd_hda_intel
```

```
snd                    59132  19
snd_hwdep,snd_timer,snd_hda_codec_hdmi,snd_hda_codec_via,snd_pcm,snd_seq,snd_hda_codec,snd_h
da_intel,snd_seq_device
```

A harmadik oszlop, a `Used by`, azokat a modulokat mutatja, amelyeknek az első oszlopban szereplő modulra van szükségük a megfelelő működéshez. A Linux hangarchitektúra számos modulja, amelyek előtt az `snd` előtag szerepel, egymástól függ. Amikor a rendszerdiagnosztika során problémákat keresünk, hasznos lehet az éppen betöltött egyes modulok eltávolítása. A `modprobe` parancs használható mind a kernelmodulok betöltésére, mind a modulok betöltésének megszüntetésére: egy modul és a hozzá kapcsolódó modulok eltávolításához, amennyiben azokat nem használja egy futó folyamat, a `modprobe -r` parancsot kell használni. Például az `snd-hda-intel` modul (az Intel HDA audio eszköz modulja) és más, a hangrendszerhez kapcsolódó modulok betöltésének megszüntetéséhez az alábbi parancsra van szükség:

```
# modprobe -r snd-hda-intel
```

A kernelmodulok betöltése és eltávolítása mellett a rendszer futása közben is lehetőség van a modul paramétereinek módosítására a kernel betöltése közben, ami nem sokban különbözik a kapcsolók parancsoknak való átadásától. Minden modul elfogad bizonyos paramétereket, de a legtöbbször az alapértelmezett értékek javasoltak és extra paraméterekre nincs szükség. Bizonyos esetekben azonban szükség van a paraméterek használatára, hogy a modul viselkedése az elvárt módon történjen.

A modul nevét egyetlen argumentumként használva a `modinfo` parancs megjeleníti az adott modul leírását, a fájlt, a szerzőt, a licencet, az azonosítót, a függőségeket és az elérhető paramétereket. A modulhoz tartozó egyéni paramétereket az `/etc/modprobe.conf` fájlba vagy az `/etc/modprobe.d/` mappában a `.conf` kiterjesztésű egyedi fájlokba való beillesztéssel lehet tartósan elérhetővé tenni. A `-p` kapcsoló hatására a `modinfo` parancs megjeleníti az összes rendelkezésre álló paramétert, és figyelmen kívül hagyja a többi információt:

```
# modinfo -p nouveau
vram_pushbuf:Create DMA push buffers in VRAM (int)
tv_norm:Default TV norm.
                Supported: PAL, PAL-M, PAL-N, PAL-Nc, NTSC-M, NTSC-J,
                        hd480i, hd480p, hd576i, hd576p, hd720p, hd1080i.
                Default: PAL
                NOTE Ignored for cards with external TV encoders. (charp)
nofbaccel:Disable fbcon acceleration (int)
fbcon_bpp:fbcon bits-per-pixel (default: auto) (int)
mst:Enable DisplayPort multi-stream (default: enabled) (int)
```

```
tv_disable:Disable TV-out detection (int)
ignorelid:Ignore ACPI lid status (int)
duallink:Allow dual-link TMDS (default: enabled) (int)
hdmimhz:Force a maximum HDMI pixel clock (in MHz) (int)
config:option string to pass to driver core (charp)
debug:debug string to pass to driver core (charp)
noaccel:disable kernel/abi16 acceleration (int)
modeset:enable driver (default: auto, 0 = disabled, 1 = enabled, 2 = headless) (int)
atomic:Expose atomic ioctl (default: disabled) (int)
runpm:disable (0), force enable (1), optimus only default (-1) (int)
```

A kimeneti minta a `nouveau` modulhoz rendelkezésre álló összes paramétert mutatja, amely a *Nouveau Project* által az NVIDIA GPU-kártyák saját fejlesztésű illesztőprogramjainak alternatívájaként biztosított kernelmodul. A `modeset` kapcso például lehetővé teszi annak beállítását, hogy a kijelző felbontása és mélysége a kernel-térben legyen beállítva, ne pedig a felhasználói térben. Az `options nouveau modeset=0` hozzáadása a `/etc/modprobe.d/nouveau.conf` fájlhoz letiltja a modeset kernel funkciót.

Ha egy modul problémákat okoz, az `/etc/modprobe.d/blacklist.conf` fájl segítségével blokkolhatjuk a betöltődését. Például a `nouveau` modul automatikus betöltésének megakadályozásához a `/etc/modprobe.d/blacklist.conf` fájlba be kell írni a `blacklist nouveau` sort. Erre a műveletre akkor van szükség, ha az `nvidia` saját modulja van telepítve, és az alapértelmezett `nouveau` modult félre kell tenni.

NOTE

Módosíthatjuk az `/etc/modprobe.d/blacklist.conf` fájlt, amely alapértelmezés szerint már létezik a rendszerben. A legmegfelelőbb módszer azonban egy külön konfigurációs fájl létrehozása, `/etc/modprobe.d/<modul_név>.conf` néven, amely csak az adott kernelmodulra vonatkozó beállításokat tartalmazza.

Információs fájlok és eszközfájlok

Az `lspci`, `lsusb` és `lsmod` parancsok az operációs rendszer által tárolt hardverinformációk olvasására szolgálnak. Ezeket az információkat a `/proc` és `/sys` mappákban található speciális fájlokban tárolják. Ezek a mappák olyan fájlrendszerek csatolási pontjai, amelyek nincsenek jelen az eszközpartícióban, hanem csak a RAM-ban, amelyet a kernel a futásidejű konfiguráció és a futó folyamatokra vonatkozó információk tárolására használ. Az ilyen fájlrendszerek nem hagyományos fájltárolásra szolgálnak, ezért pszeudo-fájlrendszereknek nevezzük őket, és csak a rendszer futása alatt léteznek. A `/proc` mappa a futó folyamatokra és hardveres erőforrásokra vonatkozó információkat tartalmazó fájlokat tartalmazza. Nézzünk meg néhány fontos fájlt a hardverek vizsgálatához a `/proc` fájlok közül:

`/proc/cpuinfo`

Az operációs rendszer által talált CPU(k) részletes információit listázza.

`/proc/interrupts`

A megszakítások számának listája IO-eszközönként minden CPU-hoz.

`/proc/ioports`

A jelenleg regisztrált, használatban lévő bemeneti/kimeneti portok listája.

`/proc/dma`

A használatban lévő regisztrált DMA (közvetlen memória-hozzáférés) csatornák listája.

A `/sys` mappában található fájlok hasonló szerepet töltenek be, mint a `/proc` mappában találhatóak. A `/sys` mappa azonban kifejezetten a hardverrel kapcsolatos eszközinformációk és kerneladatok tárolására szolgál, míg a `/proc` mappa a kernel különböző adatstruktúráira vonatkozó információkat is tartalmaz, beleértve a futó folyamatokat és a konfigurációt.

Egy másik, szabványos Linux rendszerben az eszközökhöz közvetlenül kapcsolódó mappa a `/dev`. A `/dev` mappában található minden fájl egy rendszereszközhöz, különösen a tárolóeszközökhöz kapcsolódik. Egy hagyományos IDE merevlemezt például az alaplap első IDE csatornájára csatlakoztatva a `/dev/hda` fájl képviseli. Ezen a lemezen minden partíciót a `/dev/hda1`, `/dev/hda2` azonosít, egészen az utolsó megtalált partícióig.

Az eltávolítható eszközöket a *udev* alrendszer kezeli, amely létrehozza a megfelelő eszközöket a `/dev`-ben. A Linux kernel rögzíti a hardverérzékelési eseményt, és átadja azt az udev folyamatnak, amely azonosítja az eszközt, és dinamikusan létrehozza a megfelelő fájlokat a `/dev` -ben, előre meghatározott szabályok alapján.

A mostani Linux disztribúciókban az udev felelős a gép bekapcsolásakor már jelenlévő eszközök azonosításáért és konfigurálásáért (*coldplug detection*), valamint a rendszer futása közben azonosított eszközökért (*hotplug detection*). Az Udev a *SysFS*-re támaszkodik, a `/sys`-be csatolt, hardverrel kapcsolatos információk pszeudo-fájlrendszerére.

NOTE

A hotplug kifejezés egy eszköz felismerésére és konfigurálására utal a rendszer futása közben, például egy USB-eszköz behelyezésekor. A Linux kernel a 2.6-os verzió óta támogatja a hotplug funkciókat, lehetővé téve a legtöbb rendszerbusz (PCI, USB, stb.) számára, hogy hotplug eseményeket triggereljen, amikor egy eszközt csatlakoztatnak vagy leválasztanak.

Az új eszközök észlelésekor az udev a `/etc/udev/rules.d/` mappában tárolt, előre definiált szabályok között keres megfelelőt. A legfontosabb szabályokat a disztribúció biztosítja, de egyedi

esetekre újakat is hozzá lehet adni.

Tárolóeszközök

A Linuxban a tárolóeszközöket általánosságban blokkeszközöknek nevezik, mivel az adatokat különböző méretű és pozíciójú pufferelt adatblokkokban olvassák be és ki ezekről az eszközökről. Minden blokkos eszközt a `/dev` mappában található fájl azonosít, amelynek neve az eszköz típusától (IDE, SATA, SCSI stb.) és partícióitól függ. A CD/DVD- és floppy-eszközök neve például ennek megfelelően lesz megadva a `/dev` mappában: a második IDE-csatornára csatlakoztatott CD/DVD-meghajtó `/dev/hdc` néven lesz azonosítva (a `/dev/hda` és `/dev/hdb` az első IDE-csatorna master- és slave-eszközei számára van fenntartva), egy régi floppy-meghajtó pedig `/dev/fd0`, `/dev/fd1` stb. néven lesz azonosítva.

A Linux kernel 2.4-es verziójától kezdve a legtöbb tárolóeszköz a hardver típusától függetlenül SCSI eszközként kerül azonosításra. Az IDE, SSD és USB blokkos eszközök elé az `sd` előtagot írjuk. Az IDE-lemezek esetében az `sd` előtag lesz használatos, de a harmadik betű attól függően kerül kiválasztásra, hogy a meghajtó master vagy slave (az első IDE-csatornában a master az `sda`, a slave pedig az `sdb` lesz). A partíciókat numerikusan sorolja fel a rendszer. A `/dev/sda1`, `/dev/sda2` stb. útvonalakat az elsőnek azonosított blokkeszköz első és második partíciójának, a `/dev/sdb1`, `/dev/sdb2` stb. útvonalakat pedig a másodiknak azonosított blokkeszköz első és második partíciójának azonosítására használjuk. Ez alól a minta alól kivételt képeznek a memóriakártyák (SD-kártyák) és az NVMe eszközök (PCI Express buszra csatlakoztatott SSD-k). SD-kártyák esetében az elsőnek azonosított eszköz első és második partíciójának azonosítására a `/dev/mmcblk0p1`, `/dev/mmcblk0p2` stb. útvonalakat, a másodiknak azonosított eszköz első és második partíciójának azonosítására pedig a `/dev/mmcblk1p1`, `/dev/mmcblk1p2` stb. útvonalakat használjuk. Az NVMe eszközök az `nvme` előtagot kapják, mint a `/dev/nvme0n1p1` és `/dev/nvme0n1p2`.

Gyakorló feladatok

1. Tegyük fel, hogy az operációs rendszer nem tud elindulni, miután egy második SATA-lemezt adtunk a rendszerhez. Tudva, hogy nem minden alkatrész hibás, mi lehet a hiba lehetséges oka?

2. Tegyük fel, hogy meg akarunk győződni arról, hogy az újonnan beszerzett asztali számítógép PCI-buszára csatlakoztatott külső videokártya valóban az, amelyet a gyártó hirdetett, de a számítógépház felnyitásával a garancia érvényét veszítené. Milyen paranccsal lehetne listázni a videokártya operációs rendszer által észlelt adatait?

3. Az alábbi sor az `lspci` parancs által generált kimenet egy része:

   ```
   03:00.0 RAID bus controller: LSI Logic / Symbios Logic MegaRAID SAS 2208 [Thunderbolt]
   (rev 05)
   ```

   Milyen parancsot kell végrehajtanunk az adott eszközhöz használt kernelmodul azonosításához?

4. A rendszergazda a rendszer újraindítása nélkül szeretné kipróbálni a `bluetooth` kernelmodul különböző paramétereit. Azonban a modul `modprobe -r bluetooth` segítségével történő lecsatlakoztatására tett kísérletek a következő hibát eredményezik:

   ```
   modprobe: FATAL: Module bluetooth is in use.
   ```

   Mi a hiba lehetséges oka?

Gondolkodtató feladatok

1. Nem ritka, hogy a termelési környezetben régebbi gépekkel találkozunk, például amikor egyes berendezések elavult kapcsolatot használnak a vezérlő számítógéppel való kommunikációra, ami miatt különös figyelmet kell fordítani ezeknek a régebbi gépeknek néhány sajátosságára. Egyes régebbi BIOS firmware-rel rendelkező x86-os szerverek például nem bootolnak, ha nem érzékelik a billentyűzetet. Hogyan kerülhető el ez a probléma?

2. A Linux kernel köré épülő operációs rendszerek az x86-os architektúrától eltérő számítógép-architektúrák széles skáláján is elérhetők, például az ARM architektúrán alapuló egykártyás számítógépeken. A figyelmes felhasználó észreveheti, hogy az ilyen gépeken, mint például a Raspberry Pi, hiányzik az `lspci` parancs. Az x86-os gépekkel szemben milyen különbség indokolja ezt a hiányt?

3. Sok router rendelkezik USB-porttal, amely lehetővé teszi külső eszközök, például USB merevlemezek csatlakoztatását. Mivel ezek többsége Linux alapú operációs rendszert használ, hogyan lesz egy külső USB merevlemez elnevezve a `/dev/` mappában, feltételezve, hogy nincs más hagyományos blokkeszköz a routerben?

4. 2018-ban fedezték fel a *Meltdown* néven ismert hardveres sebezhetőséget. Ez számos architektúra szinte valamennyi processzorát érinti. A Linux kernel legújabb verziói képesek jelezni, ha az aktuális rendszer sebezhető. Hogyan szerezhető be ez az információ?

Összefoglalás

Ez a lecke általános fogalmakat tárgyal arról, hogy a Linux kernel hogyan kezeli a hardver erőforrásokat, főként az x86 architektúrán. A lecke a következő témákat járja körül:

- Hogyan befolyásolhatják a BIOS vagy az UEFI konfigurációs segédprogramokban meghatározott beállítások az operációs rendszer és a hardver közötti interakciót.
- Hogyan használjuk a szabványos Linux rendszer által biztosított eszközöket a hardverrel kapcsolatos információk megszerzéséhez.
- Hogyan lehet azonosítani az állandó és cserélhető tárolóeszközöket a fájlrendszerben.

A tárgyalt parancsok és eljárások a következők voltak:

- Az észlelt hardverek felderítésére szolgáló parancsok: `lspci` és `lsusb`.
- A kernelmodulok kezelésére szolgáló parancsok: `lsmod` és `modprobe`.
- Hardverrel kapcsolatos speciális fájlok, akár a `/dev/` mappában található fájlok, vagy a `/proc/` és `/sys/` pszeudo-fájlrendszerekben.

Válaszok a gyakorló feladatokra

1. Tegyük fel, hogy az operációs rendszer nem tud elindulni, miután egy második SATA-lemezt adtunk a rendszerhez. Tudva, hogy nem minden alkatrész hibás, mi lehet a hiba lehetséges oka?

   A boot-eszköz sorrendjét a BIOS-ban kell beállítani, különben a BIOS esetleg nem tudja futtatni a bootloadert.

2. Tegyük fel, hogy meg akarunk győződni arról, hogy az újonnan beszerzett asztali számítógép PCI-buszára csatlakoztatott külső videokártya valóban az, amelyet a gyártó hirdetett, de a számítógépház felnyitásával a garancia érvényét veszítené. Milyen paranccsal lehetne listázni a videokártya operációs rendszer által észlelt adatait?

   Az `lspci` parancs részletes információkat ad a PCI buszra jelenleg csatlakoztatott összes eszközről.

3. Az alábbi sor az `lspci` parancs által generált kimenet egy része:

   ```
   03:00.0 RAID bus controller: LSI Logic / Symbios Logic MegaRAID SAS 2208 [Thunderbolt]
   (rev 05)
   ```

   Milyen parancsot kell végrehajtanunk az adott eszközhöz használt kernelmodul azonosításához?

   Az `lspci -s 03:00.0 -v` vagy `lspci -s 03:00.0 -k` parancsot.

4. A rendszergazda a rendszer újraindítása nélkül szeretné kipróbálni a `bluetooth` kernelmodul különböző paramétereit. Azonban a modul `modprobe -r bluetooth` segítségével történő lecsatlakoztatására tett kísérletek a következő hibát eredményezik:

   ```
   modprobe: FATAL: Module bluetooth is in use.
   ```

   Mi a hiba lehetséges oka?

   A `bluetooth` modult egy másik processz használja.

Válaszok a gondolkodtató feladatokra

1. Nem ritka, hogy a termelési környezetben régebbi gépekkel találkozunk, például amikor egyes berendezések elavult kapcsolatot használnak a vezérlő számítógéppel való kommunikációra, ami miatt különös figyelmet kell fordítani ezeknek a régebbi gépeknek néhány sajátosságára. Egyes régebbi BIOS firmware-rel rendelkező x86-os szerverek például nem bootolnak, ha nem érzékelik a billentyűzetet. Hogyan kerülhető el ez a probléma?

   A BIOS konfigurációs segédprogramban lehetőség van a számítógép zárolásának kikapcsolására, ha nem talál billentyűzetet.

2. A Linux kernel köré épülő operációs rendszerek az x86-os architektúrától eltérő számítógép-architektúrák széles skáláján is elérhetők, például az ARM architektúrán alapuló egykártyás számítógépeken. A figyelmes felhasználó észreveheti, hogy az ilyen gépeken, mint például a Raspberry Pi, hiányzik az `lspci` parancs. Az x86-os gépekkel szemben milyen különbség indokolja ezt a hiányt?

   A legtöbb x86-os géppel ellentétben egy ARM alapú számítógép, mint a Raspberry Pi, nem rendelkezik PCI busszal, így az `lspci` parancs használhatatlan.

3. Sok router rendelkezik USB-porttal, amely lehetővé teszi külső eszközök, például USB merevlemezek csatlakoztatását. Mivel ezek többsége Linux alapú operációs rendszert használ, hogyan lesz egy külső USB merevlemez elnevezve a `/dev/` mappában, feltételezve, hogy nincs más hagyományos blokkeszköz a routerben?

   A modern Linux kernelek az USB merevlemezeket SATA eszközként azonosítják, így a megfelelő fájl a `/dev/sda` lesz, mivel a rendszerben nincs más hagyományos blokkos eszköz.

4. 2018-ban fedezték fel a *Meltdown* néven ismert hardveres sebezhetőséget. Ez számos architektúra szinte valamennyi processzorát érinti. A Linux kernel legújabb verziói képesek jelezni, ha az aktuális rendszer sebezhető. Hogyan szerezhető be ez az információ?

   A `/proc/cpuinfo` fájlban van egy sor, amely a CPU ismert hibáit mutatja, például `bugs: cpu_meltdown`.

101.2 A rendszer indítása

Hivatkozás az LPI célkitűzésre

LPIC-1 v5, Exam 101, Objective 101.2

Súlyozás

3

Kulcsfontosságú ismeretek

- Általános parancsok kiadása a bootloadernek és kapcsolók megadása a rendszermagnak a rendszerindítás során.
- A BIOS/UEFI rendszerindítási sorrend ismeretének bemutatása a rendszerindítás befejezéséig.
- A SysVinit és a systemd megértése.
- Az Upstart ismerete.
- A bootolási események ellenőrzése a naplófájlokban.

A használt fájlok, kifejezések és segédprogramok listája

- `dmesg`
- `journalctl`
- BIOS
- UEFI
- bootloader
- kernel
- `initramfs`
- `init`

- SysVinit
- systemd

Linux Professional Institute

101.2 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	101 Rendszerarchitektúra
Fejezet:	101.2 A rendszer indítása
Lecke:	1/1

Bevezetés

A gép vezérléséhez az operációs rendszer fő összetevőjét — a kernelt --, egy *bootloader* nevű programnak kell betöltenie, amit egy előre telepített firmware, például a BIOS vagy az UEFI tölt be. Testreszabható, hogy a bootloader paramétereket adjon át a kernelnek, például azt, hogy melyik partíció tartalmazza a gyökérfájlrendszert, vagy azt, hogy az operációs rendszer milyen üzemmódban fusson. A betöltés után a kernel folytatja a rendszerindítási folyamatot, azonosítva és konfigurálva a hardvert. Végül a kernel meghívja a rendszer szolgáltatásainak indításáért és kezeléséért felelős segédprogramot.

NOTE Egyes Linux disztribúciókban az ebben a leckében végrehajtott parancsok root jogosultságokat igényelhetnek.

BIOS vagy UEFI

Az x86-os gépek által a bootloader futtatásához végrehajtott eljárások különböznek attól, hogy BIOS-t vagy UEFI-t használnak. A BIOS, a *Basic Input/Output System* rövidítése, az alaplaphoz csatlakoztatott, non-volatile (megmaradó) memóriachipen tárolt program, amely a számítógép minden egyes bekapcsolásakor végrehajtódik. Az ilyen típusú programot *firmware*-nek nevezik, és a tárolási helye elkülönül a rendszer esetleges egyéb tárolóeszközeitől. A BIOS feltételezi, hogy az első tárolóeszköz első 440 bájtja — a BIOS konfigurációs segédprogramjában meghatározott sorrend szerint — a bootloader (más néven *bootstrap*) első szakasza. A tárolóeszköz első 512 bájtját a DOS szabványos partíciós sémáját használó tárolóeszközök MBR-jének (*Master Boot Record*) nevezik, és a bootloader első szakaszán kívül a partíciós táblát is tartalmazza. Ha az MBR nem tartalmazza a megfelelő adatokat, a rendszer nem tud elindulni, hacsak nem alkalmazunk alternatív módszert.

Általánosságban elmondható, hogy a BIOS-szal ellátott rendszer indítása előtti lépések a következők:

1. A POST (*power-on self-test*) folyamat az egyszerű hardverhibák azonosítása érdekében a gép bekapcsolásakor azonnal végrehajtódik.
2. A BIOS aktiválja az alapvető komponenseket a rendszer betöltéséhez, mint például a videokimenetet, a billentyűzetet és a tárolóeszközöket.
3. A BIOS betölti a bootloader első szakaszát az MBR-ből (az első eszköz első 440 bájtja a BIOS konfigurációs segédprogramjában meghatározottak szerint).
4. A bootloader első szakasza hívja a bootloader második szakaszát, amely a bootolási opciók megjelenítéséért és a kernel betöltéséért felelős.

Az UEFI, amely az *Unified Extensible Firmware Interface* rövidítése, néhány kulcsfontosságú ponton különbözik a BIOS-tól. A BIOS-hoz hasonlóan az UEFI is egy firmware, de képes azonosítani a partíciókat és olvasni számos, azokban található fájlrendszert. Az UEFI nem támaszkodik az MBR-re, csak az alaplaphoz csatlakoztatott non-volatile memóriájában (*NVRAM*) tárolt beállításokat veszi figyelembe. Ezek a definíciók jelzik az UEFI-kompatibilis programok, az úgynevezett *EFI-alkalmazások* helyét, amelyek automatikusan végrehajtódnak vagy a rendszerindítási menüből hívhatók. Az EFI-alkalmazások lehetnek bootloaderek, operációs rendszer kiválasztók, a rendszer diagnosztikájához és javításához szükséges eszközök stb. Ezeknek egy hagyományos tárolóeszköz partíciójában és egy kompatibilis fájlrendszerben kell lenniük. A szabványos kompatibilis fájlrendszerek a FAT12, FAT16 és FAT32 a blokkos eszközök esetén és az ISO-9660 az optikai adathordozók esetén. Ez a megközelítés sokkal kifinomultabb eszközök megvalósítását teszi lehetővé, mint amire a BIOS lehetőséget nyújt.

Az EFI-alkalmazásokat tartalmazó partíciót *EFI rendszerpartíciónak* vagy egyszerűen ESP-nek nevezik. Ezt a partíciót nem szabad megosztani más rendszer-fájlrendszerekkel, például a gyökérfájlrendszerrel vagy a felhasználói adatfájlrendszerekkel. Az ESP partícióban található EFI mappa tartalmazza azokat az alkalmazásokat, amelyekre az NVRAM-ba mentett bejegyzések mutatnak.

Általánosságban elmondható, hogy az UEFI-vel rendelkező rendszerek esetében a rendszerindítás előtti lépések a következők:

1. A POST (*power-on self-test*) folyamat az egyszerű hardverhibák azonosítása érdekében a gép bekapcsolásakor azonnal végrehajtódik.
2. Az UEFI aktiválja az alapvető komponenseket a rendszer betöltéséhez, például a videokimenetet, a billentyűzetet és a tárolóeszközöket.
3. Az UEFI firmware beolvassa az NVRAM-ban tárolt definíciókat az ESP partíció fájlrendszerében tárolt, előredefiniált EFI-alkalmazás futtatásához. Általában az előredefiniált EFI-alkalmazás egy bootloader.
4. Ha az előre definiált EFI-alkalmazás egy bootloader, akkor az operációs rendszer indításához betölti a kernelt.

Az UEFI-szabvány támogatja a *Secure Boot* nevű funkciót is, amely csak a hardver gyártója által engedélyezett, azaz aláírt EFI-alkalmazások futtatását teszi lehetővé. Ez a funkció növeli a rosszindulatú szoftverekkel szembeni védelmet, de megnehezítheti a gyártó garanciáján kívüli operációs rendszerek telepítését.

A bootloader

Az x86-os architektúrájú Linux legnépszerűbb bootloadere a GRUB (*Grand Unified Bootloader*). Amint a BIOS vagy az UEFI meghívja, a GRUB megjeleníti a bootolásra rendelkezésre álló operációs rendszerek listáját. Néha a lista nem jelenik meg automatikusan, de előhívható a `Shift` lenyomásával, miközben a GRUB-ot a BIOS hívja. UEFI rendszerekben az `Esc` billentyűt kell használni.

A GRUB menüből kiválasztható, hogy a telepített kernelek közül melyik legyen betöltve, és új paramétereket adhatunk át neki. A legtöbb kernelparaméter az `opció=érték` mintát követi. Néhány a leghasznosabb paraméterek közül:

`acpi`

ACPI-támogatás engedélyezése/letiltása. Az `acpi=off` letiltja az ACPI támogatását.

`init`

Alternatív rendszerinicializálót állít be. Például az `init=/bin/bash` a Bash shell-t állítja be. Ez azt jelenti, hogy egy shell munkamenet közvetlenül a kernel indítási folyamata után fog elindulni.

`systemd.unit`

Beállítja az aktiválandó *systemd* célpontot. Például `systemd.unit=graphical.target`. A systemd elfogadja a *SysV* számára meghatározott numerikus futási szinteket is. Az 1-es futási szint aktiválásához például csak az `1` számot vagy az `S` betűt (a “single” rövidítése) kell kernelparaméterként megadni.

`mem`

Beállítja a rendszer számára rendelkezésre álló RAM mennyiségét. Ez a paraméter a virtuális gépeknél hasznos, hogy korlátozza, mennyi RAM álljon az egyes vendégek rendelkezésére. A `mem=512M` használata 512 megabájtra korlátozza az adott vendégrendszer számára rendelkezésre álló RAM mennyiségét.

`maxcpus`

Korlátozza a rendszer számára látható processzorok (vagy processzormagok) számát szimmetrikus többprocesszoros gépeken. Ez a virtuális gépeknél is hasznos. A `0` érték kikapcsolja a többprocesszoros gépek támogatását, és ugyanolyan hatású, mint a kernel `nosmp` paramétere. A `maxcpus=2` paraméter az operációs rendszer számára elérhető processzorok számát kettőre korlátozza.

`quiet`

Elrejti a legtöbb indítási üzenetet.

`vga`

Kiválasztja a videomódot. A `vga=ask` paraméter megjeleníti a rendelkezésre álló módok listáját, amelyek közül választani lehet.

`root`

Beállítja a rootpartíciót, amely különbözik a bootloaderben előre beállított partíciótól. Például `root=/dev/sda3`.

`rootflags`

A root-fájlrendszer csatolási beállításai.

`ro`

A root-fájlrendszer kezdeti csatolását csak olvashatóvá teszi.

`rw`

Lehetővé teszi az írást a gyökér fájlrendszerbe a kezdeti csatolás során.

A kernel paramétereinek módosítása általában nem szükséges, de hasznos lehet az operációs rendszerrel kapcsolatos problémák észleléséhez és megoldásához. A kernelparamétereket hozzá kell adni az `/etc/default/grub` fájlhoz a `GRUB_CMDLINE_LINUX` sorban, hogy azok az újraindítások során is megmaradjanak. A bootloaderhez új konfigurációs fájlt kell generálni minden egyes `/etc/default/grub` változáskor, ami a `grub-mkconfig -o /boot/grub/grub/grub.cfg` paranccsal érhető el. Ha az operációs rendszer fut, az aktuális munkamenet betöltéséhez használt kernelparaméterek a `/proc/cmdline` fájlban olvashatók.

NOTE A GRUB konfigurálásáról egy későbbi leckében lesz szó.

Rendszerinicializálás

Az operációs rendszer a kernelen kívül más komponensektől is függ, amelyek az elvárt funkciókat biztosítják. Ezen összetevők közül sokan a rendszer inicializálási folyamata során töltődnek be, az egyszerű shellszkriptektől kezdve az összetettebb szolgáltatási programokig. A szkripteket gyakran rövid életű feladatok elvégzésére használják, amelyek a rendszer inicializálási folyamata során futnak le és fejeződnek be. A szolgáltatások, más néven *daemonok*, állandóan aktívak lehetnek, mivel az operációs rendszer belső aspektusaiért lehetnek felelősek.

A legkülönbözőbb tulajdonságokkal rendelkező indítószkriptek és daemonok Linux-disztribúcióba való beépítésének sokfélesége hatalmas, ez történelmileg akadályozta egy olyan egységes megoldás kifejlesztését, amely megfelel az összes Linux-disztribúció karbantartói és felhasználói elvárásainak. Azonban bármilyen eszközt is választanak a disztribúciók karbantartói ezen funkció ellátására, az legalább a rendszerszolgáltatások indítására, leállítására és újraindítására képes. Ezeket a műveleteket gyakran maga a rendszer végzi el például egy szoftverfrissítés után, de a rendszergazdának szinte mindig manuálisan kell újraindítania a szolgáltatást, miután módosította annak konfigurációs fájlját.

A rendszergazda számára is kényelmes, ha a körülményektől függően aktiválhatja a daemonok egy adott csoportját. Lehetővé kell tenni például, hogy a rendszerkarbantartási feladatok elvégzéséhez csak egy minimális szolgáltatáskészletet futtasson.

NOTE Szigorúan véve az operációs rendszer nem más, mint a kernel és annak összetevői, amelyek a hardvert vezérlik és az összes folyamatot irányítják. Az “operációs rendszer” kifejezést azonban szokás lazábban használni, hogy a különböző programok egész csoportját jelölje, amelyek azt a szoftverkörnyezetet alkotják, ahol a felhasználó az alapvető számítási feladatokat elvégezheti.

Az operációs rendszer inicializálása akkor kezdődik, amikor a bootloader betölti a kernelt a RAM-ba. Ezután a kernel átveszi a CPU-t, és elkezdi felismerni és beállítani az operációs rendszer alapvető beállításait, például az alapvető hardverkonfigurációt és a memóriacímzést.

A kernel ezután megnyitja az *initramfs*-t (*inicial RAM filesystem*). Az initramfs egy olyan archívum, amely egy olyan fájlrendszert tartalmaz, amelyet a rendszerindítás során ideiglenes gyökérfájlrendszerként használ. Az initramfs fájl fő célja a szükséges modulok biztosítása, hogy a kernel hozzáférhessen az operációs rendszer "valódi" gyökérfájlrendszeréhez.

Amint a root fájlrendszer elérhetővé válik, a kernel csatlakoztatja az összes, az `/etc/fstab'-ban konfigurált fájlrendszert, majd végrehajtja az első programot, az `init` nevű segédprogramot. Az `init` program felelős az összes inicializáló szkript és rendszerdaemon futtatásáért. Az ilyen rendszerindítóknak a hagyományos init mellett külön implementációi vannak, mint például a *systemd* és az *Upstart*. Az init program betöltése után az initramfs eltávolításra kerül a RAM-ból.

SysV standard

A SysVinit szabványon alapuló szolgáltatáskezelő a *runlevels* (futtatási szintek) fogalmával szabályozza, hogy mely daemonok és erőforrások lesznek elérhetőek. A futási szintek számozása 0-tól 6-ig terjed és a disztribúció karbantartói tervezték meg őket, úgy, hogy meghatározott célokat szolgáljanak. Az egyetlen, minden disztribúcióban közös runlevel definíciók a 0, 1 és 6-os futtatási szintek.

systemd

A systemd egy modern rendszer- és szolgáltatáskezelő, amely kompatibilitási réteggel rendelkezik a SysV parancsokhoz és futtatási szintekhez. A systemd egyidejű struktúrával rendelkezik, socketeket és D-Bus-t használ a szolgáltatások aktiválásához, igény szerinti daemonfuttatást, folyamatfelügyeletet *cgroups* segítségével, snapshot-támogatást, rendszermunkamenet-helyreállítást, csatolási pont ellenőrzést és függőségi alapú szolgáltatás-ellenőrzést. Az elmúlt években a legtöbb nagyobb Linux-disztribúció fokozatosan átvette a systemd-t alapértelmezett rendszerkezelőként.

Upstart

A systemd-hez hasonlóan az Upstart is az init helyettesítője. Az Upstart középpontjában a rendszerindítási folyamat felgyorsítása áll a rendszerszolgáltatások betöltési folyamatának párhuzamosításával. Az Upstartot az Ubuntu alapú disztribúciók használták a korábbi kiadásokban, de mára átadta helyét a systemd-nek.

Az inicializálás ellenőrzése

A rendszerindítás során előfordulhatnak hibák, de ezek nem feltétlenül olyan kritikusak, hogy teljesen leállítsák az operációs rendszert. Ettől függetlenül ezek a hibák veszélyeztethetik a rendszer elvárt viselkedését. Minden hiba olyan üzeneteket eredményez, amelyek felhasználhatók a későbbi vizsgálatokhoz, mivel értékes információkat tartalmaznak arról, hogy mikor és hogyan következett be a hiba. Még ha nem is keletkeznek hibaüzenetek, a rendszerindítási folyamat során gyűjtött információk hasznosak lehetnek finomhangolási és konfigurációs célokra.

A memóriaterületet, ahol a kernel az üzeneteit tárolja, beleértve a boot-üzeneteket is, *kernel ring buffer*-nek nevezik. Az üzenetek akkor is a kernel gyűrűpufferben maradnak, ha az inicializálási folyamat során nem jelennek meg, például ha helyette egy animáció jelenik meg. A kernel gyűrűpuffer azonban elveszíti az összes üzenetet, amikor a rendszert kikapcsoljuk, vagy a `dmesg --clear` parancs végrehajtásákor. Kapcsolók nélkül a `dmesg` parancs megjeleníti a kernel gyűrűpufferben lévő aktuális üzeneteket:

```
$ dmesg
[    5.262389] EXT4-fs (sda1): mounted filesystem with ordered data mode. Opts: (null)
[    5.449712] ip_tables: (C) 2000-2006 Netfilter Core Team
[    5.460286] systemd[1]: systemd 237 running in system mode.
[    5.480138] systemd[1]: Detected architecture x86-64.
[    5.481767] systemd[1]: Set hostname to <torre>.
[    5.636607] systemd[1]: Reached target User and Group Name Lookups.
[    5.636866] systemd[1]: Created slice System Slice.
[    5.637000] systemd[1]: Listening on Journal Audit Socket.
[    5.637085] systemd[1]: Listening on Journal Socket.
[    5.637827] systemd[1]: Mounting POSIX Message Queue File System...
[    5.638639] systemd[1]: Started Read required files in advance.
[    5.641661] systemd[1]: Starting Load Kernel Modules...
[    5.661672] EXT4-fs (sda1): re-mounted. Opts: errors=remount-ro
[    5.694322] lp: driver loaded but no devices found
[    5.702609] ppdev: user-space parallel port driver
[    5.705384] parport_pc 00:02: reported by Plug and Play ACPI
[    5.705468] parport0: PC-style at 0x378 (0x778), irq 7, dma 3
[PCSPP,TRISTATE,COMPAT,EPP,ECP,DMA]
[    5.800146] lp0: using parport0 (interrupt-driven).
[    5.897421] systemd-journald[352]: Received request to flush runtime journal from PID 1
```

A `dmesg` kimenete több száz sor hosszú is lehet, ezért az előző felsorolás csak azt a kivonatot tartalmazza, amely a kernel systemd szolgáltatáskezelő hívását jeleníti meg. A sorok elején lévő

értékek a kernel betöltésének kezdetéhez viszonyított másodpercek számát jelentik.

A systemd alapú rendszerekben a `journalctl` parancs megjeleníti az inicializálási üzeneteket a `-b`, `--boot`, `-k` vagy `--dmesg` opciókkal. A `journalctl --list-boots` parancs megjeleníti az aktuális boothoz viszonyított boot-számok listáját, az azonosító hash-üket és az első és utolsó üzenet időbélyegét:

```
$ journalctl --list-boots
 -4 9e5b3eb4952845208b841ad4dbefa1a6 Thu 2019-10-03 13:39:23 -03—Thu 2019-10-03 13:40:30 -03
 -3 9e3d79955535430aa43baa17758f40fa Thu 2019-10-03 13:41:15 -03—Thu 2019-10-03 14:56:19 -03
 -2 17672d8851694e6c9bb102df7355452c Thu 2019-10-03 14:56:57 -03—Thu 2019-10-03 19:27:16 -03
 -1 55c0d9439bfb4e85a20a62776d0dbb4d Thu 2019-10-03 19:27:53 -03—Fri 2019-10-04 00:28:47 -03
  0 08fbbebd9f964a74b8a02bb27b200622 Fri 2019-10-04 00:31:01 -03—Fri 2019-10-04 10:17:01 -03
```

A régebbi inicializálási logok a systemd-alapú rendszerekben is megmaradnak, így a korábbi operációs rendszer munkamenet-üzenetei továbbra is ellenőrizhetők. Ha a `-b 0` vagy `--boot=0` opciókat adjuk meg, akkor az aktuális rendszerindításhoz tartozó üzenetek jelennek meg. A `-b -1` vagy `--boot=-1` opciók az előző inicializálás üzeneteit mutatják. A `-b -2` vagy `--boot=-2` opciók az azt megelőző inicializálás üzeneteit mutatják, és így tovább. A következő részlet azt jeleníti meg, hogy a kernel az utolsó inicializálási folyamathoz hívja a systemd szolgáltatáskezelőt:

```
$ journalctl -b 0
oct 04 00:31:01 ubuntu-host kernel: EXT4-fs (sda1): mounted filesystem with ordered data
mode. Opts: (null)
oct 04 00:31:01 ubuntu-host kernel: ip_tables: (C) 2000-2006 Netfilter Core Team
oct 04 00:31:01 ubuntu-host systemd[1]: systemd 237 running in system mode.
oct 04 00:31:01 ubuntu-host systemd[1]: Detected architecture x86-64.
oct 04 00:31:01 ubuntu-host systemd[1]: Set hostname to <torre>.
oct 04 00:31:01 ubuntu-host systemd[1]: Reached target User and Group Name Lookups.
oct 04 00:31:01 ubuntu-host systemd[1]: Created slice System Slice.
oct 04 00:31:01 ubuntu-host systemd[1]: Listening on Journal Audit Socket.
oct 04 00:31:01 ubuntu-host systemd[1]: Listening on Journal Socket.
oct 04 00:31:01 ubuntu-host systemd[1]: Mounting POSIX Message Queue File System...
oct 04 00:31:01 ubuntu-host systemd[1]: Started Read required files in advance.
oct 04 00:31:01 ubuntu-host systemd[1]: Starting Load Kernel Modules...
oct 04 00:31:01 ubuntu-host kernel: EXT4-fs (sda1): re-mounted. Opts:
commit=300,barrier=0,errors=remount-ro
oct 04 00:31:01 ubuntu-host kernel: lp: driver loaded but no devices found
oct 04 00:31:01 ubuntu-host kernel: ppdev: user-space parallel port driver
oct 04 00:31:01 ubuntu-host kernel: parport_pc 00:02: reported by Plug and Play ACPI
oct 04 00:31:01 ubuntu-host kernel: parport0: PC-style at 0x378 (0x778), irq 7, dma 3
[PCSPP,TRISTATE,COMPAT,EPP,ECP,DMA]
```

```
oct 04 00:31:01 ubuntu-host kernel: lp0: using parport0 (interrupt-driven).
oct 04 00:31:01 ubuntu-host systemd-journald[352]: Journal started
oct 04 00:31:01 ubuntu-host systemd-journald[352]: Runtime journal
(/run/log/journal/abb765408f3741ae9519ab3b96063a15) is 4.9M, max 39.4M, 34.5M free.
oct 04 00:31:01 ubuntu-host systemd-modules-load[335]: Inserted module 'lp'
oct 04 00:31:01 ubuntu-host systemd-modules-load[335]: Inserted module 'ppdev'
oct 04 00:31:01 ubuntu-host systemd-modules-load[335]: Inserted module 'parport_pc'
oct 04 00:31:01 ubuntu-host systemd[1]: Starting Flush Journal to Persistent Storage...
```

Az operációs rendszer által kiadott inicializálási és egyéb üzeneteket a `/var/log/` mappában található fájlokban tárolja. Ha kritikus hiba történik, és az operációs rendszer a kernel és az initramfs betöltése után nem tudja folytatni az inicializálási folyamatot, akkor egy alternatív boot mediumot lehet használni a rendszer indítására és a megfelelő fájlrendszer elérésére. Ezután a `/var/log/` alatt található fájlokat át lehet kutatni a rendszerindítási folyamat megszakadását okozó lehetséges okok után. A `journalctl` parancs `-D` vagy `--directory` opcióival a `/var/log/journal/` mappán kívüli mappákban lévő logüzenetek olvashatók, ami a systemd naplóüzenetek alapértelmezett helye. Mivel a systemd logüzenetei nem nyers szövegben vannak tárolva, a `journalctl` parancsra van szükség a beolvasásukhoz.

Gyakorló feladatok

1. Egy BIOS firmware-rel ellátott gépen hol található a bootstrap bináris állomány?

2. Az UEFI firmware támogatja a külső programok, az úgynevezett EFI-alkalmazások által biztosított kibővített funkciókat. Ezeknek az alkalmazásoknak azonban saját speciális helyük van. Hol helyezkednek el a rendszerben az EFI-alkalmazások?

3. A bootloaderek lehetővé teszik az egyéni kernel paraméterek átadását a betöltés előtt. Tegyük fel, hogy a rendszer nem tud elindulni a gyökérfájlrendszer helyének téves tájékoztatása miatt. Hogyan lehetne a helyes root fájlrendszert, amely a `/dev/sda3` címen található, paraméterként megadni a kernelnek?

4. A Linux boot folyamata a következő üzenettel végződik:

   ```
   ALERT! /dev/sda3 does not exist. Dropping to a shell!
   ```

   Mi a probléma legvalószínűbb oka?

Gondolkodtató feladatok

1. A bootloader megjeleníti az operációs rendszerek listáját, amelyek közül választhatunk, ha egynél több operációs rendszer van telepítve a gépre. Egy újonnan telepített operációs rendszer azonban felülírhatja a merevlemez MBR-jét, ami törli a bootloader első szakaszát, és a másik operációs rendszert elérhetetlenné teszi. Miért nem történhet ez meg egy UEFI firmware-es gépen?

2. Mi a gyakori következménye annak, ha egy egyéni kernelt úgy telepítünk, hogy nem adunk meg megfelelő initramfs imaget?

3. Az inicializálási log több száz sor hosszú, ezért a `dmesg` parancs kimenete gyakran átkerül egy pager parancsba — ilyen például a `less` parancs --, az olvasás megkönnyítése érdekében. Melyik `dmesg` opció fogja automatikusan oldalszámozni a kimenetet, így nem lesz szükség a pager parancs explicit használatára?

4. Egy offline gép teljes fájlrendszerét tartalmazó merevlemezt eltávolítottak és másodlagos meghajtóként egy működő géphez csatlakoztattak. Feltételezve, hogy a csatolási pont a `/mnt/hd`, hogyan lehetne a `journalctl` segítségével megvizsgálni a `/mnt/hd/var/log/journal/` címen található logfájlok tartalmát?

Összefoglalás

Ez a lecke egy szabványos Linux rendszer indítási sorrendjét tárgyalja. A Linux-rendszer indítási folyamatának megfelelő ismerete segít megelőzni a hibákat, amelyek miatt a rendszer elérhetetlenné válhat. A lecke a következő témaköröket járja körül:

- Miben különböznek a BIOS és az UEFI rendszerindítási módszerek.
- A rendszer tipikus inicializálási fázisai.
- A rendszerindítási üzenetek helyreállítása.

A tárgyalt parancsok és eljárások a következők voltak:

- Közös kernel paraméterek.
- Boot üzenetek olvasására szolgáló parancsok: `dmesg` és a `journalctl`.

Válaszok a gondolkodtató feladatokra

1. Egy BIOS firmware-rel ellátott gépen hol található a bootstrap bináris állomány?

   Az első tárolóeszköz MBR-ében, a BIOS konfigurációs segédprogramjában meghatározottak szerint.

2. Az UEFI firmware támogatja a külső programok, az úgynevezett EFI-alkalmazások által biztosított kibővített funkciókat. Ezeknek az alkalmazásoknak azonban saját speciális helyük van. Hol helyezkednek el a rendszerben az EFI-alkalmazások?

   Az EFI-alkalmazások az EFI rendszerpartícióban (ESP) tárolódnak, amely bármely kompatibilis fájlrendszerrel (általában FAT32 fájlrendszerrel) rendelkező szabad tárolóblokkban található.

3. A bootloaderek lehetővé teszik az egyéni kernel paraméterek átadását a betöltés előtt. Tegyük fel, hogy a rendszer nem tud elindulni a gyökérfájlrendszer helyének téves tájékoztatása miatt. Hogyan lehetne a helyes root fájlrendszert, amely a `/dev/sda3` címen található, paraméterként megadni a kernelnek?

   A `root` paramétert kell használni, mint például `root=/dev/sda3`.

4. A Linux boot folyamata a következő üzenettel végződik:

   ```
   ALERT! /dev/sda3 does not exist. Dropping to a shell!
   ```

   Mi a probléma legvalószínűbb oka?

   A kernel nem találta a root fájlrendszerként feltüntetett `/dev/sda3` eszközt.

Válaszok a gondolkodtató feladatokra

1. A bootloader megjeleníti az operációs rendszerek listáját, amelyek közül választhatunk, ha egynél több operációs rendszer van telepítve a gépre. Egy újonnan telepített operációs rendszer azonban felülírhatja a merevlemez MBR-jét, ami törli a bootloader első szakaszát, és a másik operációs rendszert elérhetetlenné teszi. Miért nem történhet ez meg egy UEFI firmware-es gépen?

   Az UEFI-s számítógépek nem használják a merevlemez MBR-jét a bootloader első szakaszának tárolására.

2. Mi a gyakori következménye annak, ha egy egyéni kernelt úgy telepítünk, hogy nem adunk meg megfelelő initramfs imaget?

   A root fájlrendszer elérhetetlenné válhat, ha a típusát külső kernelmodulként fordították le.

3. Az inicializálási log több száz sor hosszú, ezért a `dmesg` parancs kimenete gyakran átkerül egy pager parancsba — ilyen például a `less` parancs --, az olvasás megkönnyítése érdekében. Melyik `dmesg` opció fogja automatikusan oldalszámozni a kimenetet, így nem lesz szükség a lapozó parancs explicit használatára?

   A `dmesg -H` vagy a `dmesg --human` parancsok alapértelmezetté teszik a lapozást.

4. Egy offline gép teljes fájlrendszerét tartalmazó merevlemezt eltávolítottak és másodlagos meghajtóként egy működő géphez csatlakoztattak. Feltételezve, hogy a csatolási pont a `/mnt/hd`, hogyan lehetne a `journalctl` segítségével megvizsgálni a `/mnt/hd/var/log/journal/` címen található logfájlok tartalmát?

   A `journalctl -D /mnt/hd/var/log/journal` vagy a `journalctl --directory=/mnt/hd/var/log/journal` parancsokkal.

101.3 Futási szintek megváltoztatása / boot targetek és a rendszer leállítása vagy újraindítása

Hivatkozás az LPI célkitűzésre

LPIC-1 v5, Exam 101, Objective 101.3

Súlyozás

3

Kulcsfontosságú ismeretek

- Az alapértelmezett futtatási szint vagy indítási target beállítása.
- Váltás a futtatási szintek/indítási targetek között, beleértve az egyfelhasználós üzemmódot is.
- Leállítás és újraindítás a parancssorból.
- A felhasználók figyelmeztetése a futtatási szintek/indítási targetek váltása vagy más fontos rendszeresemények előtt.
- Folyamatok megfelelő befejezése.
- Az acpid ismerete.

A használt fájlok, kifejezések és segédprogramok listája

- `/etc/inittab`
- `shutdown`
- `init`
- `/etc/init.d/`
- `telinit`
- `systemd`

- `systemctl`
- `/etc/systemd/`
- `/usr/lib/systemd/`
- `wall`

101.3 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	101 Rendszerarchitektúra
Fejezet:	101.3 Futási szintek megváltoztatása / boot targetek és a rendszer leállítása vagy újraindítása
Lecke:	1/1

Bevezetés

A Unix tervezési elveit követő operációs rendszerek közös jellemzője, hogy a rendszer különböző funkcióit különálló folyamatok vezérlik. Ezek a *daemonoknak* (vagy általánosabban *szolgáltatásoknak* (service)) nevezett folyamatok felelősek az operációs rendszer alapjául szolgáló olyan kiterjesztett funkciókért is, mint a hálózati alkalmazás-szolgáltatások (HTTP-kiszolgáló, fájlmegosztás, e-mail stb.), adatbázisok, igény szerinti konfiguráció stb. Bár a Linux monolitikus kernelt használ, az operációs rendszer számos alacsony szintű aspektusát befolyásolják a daemonok, például a terheléselosztást és a tűzfal konfigurálását.

A rendszer céljától függ, hogy mely daemonok legyenek aktívak. Az aktív daemonok halmazának futás közben is módosíthatónak kell lennie, így a szolgáltatások az egész rendszer újraindítása nélkül elindíthatók és leállíthatók. Ennek a problémának a megoldására minden nagyobb Linux-disztribúció kínál valamilyen szolgáltatáskezelő segédprogramot a rendszer vezérlésére.

A szolgáltatások vezérelhetők shell szkriptekkel vagy egy programmal és az azt támogató konfigurációs fájlokkal. Az első módszert a *SysVinit* szabvány, más néven *System V* vagy

egyszerűen csak *SysV* valósítja meg. A második módszert a *systemd* és az *Upstart* valósítja meg. Történelmileg a Linux disztribúciók leggyakrabban a SysV alapú szolgáltatáskezelőket használták. Ma már a legtöbb Linux-disztribúcióban gyakrabban találunk systemd-alapú szolgáltatáskezelőket. A szolgáltatáskezelő az első program, amelyet a kernel a boot folyamat során elindít, ezért a PID-je (process identification number - processz-azonosító szám) mindig `1`.

SysVinit

A SysVinit szabványon alapuló szolgáltatáskezelő a rendszerállapotok előre definiált készleteit, úgynevezett *runleveleket* (futási szinteket) és a hozzájuk tartozó, végrehajtandó szolgáltatási szkriptfájlokat biztosít. A futási szintek számozása `0`-tól `6`-ig tart, és általában a következő célokat szolgálják:

Runlevel 0

A rendszer leállítása.

Runlevel 1, s vagy single

Egyfelhasználós üzemmód, hálózati és egyéb nem létfontosságú tulajdonságok nélkül (karbantartási üzemmód).

Runlevel 2, 3 vagy 4

Többfelhasználós üzemmód. A felhasználók bejelentkezhetnek konzolon vagy hálózaton keresztül. A 2-es és 4-es futási szinteket nem gyakran használják.

Runlevel 5

Többfelhasználós üzemmód. Ez megegyezik a 3-assal, plusz rendelkezik grafikus bejelentkezéssel.

Runlevel 6

A rendszer újraindítása.

A futási szintek és a kapcsolódó daemonok/források kezeléséért felelős program az `/sbin/init`. A rendszer inicializálása során az `init` program azonosítja a kernelparaméterrel vagy az `/etc/inittab` fájlban meghatározott kért futási szintet, és betölti az adott futási szinthez tartozó, ott felsorolt szkripteket. Minden futási szinthez számos szolgáltatási fájl tartozhat, általában az `/etc/init.d/` mappában található szkriptek. Mivel nem minden runlevel egyenértékű a különböző Linux disztribúciókban, a SysV alapú disztribúciókban is megtalálható a futási szintek céljának rövid leírása.

Az `/etc/inittab` fájl szintaxisának formátuma az alábbi:

```
id:runlevels:action:process
```

Az `id` egy maximum négy karakter hosszúságú összefoglaló név, amely a bejegyzés azonosítására szolgál. A `runlevels` bejegyzés azon futási szintek számainak listája, amelyeknél a megadott műveletet végre kell hajtani. Az `action` kifejezés határozza meg, hogy az `init` hogyan fogja végrehajtani a `process` kifejezés által megjelölt folyamatot. A rendelkezésre álló műveletek a következők:

`boot`

A folyamat a rendszer inicializálása során kerül végrehajtásra. A `runlevels` mező itt nem számít, figyelmen kívül marad.

`bootwait`

A folyamat a rendszer inicializálása közben fog végrehajtódni és az `init` várni fog a folyamat befejezéséig. A `runlevels` mező itt is figyelmen kívül marad.

`sysinit`

A folyamat a rendszer inicializálása után fog végrehajtódni, függetlenül a futási szinttől. A `runlevels` mező itt is figyelmen kívül marad.

`wait`

A folyamat a megadott futtatási szinteken fog végrehajtódni, és az `init` várni fog a folytatással, amíg befejeződik.

`respawn`

A folyamat újraindul, ha terminálódik.

`ctrlaltdel`

A folyamat akkor fog végrehajtódni, amikor az init folyamat megkapja a `SIGINT` jelet, amelyet a `Ctrl` + `Alt` + `Del` billentyűkombináció lenyomása vált ki.

Az alapértelmezett futási szint — az, amelyik kiválasztásra kerül, ha nincs más kernelparaméter megadva — szintén az `/etc/inittab` állományban van megadva, az `id:x:initdefault` bejegyzésben. Az `x` az alapértelmezett futási szint száma. Ez a szám soha nem lehet `0` vagy `6`, mivel ez a rendszer leállását vagy újraindítását eredményezné, amint befejezi a rendszerindítási folyamatot. Egy tipikus `/etc/inittab` fájl az alábbiakban látható (a zárójelben lévő magyar fordítás természetesen nem része a fájlnak):

```
# Default runlevel (alapértelmezett futási szint)
id:3:initdefault:
```

```
# Configuration script executed during boot (A rendszerindítás során végrehajtott
konfigurációs szkript)
si::sysinit:/etc/init.d/rcS

# Action taken on runlevel S (single user) (S futási szinten végrehajtott művelet (egyetlen
felhasználó))
~:S:wait:/sbin/sulogin

# Configuration for each execution level (Konfiguráció az egyes végrehajtási szintekhez)
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6

# Action taken upon ctrl+alt+del keystroke (Ctrl+alt+del billentyűkombináció után
végrehajtott művelet)
ca::ctrlaltdel:/sbin/shutdown -r now

# Enable consoles for runlevels 2 and 3 (konzolok engedélyezése a 2-es és 3-as futási
szintekhez)
1:23:respawn:/sbin/getty tty1 VC linux
2:23:respawn:/sbin/getty tty2 VC linux
3:23:respawn:/sbin/getty tty3 VC linux
4:23:respawn:/sbin/getty tty4 VC linux

# For runlevel 3, also enable serial (3-as futási szinthez, serial engedélyezése)
# terminals ttyS0 and ttyS1 (modem) consoles (ttyS0 terminálok és ttyS1 (modem) konzolok)
S0:3:respawn:/sbin/getty -L 9600 ttyS0 vt320
S1:3:respawn:/sbin/mgetty -x0 -D ttyS1
```

A `telinit q` parancsot minden alkalommal le kell futtatni, miután az `/etc/inittab` fájl módosításra került. A `q` (vagy `Q`) argumentum jelzi az init-nek, hogy töltse újra a konfigurációt. Ez a lépés azért fontos, hogy elkerüljük a rendszer leállását a `/etc/inittab` hibás konfigurációja miatt.

Az `init` által az egyes futási szintek beállításához használt szkriptek az `/etc/init.d/` mappában találhatók. Minden futási szintnek van egy kapcsolódó mappája az `/etc/` mappában, amelynek neve `/etc/rc0.d/`, `/etc/rc1.d/`, `/etc/rc2.d/`, stb., és amely tartalmazza azokat a szkripteket, amelyeket a megfelelő futási szint indításakor végre kell hajtani. Mivel ugyanazt a szkriptet

különböző futási szintek használhatják, az ezekben a mappákban lévő fájlok csak szimbolikus hivatkozások a `/etc/init.d/` mappában lévő tényleges szkriptekre. Továbbá a futási szint mappájában lévő link fájlnevének első betűje jelzi, hogy a szolgáltatást a megfelelő futási szinthez tartozóan kell-e elindítani vagy megszüntetni. A link `K` betűvel kezdődő fájlneve azt határozza meg, hogy a szolgáltatás a runlevelbe való belépéskor meg fog szűnni (kill). Az `S` betűvel kezdődő hivatkozás azt jelenti, hogy a szolgáltatás a futási szintre való belépéskor elindul (start). Az `/etc/rc1.d/` mappában például sok, `K` betűvel kezdődő hivatkozás található hálózati szkriptekre, figyelembe véve, hogy az 1-es futási szint a single felhasználó futási szintje, hálózati kapcsolat nélkül.

A `runlevel` parancs megmutatja a rendszer aktuális futási szintjét. A `runlevel` parancs két értéket mutat, az első az előző futási szint, a második pedig az aktuális futási szint:

```
$ runlevel
N 3
```

Az `N` betű a kimeneten azt mutatja, hogy a futási szint nem változott a legutóbbi indítás óta. A példában szereplő `runlevel 3` a rendszer jelenlegi futási szintje.

Ugyanez az `init` program használható a futó rendszer futási szintjei közötti váltásra, újraindítás nélkül. A `telinit` parancs is használható a futási szintek közötti váltásra. Például a `telinit 1`, `telinit s` vagy `telinit S` parancsok az 1-es futási szintre váltanak.

systemd

Jelenleg a systemd a legszélesebb körben használt eszközkészlet a rendszer erőforrásainak és szolgáltatásainak kezelésére, amelyeket a systemd *units*-nak nevez. Egy unit (egység) egy névből, egy típusból és egy hozzá tartozó konfigurációs fájlból áll. Például egy *httpd* szerverfolyamat (mint például az Apache webkiszolgáló) egysége a Red Hat alapú disztribúciókban `httpd.service`, és a konfigurációs fájl neve is `httpd.service` (Debian alapú disztribúciókban ezen unit neve `apache2.service`).

A systemd-egységeknek hét különböző típusa létezik:

`service`

A leggyakoribb egységtípus, aktív rendszererőforrásokhoz, amelyek kezdeményezhetők, megszakíthatók és újratölthetők.

`socket`

A socket egységtípus lehet fájlrendszeri vagy hálózati socket. Minden socket egységnek van egy

megfelelő szolgáltatási egysége, amely akkor töltődik be, amikor a socket kérést kap.

`device`

A device (eszköz) egység a kernel által azonosított hardvereszközhöz kapcsolódik. Egy device csak akkor lesz systemd egységnek véve, ha létezik erre a célra udev szabály. Egy device egység használható a konfigurációs függőségek feloldására, amikor bizonyos hardvereket észlel, feltéve, hogy az udev szabály tulajdonságai a device egység paramétereiként használhatók.

`mount`

A mount unit egy csatolási pont definíciója a fájlrendszerben, hasonlóan a `/etc/fstab` bejegyzéshez.

`automount`

Az automount egység szintén egy csatolási pontdefiníció a fájlrendszerben, de automatikusan csatlakoztatva. Minden automount egységhez tartozik egy megfelelő mount egység, amely akkor indul el, amikor az automount csatolási ponthoz hozzáférünk.

`target`

A target egység más egységek csoportosítása, egyetlen egységként kezelve.

`snapshot`

A snapshot egység a systemd menedzser elmentett állapota (nem minden Linux disztribúción érhető el).

A systemd egységek vezérlésére szolgáló fő parancs a `systemctl`. A `systemctl` parancsot az egységek aktiválásával, deaktiválásával, végrehajtásával, megszakításával, felügyeletével stb. kapcsolatos összes feladat végrehajtására használjuk. Egy `unit.service` nevű fiktív egység esetében például a leggyakoribb `systemctl` műveletek a következők lesznek:

`systemctl start unit.service`

Elindítja a `unit`-ot.

`systemctl stop unit.service`

Leállítja a `unit`-ot.

`systemctl restart unit.service`

Újraindítja a `unit`-ot.

`systemctl status unit.service`

Kiírja a `unit` állapotát, beleértve, hogy fut-e vagy sem.

`systemctl is-active unit.service`

Kiírja, hogy *active*, ha a `unit` fut, ellenkező esetben pedig azt, hogy inactive.

`systemctl enable unit.service`

Engedélyezi a `unit`-ot, azaz a `unit` betöltődik a rendszer inicializálása során.

`systemctl disable unit.service`

A `unit` nem indul el a rendszerrel.

`systemctl is-enabled unit.service`

Ellenőrzi, hogy a `unit` elindul-e a rendszerrel. A választ a `$?` változóban tárolja. A `0` érték azt jelzi, hogy a `unit` elindul a rendszerrel és az `1` azt, hogy a `unit` nem indul el a rendszerrel.

NOTE

A systemd újabb telepítései ténylegesen listázzák a unit konfigurációját a rendszerindításkor. Például:

```
$ systemctl is-enabled apparmor.service
enabled
```

Ha a rendszerben nincs más azonos nevű egység, akkor a pont utáni utótag elhagyható. Ha például csak egy `httpd` típusú `service` egység létezik, akkor csak a `httpd` elég a `systemctl` egység paramétereként.

A `systemctl` parancs a *rendszercélokat* is vezérelheti. A `multi-user.target` egység például egyesíti a többfelhasználós rendszerkörnyezethez szükséges összes egységet. Ez hasonló a SysV-t használó rendszerben a 3-as futási szinthez.

A `systemctl isolate` parancs váltogatja a különböző célokat (target). A `multi-user` célra történő manuális váltás:

```
# systemctl isolate multi-user.target
```

Vannak SysV futtatási szintekhez tartozó célok (targets), kezdve a `runlevel0.target`-től egészen a `runlevel6.target`-ig. A systemd azonban nem használja az `/etc/inittab` fájlt. Az alapértelmezett rendszercél (sysrem tagret) megváltoztatásához a `systemd.unit` opciót hozzáadhatjuk a kernel paramétereinek listájához. Például, ha a `multi-user.target`-t szeretnénk használni alapértelmezett célként, a kernel paramétere a `systemd.unit=multi-user.target` kell legyen. Minden kernelparaméter állandósítható a bootloader konfigurációjának módosításával.

Az alapértelmezett cél megváltoztatásának másik módja, ha módosítjuk a `/etc/systemd/system/default.target` szimbolikus linket, hogy az a kívánt célra mutasson. A link átdefiniálása önmagában a `systemctl` paranccsal is elvégezhető:

```
# systemctl set-default multi-user.target
```

Hasonlóképpen, a következő paranccsal meghatározhatjuk, hogy mi a rendszer alapértelmezett rendszerindítási célja:

```
$ systemctl get-default
graphical.target
```

A SysV-t alkalmazó rendszerekhez hasonlóan az alapértelmezett cél soha nem mutathat a `shutdown.target` címre, mivel az a 0 futási szintnek (shutdown) felel meg.

Az egyes egységekhez tartozó konfigurációs fájlok a `/lib/systemd/system/` mappában találhatók. A `systemctl list-unit-files` parancs felsorolja az összes elérhető egységet, és megmutatja, hogy engedélyezve van-e az indításuk a rendszer indításakor. A `--type` kapcsoló csak az adott típushoz tartozó egységeket választja ki, mint a `systemctl list-unit-files --type=service` és a `systemctl list-unit-files --type=target` parancsok esetén.

Az aktív egységek vagy az aktuális rendszermunkamenet alatti aktív egységek a `systemctl list-units` paranccsal listázhatók. A `list-unit-files` opcióhoz hasonlóan a `systemctl list-units --type=service` parancs csak a `service` típusú egységeket, a `systemctl list-units --type=target` parancs pedig csak a `target` típusú egységeket választja ki.

A systemd felelős az energiával kapcsolatos események kiváltásáért és az azokra való reagálásért is. A `systemctl suspend` parancs a rendszert alacsony energiafogyasztású üzemmódba helyezi, az aktuális adatokat a memóriában tartja. A `systemctl hibernate` parancs az összes memóriaadatot lemezre másolja, így a rendszer aktuális állapota a kikapcsolás után visszaállítható. Az ilyen eseményekhez kapcsolódó műveleteket az `/etc/systemd/logind.conf` fájlban vagy az `/etc/systemd/logind.conf.d.d/` mappán belüli különálló fájlokban határozzuk meg. Ez a systemd funkció azonban csak akkor használható, ha a rendszerben nem fut más energiagazdálkodó, például az `acpid` daemon. Az `acpid` daemon a Linux fő energiakezelője, és finomabb beállításokat tesz lehetővé az energiával kapcsolatos eseményeket követő műveletekben, mint például a laptop fedelének becsukása, az alacsony akkumulátor-szint vagy az akkumulátor töltöttségi szintje.

Upstart

Az Upstart által használt inicializálási szkriptek az `/etc/init/` mappában találhatók. A rendszerszolgáltatásokat az `initctl list` paranccsal lehet listázni, amely a szolgáltatások aktuális állapotát és, ha rendelkezésre áll, a PID számukat is megjeleníti.

```
# initctl list
avahi-cups-reload stop/waiting
avahi-daemon start/running, process 1123
mountall-net stop/waiting
mountnfs-bootclean.sh start/running
nmbd start/running, process 3085
passwd stop/waiting
rc stop/waiting
rsyslog start/running, process 1095
tty4 start/running, process 1761
udev start/running, process 1073
upstart-udev-bridge start/running, process 1066
console-setup stop/waiting
irqbalance start/running, process 1842
plymouth-log stop/waiting
smbd start/running, process 1457
tty5 start/running, process 1764
failsafe stop/waiting
```

Minden Upstart-műveletnek saját, független parancsa van. Például a `start` parancs használható egy hatodik virtuális terminál elindítására:

```
# start tty6
```

Az erőforrás aktuális állapota a `status` paranccsal ellenőrizhető:

```
# status tty6
tty6 start/running, process 3282
```

A szolgáltatás megszakítása pedig a `stop` paranccsal történik:

```
# stop tty6
```

Az Upstart nem használja az `/etc/inittab` fájlt a futási szintek meghatározására, de a `runlevel`

és a `telinit` parancsok továbbra is használhatók a futási szintek ellenőrzésére és váltogatására.

NOTE Az Upstartot az Ubuntu Linux disztribúcióhoz fejlesztették, hogy megkönnyítse a folyamatok párhuzamos indítását. Az Ubuntu 2015-ben a systemd-re váltott, azóta nem használja az Upstartot.

Leállítás és újraindítás

A rendszer leállítására vagy újraindítására használt hagyományos parancs neve nem meglepő módon `shutdown`. A `shutdown` parancs extra funkciókkal egészíti ki a kikapcsolási folyamatot: automatikusan figyelmeztető üzenettel értesíti az összes bejelentkezett felhasználót a shell munkamenetükben, és megakadályozza az új bejelentkezéseket. A `shutdown` parancs a SysV vagy systemd eljárások közvetítőjeként működik, azaz a rendszer által elfogadott szolgáltatáskezelő megfelelő műveletének meghívásával hajtja végre a kért műveletet.

A `shutdown` végrehajtása után minden folyamat megkapja a `SIGTERM` jelet, amelyet a `SIGKILL` jel követ, majd a rendszer leáll vagy megváltoztatja a futási szintjét. Alapértelmezés szerint, ha sem a `-h`, sem az `-r` opciót nem használjuk, a rendszer az 1-es futási szintre, azaz az egyfelhasználós üzemmódra vált. A `shutdown` alapértelmezett beállításainak módosításához a parancsot a következő szintaxissal kell végrehajtani:

```
$ shutdown [option] time [message]
```

Csak a `time` paraméter elvárt. A `time` paraméter határozza meg, hogy a kért művelet mikor kerül végrehajtásra, a következő formátumokat fogadva el:

`hh:mm`
: Ez a formátum a végrehajtási időt órában és percben adja meg.

`+m`
: Ez a formátum megadja, hogy hány percet kell várni a végrehajtás előtt.

`now` or `+0`
: Ez a formátum az azonnali végrehajtást határozza meg.

Az `message` paraméter a bejelentkezett felhasználók összes terminál munkamenetére küldött figyelmeztető szöveg.

A SysV implementáció lehetővé teszi azon felhasználók limitálását, akik a `Ctrl` + `Alt` + `Del` billentyűkombinációval újraindíthatják a gépet. Ez úgy lehetséges, hogy az `/etc/inittab` fájlban

az `/etc/inittab` sorban a `ctrlaltdel`-re vonatkozó `-a` opciót a `shutdown` parancshoz helyezzük. Ezzel csak azok a felhasználók lesznek képesek újraindítani a rendszert a `Ctrl` + `Alt` + `Del` billentyűkombinációval, akiknek a felhasználóneve szerepel az `/etc/shutdown.allow` fájlban.

A `systemctl` parancs a számítógép kikapcsolására vagy újraindítására is használható a systemd-t használó rendszerekben. A rendszer újraindításához a `systemctl reboot`, a rendszer kikapcsolásához pedig a `systemctl poweroff` parancsot kell használni. Mindkét parancs futtatásához root jogosultságok szükségesek, mivel a hétköznapi felhasználók nem hajthatnak végre ilyen műveleteket.

NOTE

Egyes Linux disztribúciók a `poweroff` és `reboot` parancsokat különálló parancsokként linkelik a `systemctl`-hez. Például:

```
$ sudo which poweroff
/usr/sbin/poweroff
$ sudo ls -l /usr/sbin/poweroff
lrwxrwxrwx 1 root root 14 Aug 20 07:50 /usr/sbin/poweroff -> /bin/systemctl
```

Nem minden karbantartási tevékenységhez szükséges a rendszer kikapcsolása vagy újraindítása. Amikor azonban a rendszer állapotát egyfelhasználós üzemmódra kell váltani, fontos figyelmeztetni a bejelentkezett felhasználókat, hogy ne érje őket kár a tevékenységük hirtelen megszakítása miatt.

Hasonlóan ahhoz, amit a `shutdown` parancs tesz a rendszer kikapcsolásakor vagy újraindításakor, a `wall` parancs képes üzenetet küldeni az összes bejelentkezett felhasználó termináljába. Ehhez a rendszergazdának csak meg kell adnia egy fájlt, vagy közvetlenül az üzenetet kell paraméterként a `wall` parancsba írnia.

Gyakorló feladatok

1. Hogyan lehet a `telinit` parancsot a rendszer újraindítására használni?

2. Mi történik a `/etc/rc1.d/K90network` fájlhoz kapcsolódó szolgáltatásokkal, amikor a rendszer 1-es futási szintre lép?

3. A `systemctl` parancs segítségével hogyan tudja egy felhasználó ellenőrizni, hogy az `sshd.service` unit fut-e?

4. Egy systemd-alapú rendszerben milyen parancsot kell végrehajtani ahhoz, hogy a rendszer inicializálása során aktiválni lehessen az `sshd.service` unitot?

Gondolkodtató feladatok

1. Tegyük fel, hogy egy SysV-alapú rendszerben, hogy az `/etc/inittab'-ben meghatározott alapértelmezett futási szint 3, de a rendszer mindig 1-es futási szinten indul. Mi lehet ennek a legvalószínűbb oka?

2. Bár az `/sbin/init` fájl megtalálható a systemd alapú rendszerekben, ez csak egy szimbolikus link egy másik futtatható fájlhoz. Ilyen rendszerekben mi az a fájl, amelyre az `/sbin/init` mutat?

3. Hogyan ellenőrizhető az alapértelmezett rendszercél egy systemd alapú rendszerben?

4. Hogyan lehet a `shutdown` paranccsal tervezett újraindítást törölni?

Összefoglalás

Ez a lecke a Linux disztribúciók által szolgáltatáskezelőként használt főbb segédprogramokat tárgyalja. A SysVinit, systemd és Upstart segédprogramok mindegyike saját megközelítéssel irányítja a rendszerszolgáltatásokat és a rendszerállapotokat. A lecke a következő témákat járja körül:

- Mik azok a rendszerszolgáltatások és mi a szerepük az operációs rendszerben.
- A SysVinit, systemd és Upstart parancsok fogalma és alapvető használata.
- Hogyan kell megfelelően elindítani, leállítani és újraindítani a rendszerszolgáltatásokat és magát a rendszert.

A tárgyalt parancsok és eljárások a következők voltak:

- A SysVinit-hez kapcsolódó parancsok és fájlok, mint a `init`, `/etc/inittab` és a `telinit`.
- A fő systemd parancs: `systemctl`.
- Upstart parancsok: `initctl`, `status`, `start`, `stop`.
- Hagyományos energiagazdálkodási parancsok, mint a `shutdown` és a `wall`.

Válaszok a gyakorló feladatokra

1. Hogyan lehet a `telinit` parancsot a rendszer újraindítására használni?

   A `telinit 6` parancs 6-os futtatási szintre vált, vagyis újraindítja a rendszert.

2. Mi történik a `/etc/rc1.d/K90network` fájlhoz kapcsolódó szolgáltatásokkal, amikor a rendszer 1-es futási szintre lép?

   A fájlnév elején lévő `K` betű miatt a kapcsolódó szolgáltatások leállnak.

3. A `systemctl` parancs segítségével hogyan tudja egy felhasználó ellenőrizni, hogy az `sshd.service` unit fut-e?

   A `systemctl status sshd.service` vagy a `systemctl is-active sshd.service` paranccsal.

4. Egy systemd-alapú rendszerben milyen parancsot kell végrehajtani ahhoz, hogy a rendszer inicializálása során aktiválni lehessen az `sshd.service` unitot?

   A `systemctl enable sshd.service` parancs a root által futtatva.

Válaszok a gondolkodtató feladatokra

1. Tegyük fel, hogy egy SysV-alapú rendszerben, hogy az `/etc/inittab'-ben meghatározott alapértelmezett futási szint 3, de a rendszer mindig 1-es futási szinten indul. Mi lehet ennek a legvalószínűbb oka?

   `1` vagy `S` paraméterek lehetnek a kernel paraméterlistájában.

2. Bár az `/sbin/init` fájl megtalálható a systemd alapú rendszerekben, ez csak egy szimbolikus link egy másik futtatható fájlhoz. Ilyen rendszerekben mi az a fájl, amelyre az `/sbin/init` mutat?

   A systemd fő bináris állománya: `/lib/systemd/systemd`.

3. Hogyan ellenőrizhető az alapértelmezett rendszercél egy systemd alapú rendszerben?

   Az `/etc/systemd/system/default.target` szimbolikus link az alapértelmezett célként definiált egységfájlra mutat. Használható még a `systemctl get-default` parancs is.

4. Hogyan lehet a `shutdown` paranccsal tervezett újraindítást törölni?

   A `shutdown -c` parancsot kell használni.

Linux Professional Institute

Témakör 102: A Linux telepítése és a csomagkezelés

102.1 A merevlemez layout megtervezése

Hivatkozás az LPI célkitűzésre

LPIC-1 v5, Exam 101, Objective 102.1

Súlyozás

2

Kulcsfontosságú ismeretek

- A fájlrendszerek és a háttértár külön partíciókhoz vagy lemezekhez rendelése.
- A tervezést igazítása a rendszer tervezett használatához.
- Annak biztosítása, hogy a /boot partíció megfelel a hardverarchitektúra bootolási követelményeinek.
- Az LVM alapvető jellemzőinek ismerete.

A használt fájlok, kifejezések és segédprogramok listája

- `/` (root) filesystem
- `/var` filesystem
- `/home` filesystem
- `/boot` filesystem
- EFI System Partition (ESP)
- swap space
- mount points
- partitions

102.1 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	102 A Linux telepítése és a csomagkezelés
Fejezet:	102.1 A merevlemez layout megtervezése
Lecke:	1/1

Bevezetés

Ahhoz, hogy ezt a fejezetet sikerrel teljesíthessük, meg kell értenünk a *lemezek*, *partíciók*, *fájlrendszerek* és *kötetek* közötti kapcsolatot.

Gondoljunk úgy a lemezre (vagy *tárolóeszközre*, mivel a modern eszközök egyáltalán nem tartalmaznak “lemezeket”), mint az adataink "`fizikai tárolójára"`.

Mielőtt egy lemezt a számítógép használhatna, fel kell particionálni. A partíció a fizikai lemez egy logikai részhalmaza, mint egy logikai “kerítés”. A particionálással a lemezen tárolt információkat “felosztjuk”, elkülönítve például az operációs rendszer adatait a felhasználói adatoktól.

Minden lemeznek legalább egy partícióra van szüksége, de szükség esetén több partíciónk is lehet, és a róluk szóló információkat egy partíciós táblázatban tároljuk. Ez a táblázat tartalmazza a partíció első és utolsó szektorára és típusára vonatkozó információkat, valamint az egyes partíciók további részleteit.

Minden partíción belül van egy fájlrendszer. A fájlrendszer azt írja le, hogy az információ ténylegesen hogyan tárolódik a lemezen. Ez az információ magában foglalja, hogyan vannak szervezve a mappák, milyen kapcsolat van közöttük, hol vannak az egyes fájlok adatai, stb.

A partíciók nem terjedhetnek ki több lemezre. A *Logical Volume Manager* (LVM) segítségével azonban több partíciót is össze lehet vonni, akár lemezek között is, egyetlen logikai kötetet alkotva.

A logikai kötetek elvonatkoztatnak a fizikai eszközök korlátaitól és lehetővé teszik, hogy a hagyományos partíciókhoz képest sokkal rugalmasabban kombinálható vagy szétoszthatóó lemezterület “poolokkal” dolgozzon. Az LVM olyan helyzetekben hasznos, amikor egy partícióhoz több helyet kell hozzáadni anélkül, hogy az adatokat egy nagyobb eszközre kellene áttelepíteni.

Ebben a fejezetben megtanuljuk, hogyan tervezhetünk lemezparticionálási sémát egy Linux rendszerhez, fájlrendszereket és swap helyet rendelve külön partíciókhoz vagy lemezekhez, ha szükség van rá.

A partíciók és fájlrendszerek *létrehozását* és *kezelését* más leckékben tárgyaljuk. Ebben a fejezetben az LVM-et tekintjüt át, de a részletes magyarázat nem része a tananyagnak.

Csatolási pontok

Mielőtt egy fájlrendszert elérhetnénk Linuxon, azt *csatolni* (mountolni) kell. Ez azt jelenti, hogy a fájlrendszert a rendszer könyvtárfájának egy adott pontjához, az úgynevezett *csatolási ponthoz* (mount point) kell csatolni.

A csatolás után a fájlrendszer tartalma elérhető lesz a csatolási pont alatt. Képzeljük el például, hogy van egy partíció a felhasználók személyes adataival (a home mappáikkal), amely a `/john`, `/jack` és `/carol` mappákat tartalmazza. Ha a `/home` alá mountoljuk, akkor ezeknek a mappáknak a tartalma elérhető lesz a `/home/john`, `/home/jack` és `/home/carol` alatt.

A csatolási pontnak léteznie kell a fájlrendszer mountolása előtt. Nem lehet a `/mnt/userdata` alatti partíciót csatlakoztatni, ha ez a mappa nem létezik. Ha azonban a mappa létezik, és tartalmaz fájlokat, akkor ezek a fájlok nem lesznek elérhetők, amíg a fájlrendszer csatolását fel nem oldjuk (unmount). Ha kilistázzuk a mappa tartalmát, akkor a csatolt fájlrendszeren tárolt fájlokat fogjuk látni, nem pedig a mappa eredeti tartalmát.

A fájlrendszerek bárhová felcsatolhatók. Van azonban néhány jó gyakorlat, amelyet érdemes betartani a rendszer adminisztrációjának megkönnyítése érdekében.

Hagyományosan a `/mnt` volt az a mappa, amely alá minden külső eszközt csatlakoztattak, és számos előre konfigurált *horgonypont* (anchor point) létezett az általános eszközökhöz, mint például a CD-ROM meghajtók (`/mnt/cdrom`) és a floppy lemezek (`/mnt/floppy`).

Ezt felváltotta a `/media`, amely mára a felhasználó által eltávolítható adathordozók (pl. külső lemezek, USB flash meghajtók, memóriakártya-olvasók, optikai lemezek stb.) alapértelmezett

csatolási pontja.

A legtöbb modern Linux disztribúcióban és asztali környezetben a cserélhető eszközök automatikusan a `/media/USER/LABEL` alá lesznek csatlakoztatva, amikor a rendszerhez csatlakoznak, ahol a `USER` a felhasználónév, a `LABEL` pedig az eszköz címkéje. Például a `john` felhasználó által csatlakoztatott `FlashDrive` feliratú USB flash meghajtó a `/media/john/FlashDrive/` alá lesz csatlakoztatva. Ennek kezelése az asztali környezettől függ.

Ennek ellenére, amikor *manuálisan* kell csatolnunk egy fájlrendszert, jó gyakorlat, hogy a `/mnt` alá csatoljuk. A Linux alatt a fájlrendszerek csatolását és leválasztását vezérlő konkrét parancsokat egy másik leckében tárgyaljuk.

A dolgok szeparált elrendezése

Linuxon vannak olyan mappák, amelyeket érdemes külön partíciókon tartani. Ennek több oka is van: például a bootloaderrel kapcsolatos (a `/boot` alatt tárolt) fájlok *boot partíción* tartásával biztosíthatjuk, hogy a rendszer akkor is képes legyen elindulni, ha a root fájlrendszer összeomlik.

Ha a felhasználó személyes mappáját (a `/home` alatt) külön partíción tartjuk, könnyebbé válik a rendszer újratelepítése anélkül, hogy a felhasználói adatokhoz véletlenül hozzányúlnánk. A web- vagy adatbázis-kiszolgálóhoz kapcsolódó adatok (általában a `/var` alatt) külön partíción (vagy akár külön lemezen) tartása megkönnyíti a rendszeradminisztrációt, ha több lemezterületet kell hozzáadni az ilyen felhasználási esetekhez.

Teljesítménybeli okai is lehetnek annak, hogy bizonyos mappákat külön partíciókon tartsunk. Lehet, hogy a gyökér fájlrendszert (`/`) egy gyors SSD-n szeretnénk tartani, a nagyobb mappákat, mint például a `/home` és `/var` pedig lassabb merevlemezeken, amelyek sokkal több helyet kínálnak a költségek töredékéért.

A boot partíció (`/boot`)

A bootpartíció az operációs rendszer betöltéséhez a bootloader által használt fájlokat tartalmazza. Linux rendszereken a bootloader általában a GRUB2 vagy régebbi rendszereken a GRUB Legacy. A partíció általában a `/boot` alá van csatolva, a fájlokat pedig a `/boot/grub` alatt tárolja.

Gyakorlatilag nincs szükség boot partícióra, mivel a GRUB a legtöbb esetben képes csatlakoztatni a gyökérpartíciót (`/`) és betölteni a fájlokat egy külön `/boot` mappából.

A biztonság érdekében azonban szükséges lehet egy külön bootpartíció (amely biztosítja, hogy a rendszer akkor is elinduljon, ha a root fájlrendszer összeomlik), vagy ha olyan fájlrendszert szeretnénk használni, amelyet a bootloader nem ért a root partícióban, vagy ha nem támogatott

titkosítási vagy tömörítési módszert használ.

A rendszerindító partíció általában az első partíció a lemezen. Ennek az az oka, hogy az eredeti IBM PC BIOS a lemezeket *cilinderek*, *fejek* és *szektorok* (CHS) segítségével címezte, maximum 1024 cilinderrel, 256 fejjel és 63 szektorral, ami 528 MB maximális lemezméretet eredményezett (MS-DOS esetén 504 MB). Ez azt jelenti, hogy minden, ami ezen a határon túl van, nem lenne elérhető a régi rendszereken, hacsak nem használunk más lemezcímzési sémát (például *Logical Block Addressing*, LBA).

A maximális kompatibilitás érdekében a bootpartíció általában a lemez elején található és az 1024-es cilinder (528 MB) előtt ér véget, így biztosítva, hogy bármi történjen is, a gép mindig be tudja tölteni a kernelt.

Mivel a bootpartíció csak a bootloader, a kezdeti RAM lemez és a kernel imagek számára szükséges fájlokat tárolja, a mai szabványok szerint elég kicsi is lehet. A jó méret 300 MB körül van.

Az EFI rendszerpartíció (ESP)

Az *EFI rendszerpartíciót* (ESP) az *Unified Extensible Firmware Interface* (UEFI) rendszeren alapuló gépek használják a telepített operációs rendszerek rendszerbetöltőinek és kernel imageinek tárolására.

Ez a partíció FAT-alapú fájlrendszerűre van formázva. A GUID partíciós táblával particionált lemez globálisan egyedi azonosítója "C12A7328-F81F-11D2-BA4B-00A0C93EC93B". Ha a lemez MBR particionálási séma szerint lett formázva, a partíció azonosítója `0xEF`.

A Microsoft Windows-t futtató gépeken ez a partíció általában az első a lemezen, bár ez nem kötelező. Az ESP-t az operációs rendszer hozza létre (vagy tölti fel) a telepítéskor, és a Linux rendszereken a `/boot/efi` alá van csatolva.

A `/home` partíció

A rendszerben minden felhasználónak van egy saját mappája a személyes fájlok és beállítások tárolására, és ezek többsége a `/home` alatt található. Általában a home mappa megegyezik a felhasználónévvel, így a John felhasználó mappája a `/home/john` alatt található.

Vannak azonban kivételek. Például a root felhasználó otthoni mappája a `/root`, és néhány rendszerszolgáltatásnak máshol is lehet home mappája.

Nincs szabály a `/home` mappa partíciójának (a home partíció) méretének meghatározására. Figyelembe kell venni a rendszerben lévő felhasználók számát és a használat módját. Egy olyan

felhasználó, aki csak webböngészéssel és szövegszerkesztéssel foglalkozik, kevesebb helyet igényel, mint például egy olyan, aki videószerkesztéssel foglalkozik.

Változó adatok (`/var`)

Ez a mappa tartalmazza a “változó adatokat”, vagyis azokat a fájlokat és mappákat, amelyekbe a rendszernek képesnek kell lennie írni a működés során. Ide tartoznak a rendszernaplók (a `/var/log` mappában), az ideiglenes fájlok (`/var/tmp`) és a gyorsítótárazott alkalmazási adatok (a `/var/cache` mappában).

A `/var/www/html` az Apache webkiszolgáló adatfájljainak alapértelmezett mappája, a `/var/lib/mysql` pedig a MySQL szerver adatbázisfájljainak alapértelmezett helye. Azonban mindkettő megváltoztatható.

A stabilitás az egyik jó ok arra, hogy a `/var`-t külön partícióra helyezzük. Sok alkalmazás és folyamat ír a `/var`-ba és alkönyvtáraiba, mint például a `/var/log`-ba vagy a `/var/tmp`-be. Egy rossz működésű folyamat addig írhat adatokat, amíg nem marad szabad hely a fájlrendszerben.

Ha a `/var` a `/` alatt van, ez kernelpánikot és a fájlrendszer sérülését okozhatja, ezzel olyan helyzetet idézve elő, amelyből nehéz helyreállni. Ha azonban a `/var` egy külön partíció alatt van, akkor a gyökér fájlrendszer nem lesz érintett.

A `/home`-hoz hasonlóan a `/var` partíció méretének meghatározására sincs általános szabály, mivel az a rendszer használatától függően változik. Egy otthoni rendszeren lehet, hogy csak néhány gigabájtot vesz igénybe. Egy adatbázis- vagy webszerver esetében azonban sokkal több helyre lehet szükség. Ilyen esetekben bölcs dolog lehet a `/var` partíciót a gyökérpartíciótól eltérő lemezen elhelyezni, ami további védelmet nyújt a fizikai lemez meghibásodása ellen.

Swap

A swap partíciót arra használják, hogy szükség szerint memóriaoldalakat váltson a RAM-ból a lemezre. Ennek a partíciónak meghatározott típusúnak kell lennie, és a megfelelő `mkswap` nevű segédprogrammal kell beállítani, mielőtt használni lehetne.

A swap partíciót nem lehet a többihez hasonlóan csatlakoztatni, ami azt jelenti, hogy nem lehet normál mappaként hozzáférni és megnézni a tartalmát.

Egy rendszernek több swap partíciója is lehet (bár ez nem gyakori) és a Linux támogatja a swap *fájlok* használatát is a partíciók helyett, ami hasznos lehet a swap hely gyors növeléséhez, ha szükséges.

A swap partíció mérete vitatott kérdés. A Linux korai időszakából származó régi szabály (“a RAM

kétszerese") már nem feltétlenül érvényes - attól függ, hogy a rendszert hogyan használják, és mennyi fizikai RAM van benne.

A Red Hat Enterprise Linux 7 dokumentációjában a Red Hat a következőket ajánlja:

RAM mennyisége	**Javasolt Swap-méret**	**Javasolt Swap-méret hibernálással**
< 2 GB RAM	A RAM duplája	A RAM háromszorosa
2-8 GB RAM	A RAM mennyiségével megegyező	A RAM duplája
8-64 GB RAM	Legalább 4 GB	A RAM másfélszerese
> 64 GB RAM	Legalább 4 GB	Nem javasolt

Természetesen a swap mennyisége függhet a munkaterheléstől. Ha a gépen kritikus szolgáltatást, például adatbázis-, web- vagy SAP-kiszolgálót futtatunk, célszerű az ilyen szolgáltatások dokumentációjában (vagy a szoftver gyártójánál) ellenőrizni az ajánlást.

NOTE A swap-partíciók és swap-fájlok létrehozásával és engedélyezésével kapcsolatban lsd. az LPIC-1 104.1. fejezetét.

LVM

Már volt róla szó, hogy a lemezek egy vagy több partícióba vannak szervezve és minden partíció tartalmaz egy fájlrendszert, amely leírja a fájlok és a hozzájuk tartozó metaadatok tárolásának módját. A particionálás egyik hátránya, hogy a rendszergazdának előre el kell döntenie, hogy az eszközön rendelkezésre álló lemezterületet hogyan osztja el. Ez később kihívást jelenthet, ha egy partíció az eredetileg tervezettnél több helyet igényel. A partíciók természetesen átméretezhetők, de ez nem biztos, hogy lehetséges, ha például nincs szabad hely a lemezen.

A *Logical Volume Management* (LVM) a tároló-virtualizáció egy formája, amely a rendszergazdáknak a hagyományos particionálásnál rugalmasabb megközelítést kínál a lemezterület kezeléséhez. Az LVM célja, hogy megkönnyítse a végfelhasználók tárolási igényeinek kezelését. Az alapegység a *Physical Volume* (PV - fizikai kötet), amely egy blokkeszköz a rendszeren, mint például egy lemezpartíció vagy egy RAID-tömb.

A PV-ket *Volume Groups*-okba (VG - kötetcsoport) csoportosítják, amelyek absztrahálják a mögöttes eszközöket, és egyetlen logikai eszköznek tekinthetők, az egyes PV-k együttes tárolókapacitásával.

A kötetcsoport minden egyes kötete fix méretű darabokra van felosztva, amelyeket *extenteknek* nevezünk. A PV-n található kiterjedéseket *Physical Extents* (PE), míg a logikai köteteken

találhatóakat *Logical Extents* (LE) nevezzük. Általában minden logikai kiterjedés egy fizikai kiterjedéshez van hozzárendelve, de ez változhat, ha olyan funkciókat használunk, mint például a lemeztükrözés.

A kötetcsoportok logikai kötetekre (LV) oszthatók, amelyek funkcionálisan a partíciókhoz hasonlítanak, de rugalmasabban működnek.

A logikai kötet méretét létrehozásakor megadott méretét valójában a fizikai kiterjesztések mérete (alapértelmezés szerint 4 MB) és a kötetben lévő kiterjesztések számának szorzata határozza meg. Ebből könnyen érthető, hogy például egy logikai kötet növeléséhez a rendszergazdának csak annyi a dolga, hogy a kötetcsoportban rendelkezésre álló állományból több extentet adjon hozzá. Hasonlóképpen, az LV zsugorításához extenteket lehet eltávolítani.

A logikai kötet létrehozása után az operációs rendszer úgy tekint rá, mint egy normál blokkeszközre. Egy eszköz jön létre a `/dev` könyvtárban, amelynek neve `/dev/VGNAME/LVNAME`, ahol a `VGNAME` a kötetcsoport neve, az `LVNAME` pedig a logikai kötet neve.

Ezeket az eszközöket szabványos segédprogramokkal (például `mkfs.ext4`) lehet formázni a kívánt fájlrendszerrel, és a szokásos módszerekkel lehet csatlakoztatni, akár manuálisan a `mount` paranccsal, akár automatikusan az `/etc/fstab` fájlhoz való hozzáadással.

Gyakorló feladatok

1. Linux rendszereken hol vannak a GRUB bootloader fájljai?

2. Hol végződjön a boot partíció, hogy a számítógép mindig be tudja tölteni a kernelt?

3. Általában hová van mountolva az EFI partíció?

4. Ha manuálisan csatlakoztatunk egy fájlrendszert, általában melyik mappa alá kell csatlakoztatni?

Gondolkodtató feladatok

1. Mi a legkisebb egység egy kötetcsoporton (VG) belül?

2. Hogyan határozható meg a logikai kötet (LV) mérete?

3. Egy MBR partícionálási sémával formázott lemezen mi az EFI rendszerpartíció azonosítója?

4. A swap partíciókon kívül hogyan lehet gyorsan növelni a swap helyet egy Linux rendszerben?

Összefoglalás

Ebben a leckében a particionálásról tanultunk és arról, hogy mely mappákat tartják általában külön partíciókban, és hogy mi ennek az oka. Emellett áttekintést kaptunk az LVM-ről (Logical Volume Management), és arról, hogy a hagyományos partícionáláshoz képest hogyan kínál rugalmasabb módot az adatok és a lemezterület kiosztására.

A következő fájlokról, kifejezésekről és segédprogramokról volt szó:

`/`

A Linux gyökér fájlrendszere.

`/var`

A “változó adatok” standard helye, olyan adatoké, amelyek az idő múlásával nőhetnek és csökkenhetnek.

`/home`

Az állandó felhasználók home könyvtárainak a standard szülőmappája.

`/boot`

A rendszerbetöltő fájlok, a Linux kernel és a kezdeti RAM lemez szabványos helye.

EFI System Partition (ESP)

Az UEFI-t megvalósító rendszerek használják a rendszer indítófájljainak tárolására.

Swap hely

A kernelmemória-oldalak cseréjére szolgál, ha a RAM nagymértékben igénybe van véve.

Csatolási pontok

Olyan mappák, ahová egy eszköz (mint például egy merevlemez) csatolható.

Partíciók

A merevlemez felosztásai.

Válaszok a gyakorló feladatokra

1. Linux rendszereken hol vannak a GRUB bootloader fájljai?

   A `/boot/grub` alatt.

2. Hol végződjön a boot partíció, hogy a számítógép mindig be tudja tölteni a kernelt?

   Az 1024-es cilinder előtt.

3. Általában hová van mountolva az EFI partíció?

   A `/boot/efi` alá.

4. Ha manuálisan csatlakoztatunk egy fájlrendszert, általában melyik mappa alá kell csatlakoztatni?

   Az `/mnt` alá, azonban ez nem kötelező. Bármely mappa alá csatolhatunk partíciót.

Válaszok a gondolkodtató feladatokra

1. Mi a legkisebb egység egy kötetcsoporton (VG) belül?

   A kötetcsoportok extentekre vannak felosztva.

2. Hogyan határozható meg a logikai kötet (LV) mérete?

   A fizikai extentek méretének és a kötetben lévő extentek számának a szorzatával.

3. Egy MBR partícionálási sémával formázott lemezen mi az EFI rendszerpartíció azonosítója?

   `0xEF` az ID.

4. A swap partíciókon kívül hogyan lehet gyorsan növelni a swap helyet egy Linux rendszerben?

   Swap fájlok segítségével.

102.2 Boot manager telepítése

Hivatkozás az LPI célkitűzésre

LPIC-1 v5, Exam 101, Objective 102.2

Súlyozás

2

Kulcsfontosságú ismeretek

- Alternatív indítási helyek és biztonsági mentési lehetőségek biztosítása.
- Egy olyan bootloader telepítése és konfigurálása, mint például a GRUB Legacy.
- Alapvető konfigurációs módosítások elvégzése a GRUB 2 számára.
- Interakció a bootloaderrel.

A használt fájlok, kifejezések és segédprogramok listája

- `menu.lst`, `grub.cfg` and `grub.conf`
- `grub-install`
- `grub-mkconfig`
- MBR

102.2 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	102 A Linux telepítése és a csomagkezelés
Fejezet:	102.2 Boot manager telepítése
Lecke:	1/1

Bevezetés

A számítógép bekapcsolásakor az első szoftver, amely elindul, a bootloader. Ez egy olyan kódrészlet, amelynek egyetlen célja, hogy betöltse az operációs rendszer kernelét, és átadja neki az irányítást. A kernel betölti a szükséges illesztőprogramokat, inicializálja a hardvert, majd betölti az operációs rendszer többi részét.

A legtöbb Linux-disztribúcióban használt bootloader a GRUB. A Linux kernelt vagy más operációs rendszereket, például a Windowst is képes betölteni, és több kernelképet és paramétert is képes külön menüpontokként kezelni. A kernel kiválasztása a rendszerindításkor billentyűzetvezérelt felületen keresztül történik, és van egy parancssori felület a rendszerindítási opciók és paraméterek szerkesztésére.

A legtöbb Linux disztribúció automatikusan telepíti és konfigurálja a GRUB-ot (valójában a GRUB 2-t), így egy átlagos felhasználónak nem kell erre gondolnia. Rendszergazdaként azonban létfontosságú tudni, hogyan lehet irányítani a rendszerindítási folyamatot, hogy például egy sikertelen kernelfrissítés után helyreállíthassuk a rendszert egy hibás rendszerindításból.

Ebben a leckében a GRUB telepítésével, konfigurálásával és a GRUB-bal való interakcióval fogunk

foglalkozni.

GRUB Legacy vs. GRUB 2

A GRUB (*Grand Unified Bootloader*) eredeti verzióját, amely ma *GRUB Legacy* néven ismert, 1995-ben fejlesztették ki a GNU Hurd projekt részeként, és később számos Linux-disztribúció alapértelmezett rendszerbetöltőjeként használták, felváltva a korábbi alternatívákat, például a LILO-t.

A GRUB 2 a GRUB teljes átírása, amelynek célja, hogy tisztább, biztonságosabb, robusztusabb és erősebb legyen. A GRUB Legacyvel szembeni számos előnye között szerepel a sokkal rugalmasabb konfigurációs fájl (sokkal több paranccsal és feltételes utasítással, hasonlóan egy szkriptnyelvhez), a modulárisabb felépítés és a jobb lokalizáció/internacionalizálás.

Támogatja továbbá a témákat és a grafikus indítási menüket, a LiveCD ISO-k betöltését közvetlenül a merevlemezről, jobban támogatja a nem x86-os architektúrákat, univerzálisan támogatja az UUID-ket (megkönnyítve a lemezek és partíciók azonosítását) és még sok minden mást.

A GRUB Legacy már nem áll aktív fejlesztés alatt (az utolsó kiadás a 0.97-es volt 2005-ben), és ma már a legtöbb nagyobb Linux disztribúció a GRUB 2-t telepíti alapértelmezett bootloaderként. Azonban még mindig találhatunk GRUB Legacy-t használó rendszereket, ezért fontos tudni, hogyan kell használni, és miben különbözik a GRUB 2-től.

Hol van a Bootloader?

Az IBM PC-kompatibilis rendszerek merevlemezeit hagyományosan az 1982-ben az IBM PC-DOS (MS-DOS) 2.0-hoz készített MBR particionálási sémával particionálták.

Ebben a sémában a lemez első 512 bájtos szektorát *Master Boot Record*-nak nevezik, és ez tartalmazza a lemezen lévő partíciókat leíró táblázatot (a partíciós táblát), valamint a bootloadernek nevezett bootstrap kódot.

A számítógép bekapcsolásakor ez a (méretkorlátozások miatt) nagyon minimális bootloader kód betöltődik, végrehajtódik és átadja a vezérlést a lemezen lévő másodlagos bootloadernek, amely általában az MBR és az első partíció közötti 32 KB-os területen található, és ami betölti az operációs rendszer(ek)et.

MBR-partícionált lemezen a GRUB boot kódja az MBR-re települ. Ez betölti és átadja a vezérlést az MBR és az első partíció közé telepített “core” lemezképnek. Ettől a ponttól kezdve a GRUB képes a többi szükséges erőforrást (menüdefiníciók, konfigurációs fájlok és extra modulok) a lemezről

betölteni.

Az MBR azonban korlátozza a partíciók számát (eredetileg maximum 4 elsődleges partíció, később maximum 3 elsődleges partíció, 1 kiterjesztett partícióval, amely logikai partíciókra van felosztva) és a lemezek maximális mérete 2 TB. E korlátozások leküzdésére egy új particionálási séma, a GPT (*GUID Partition Table*) jött létre, amely az UEFI (*Unified Extensible Firmware Interface*) szabvány része.

A GPT-particionált lemezek hagyományos PC BIOS-szal vagy UEFI firmware-rel rendelkező számítógépeken egyaránt használhatók. BIOS-szal rendelkező gépeken a GRUB második része egy speciális BIOS bootpartícióban tárolódik.

Az UEFI firmware-rel rendelkező rendszereken a GRUB-ot a firmware a `grubia32.efi` (32 bites rendszerek esetén) vagy a `grubx64.efi` (64 bites rendszerek esetén) fájlokból tölti be az ESP (*EFI System Partition*) nevű partícióról.

A `/boot` partíció

Linuxon a rendszerindításhoz szükséges fájlokat általában egy boot partíción tárolják, amelyet a root fájlrendszer alá csatolnak, és a köznyelvben `/boot` néven emlegetnek.

A boot partícióra nincs szükség a jelenlegi rendszereken, mivel az olyan bootloaderek, mint a GRUB, általában tudják csatlakoztatni a gyökér fájlrendszert és keresik a szükséges fájlokat a `/boot` könyvtárban, de ez azért egy jó gyakorlat, mert elkülöníti a boot folyamathoz szükséges fájlokat a fájlrendszer többi részétől.

Ez a partíció általában az első a lemezen. Ennek az az oka, hogy az eredeti IBM PC BIOS a lemezeket *Cilinderek* (Cylinders), *Fejek* (Heads) és *Szektorok* (Sectors) (CHS) használatával címezte, maximum 1024 cilinderrel, 256 fejjel és 63 szektorral, ami 528 MB maximális lemezméretet eredményezett (504 MB MS-DOS alatt). Ez azt jelenti, hogy minden, ami ezen a határon túl van, nem lenne elérhető, hacsak nem használunk más lemezcímzési sémát (például LBA, *Logical Block Addressing*).

A maximális kompatibilitás érdekében a `/boot` partíció általában a lemez elején található és az 1024-es cilinder (528 MB) előtt ér véget, így biztosítva, hogy a gép mindig be tudja tölteni a kernelt. Ennek a partíciónak az ajánlott mérete egy mai gépen 300 MB.

További okok a külön `/boot` partíció létrehozására a titkosítás és a tömörítés, mivel néhány módszert a GRUB 2 még nem támogat, vagy ha a rendszer gyökérpartícióját (`/`) nem támogatott fájlrendszerrel kell formázni.

A Boot partíció tartalma

A `/boot` partíció tartalma a rendszer architektúrájától vagy a használt bootloadertől függően változhat, de egy x86-alapú rendszeren általában az alábbi fájlokat találhatjuk meg. Ezek többsége `-VERSION` utótaggal van elnevezve, ahol a `-VERSION` a megfelelő Linux kernel verziója. Így például a Linux kernel `4.15.0-65-generic` verziójához tartozó konfigurációs fájl neve `config-4.15.0-65-generic`.

Config fájl

Ez az általában `config-VERSION` nevű fájl (lsd. a fenti példát) a Linux kernel konfigurációs paramétereit tárolja. Ez a fájl automatikusan létrejön, amikor egy új kernelt fordítanak vagy telepítenek, és a felhasználónak *nem szabad* közvetlenül módosítania.

System map

Ez a fájl egy keresőtábla, amely a szimbólumok (például változók vagy függvények) neveit a memóriában lévő megfelelő pozíciójukhoz rendeli. Ez hasznos a *kernel panic* (kernelpánik) néven ismert rendszerhiba elhárításakor, mivel lehetővé teszi a felhasználó számára, hogy megtudja, melyik változót vagy függvényt hívták meg a hiba bekövetkezésekor. A config fájlhoz hasonlóan a neve általában `System.map-VERSION` (pl. `System.map-4.15.0-65-generic`).

Linux kernel

Ez az operációs rendszer tulajdonképpeni kernele. A neve általában `vmlinux-VERSION` (pl. `vmlinux-4.15.0-65-generic`). A `vmlinux` helyett a `vmlinuz` név is előfordulhat, a `z` a végén azt jelenti, hogy a fájl tömörítve van.

Initial RAM lemez

Ez általában az `initrd.img-VERSION` nevet viseli, és egy RAM lemezre betöltött minimális root fájlrendszert tartalmaz, amely tartalmazza a segédprogramokat és a kernelmodulokat, amelyek szükségesek ahhoz, hogy a kernel fel tudja csatolni a valódi root fájlrendszert.

Bootloaderrel kapcsolatos fájlok

Ezek a GRUB konfigurációs fájl (GRUB 2 esetén a `/boot/grub/grub.cfg`, illetve GRUB Legacy esetén a `/boot/grub/menu.lst`), modulok (a `/boot/grub/i386-pc`-ben), fordítási fájlok (a `/boot/grub/locale`-ben) és betűtípusok (a `/boot/grub/fonts`-ben).

GRUB 2

A GRUB 2 telepítése

A GRUB 2 a `grub-install` segédprogrammal telepíthető. Ha nem-bootoló rendszerrel

rendelkezünk, akkor egy Live CD vagy rescue lemez segítségével kell bootolni. Meg kell találnunk, hogy melyik a rendszer boot partíciója, csatlakoztatni kell azt, majd futtatni a segédprogramot.

NOTE

Az alábbi parancsok azt feltételezik, hogy root felhasználóként vagyunk bejelentkezve. Ha nem, akkor először futtassuk a `sudo su -` parancsot, hogy root "legyünk". Ha végeztünk, írjuk be az `exit` parancsot, hogy kijelentkezzünk, és visszatérjünk normál felhasználóvá.

A rendszer első lemeze általában a *boot eszköz*, és tudnunk kell, hogy van-e rajta *boot partíció*. Ezt az `fdisk` segédprogrammal lehet ellenőrizni. A gép első lemezén lévő összes partíciót a következővel tudjuk felsorolni:

```
# fdisk -l /dev/sda
Disk /dev/sda: 111,8 GiB, 120034123776 bytes, 234441648 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x97f8fef5

Device     Boot    Start       End   Sectors   Size Id Type
/dev/sda1  *        2048   2000895   1998848   976M 83 Linux
/dev/sda2        2002942 234440703 232437762 110,9G  5 Extended
/dev/sda5        2002944  18008063  16005120   7,6G 82 Linux swap / Solaris
/dev/sda6       18010112 234440703 216430592 103,2G 83 Linux
```

A boot partíciót a boot oszlop alatti `*` jelöli. A fenti példában ez a `/dev/sda1`.

Most hozzunk létre egy ideiglenes mappát az `/mnt` alatt, és csatlakoztassuk alá a partíciót:

```
# mkdir /mnt/tmp
# mount /dev/sda1 /mnt/tmp
```

Ezután futtassuk a `grub-installt`, a *boot eszközre* (nem a partícióra) és a mappára mutatva, ahová a bootpartíciót csatlakoztattuk. Ha a rendszernek van dedikált boot partíciója, a parancs a következő:

```
# grub-install --boot-directory=/mnt/tmp /dev/sda
```

Ha olyan rendszerre telepítünk, amely nem rendelkezik boot partícióval, hanem csak egy `/boot`

mappával a gyökér fájlrendszeren, akkor a `grub-install`-t kell használnunk. A parancs tehát a következő:

```
# grub-install --boot-directory=/boot /dev/sda
```

A GRUB 2 konfigurálása

A GRUB 2 alapértelmezett konfigurációs fájlja a `/boot/grub/grub.cfg`. Ez a fájl automatikusan generálódik, a kézi szerkesztése nem ajánlott. A GRUB konfiguráció megváltoztatásához szerkeszteni kell az `/etc/default/grub` fájlt, majd az `update-grub` segédprogramot kell futtatni a megfelelő fájl létrehozásához.

NOTE Az `update-grub` általában a `grub-mkconfig -o /boot/grub/grub.cfg` rövidítése, így ugyanaz az eredmény.

Az `/etc/default/grub` fájlban van néhány olyan opció, amely a GRUB 2 viselkedését szabályozza, mint például az alapértelmezetten indítandó kernel, időkorlát, extra parancssori paraméterek stb. A legfontosabbak a következők:

Az alapértelmezett menüpont a rendszerindításhoz. Ez lehet egy numerikus érték (mint `0`, `1`, stb.), egy menüpont neve (mint `debian`) vagy `saved`, amit a `GRUB_SAVEDEFAULT=`-tal együtt használunk, amiről lentebb lesz szó. Ne feledjük, hogy a menübejegyzések nullával kezdődnek, tehát az első menübejegyzés `0`, a második `1`, stb.

`GRUB_SAVEDEFAULT=`

Ha ez az opció `true`-ra van állítva, és a `GRUB_DEFAULT= saved`-re van állítva, akkor az alapértelmezett boot opció mindig az lesz, amit utoljára választottak ki a boot menüben.

`GRUB_TIMEOUT=`

Az alapértelmezett menüpont kiválasztása előtti időkorlát másodpercben. Ha `0`-ra van állítva, a rendszer az alapértelmezett bejegyzést indítja el anélkül, hogy megjelenítené a menüt. Ha `-1`-re van állítva, a rendszer várni fog, amíg a felhasználó kiválaszt egy opciót, függetlenül attól, hogy mennyi ideig tart.

`GRUB_CMDLINE_LINUX=`

Ez felsorolja azokat a parancssori opciókat, amelyek a Linux kernel bejegyzéseihez adódnak hozzá.

`GRUB_CMDLINE_LINUX_DEFAULT=`

Alapértelmezés szerint minden Linux kernelhez két menübejegyzés készül, egy az

alapértelmezett beállításokkal és egy a helyreállításhoz. Ezzel az opcióval extra paramétereket adhatunk hozzá, amelyek csak az alapértelmezett bejegyzéshez kerülnek hozzá.

`GRUB_ENABLE_CRYPTODISK=`

Ha `y`-ra van állítva, akkor az olyan parancsok, mint a `grub-mkconfig`, `update-grub` és `grub-install` keresni fogják a titkosított lemezeket, és hozzáadják a bootolás során a hozzáférésükhöz szükséges parancsokat. Ez kikapcsolja az automatikus bootolást (`GRUB_TIMEOUT=` bármilyen, `-1`-től eltérő értékkel), mivel a lemezek hozzáférése előtt egy jelszóra van szükség a lemezek visszafejtéséhez.

A menübejegyzések menedzselése

Az `update-grub` futtatásakor a GRUB 2 megvizsgálja a gépen lévő kerneleket és operációs rendszereket és létrehozza a megfelelő menübejegyzéseket a `/boot/grub/grub.cfg` fájlban. Az új bejegyzéseket manuálisan is hozzá lehet adni az `/etc/grub.d` könyvtárban található szkriptfájlokhoz.

Ezeknek a fájloknak futtathatónak kell lenniük, és az `update-grub` numerikus sorrendben dolgozza fel őket. Ezért a `05_debian_theme` a `10_linux` előtt kerül feldolgozásra, és így tovább. Az egyéni menübejegyzések általában a `40_custom` fájlba kerülnek.

A menübejegyzés alap szintaxisa az alábbiakban látható:

```
menuentry "Default OS" {
    set root=(hd0,1)
    linux /vmlinuz root=/dev/sda1 ro quiet splash
    initrd /initrd.img
}
```

Az első sor mindig `menuentry`-vel kezdődik és `{`-val végződik. Az idézőjelek közötti szöveg jelenik meg a GRUB 2 rendszerindító menüjének címkéjeként.

A `set root` paraméter meghatározza azt a lemezt és partíciót, ahol az operációs rendszer gyökérfájlrendszere található. Vegyük figyelembe, hogy a GRUB 2 esetében a lemezek számozása nullától kezdődik, így a `hd0` az első lemez (`sda` Linux alatt), a `hd1` a második, és így tovább. A partíciók számozását viszont egytől indítjuk. A fenti példában a gyökérfájlrendszer az első lemezen (`hd0`), az első partíción (`,1`), vagyis az `sda1`-en található.

Az eszköz és a partíció közvetlen megadása helyett a GRUB 2 kereshet egy adott címkével vagy UUID-vel (*Universally Unique Identifier*) ellátott fájlrendszert is. Ehhez használjuk a `search --set=root` paramétert, amelyet a `--label` paraméter és a keresendő fájlrendszer címkéje

követ, vagy a `--fs-uuid` paramétert, amelyet a fájlrendszer UUID-je követ.

A fájlrendszer UUID-jét az alábbi paranccsal találhatjuk meg:

```
$ ls -l /dev/disk/by-uuid/
total 0
lrwxrwxrwx 1 root root 10 nov  4 08:40 3e0b34e2-949c-43f2-90b0-25454ac1595d -> ../../sda5
lrwxrwxrwx 1 root root 10 nov  4 08:40 428e35ee-5ad5-4dcb-adca-539aba6c2d84 -> ../../sda6
lrwxrwxrwx 1 root root 10 nov  5 19:10 56C11DCC5D2E1334 -> ../../sdb1
lrwxrwxrwx 1 root root 10 nov  4 08:40 ae71b214-0aec-48e8-80b2-090b6986b625 -> ../../sda1
```

A fenti példában a `/dev/sda1` UUID-je `ae71b214-0aec-48e8-80b2-090b6986b625`. Ha a GRUB 2 gyökéreszközeként szeretnénk beállítani, a parancs a következő: `search --set=root --fs-uuid ae71b214-0aec-48e8-80b2-090b6986b625`.

A `search` parancs használatakor szokás hozzáadni a `--no-floppy` paramétert, hogy a GRUB ne pazarolja az időt a floppy lemezeken való keresésre.

A `linux` sor azt jelzi, hogy hol található az operációs rendszer kernele (ebben az esetben a `vmlinuz` fájl a fájlrendszer gyökerében). Ezt követően parancssori paramétereket adhatunk át a kernelnek.

A fenti példában megadtuk a root partíciót (`root=/dev/sda1`) és átadtunk három kernel paramétert: a root partíciót csak olvashatóan kell csatlakoztatni (`ro`), a legtöbb naplóüzenetet le kell tiltani (`quiet`) és egy splash képernyőt kell megjeleníteni (`splash`).

Az `initrd` sor jelzi, hogy hol található az initial RAM lemez. A fenti példában a fájl az `initrd.img`, amely a fájlrendszer gyökerében található.

NOTE A legtöbb Linux-disztribúció nem a gyökérfájlrendszer gyökérmappájába helyezi a kernelt és az initrd-t. Ehelyett ezek a `/boot` mappában vagy partícióban lévő tényleges fájlokra mutató hivatkozások.

A menübejegyzés utolsó sora csak a `}` karaktert tartalmazhatja.

A GRUB 2 használata

A GRUB 2-vel indított rendszer indításakor egy opciókból álló menü jelenik meg. A kurzorbillentyűkkel válasszunk ki egy opciót, majd `Enter` a kiválasztott bejegyzés megerősítéséhez és a rendszerindításhoz.

TIP Ha csak egy visszaszámlálót látunk, de nem egy menüt, nyomjuk meg a `Shift` billentyűt

a menü megjelenítéséhez.

Egy opció szerkesztéséhez jelöljük ki azt a kurzorbillentyűkkel, majd nyomjuk meg a `E` billentyűt. Ekkor megjelenik egy szerkesztőablak az adott opcióhoz tartozó `/boot/grub/grub.cfg'-ben meghatározott `menuentry` tartalmával.

Egy opció szerkesztése után nyomjuk meg a a `Ctrl` + `X` vagy a `F10` billentyűt a rendszerindításhoz, vagy a `Esc` billentyűt a menübe való visszatéréshez.

A GRUB 2 shellbe való belépéshez nyomjuk meg a `C` billentyűt a menüképernyőn (vagy a szerkesztőablakban a `Ctrl` + `C`). Egy ilyen command prompt fog megjelenni: `grub >`

Írjuk be a `help` szót az összes elérhető parancs listájának megtekintéséhez, vagy nyomjuk meg a `Esc` billentyűt a shellből való kilépéshez és a menüképernyőre való visszatéréshez.

NOTE Ne feledjük, hogy ez a menü nem jelenik meg, ha az `/etc/default/grub`-ban a `GRUB_TIMEOUT` értéke `0`.

Bootolás a GRUB 2 Shellből

A GRUB 2 shell segítségével indíthatjuk el a rendszert, ha egy menüpont hibás beállítása miatt nem indulna el.

Az első dolog, amit meg kell tennünk, hogy kiderítjük, hol van a boot partíció. Ezt az `ls` paranccsal tehetjük meg, amely megmutatja a GRUB 2 által talált partíciók és lemezek listáját.

```
grub> ls
(proc) (hd0) (hd0,msdos1)
```

A fenti példában egyszerűek a dolgok. Csak egy lemez van, a `(hd0)`, amelyen csak egy partíció van: `(hd0,msdos1)`.

A felsorolt lemezek és partíciók természetesen eltérőek lesznek a rendszertől függően. Példánkban a `hd0` első partíciójának neve `msdos1`, mivel a lemez MBR partícionálási sémával lett particionálva. Ha a lemez GPT partícionálással lett volna particionálva, a neve `gpt1` lenne.

A Linux indításához szükségünk van egy kernelre és egy initial RAM lemezre (initrd). Ellenőrizzük a `(hd0,msdos1)` tartalmát:

```
grub> ls (hd0,msdos1)/
lost+found/ swapfile etc/ media/ bin/ boot/ dev/ home/ lib/ lib64/ mnt/ opt/ proc/ root/
```

```
run/ sbin/ srv/ sys/ tmp/ usr/ var/ initrd.img initrd.img.old vmlinuz cdrom/
```

Az `-l` paramétert hozzáadhatjuk az `ls`-hez, hogy hosszú listát kapjunk, hasonlóan ahhoz, mint a Linux terminál esetén. A `Tab` használatával automatikusan kitölthetjük a lemez-, partíció- és fájlneveket.

Vegyük észre, hogy van egy kernel (`vmlinuz`) és egy initrd (`initrd.img`) image fájlunk a gyökérkönyvtárban. Ha nem, akkor a `/boot` tartalmát a `list (hd0,msdos1)/boot/` segítségével ellenőrizhetjük.

Most nézzük a boot partíciót:

```
grub> set root=(hd0,msdos1)
```

Töltse be a Linux kernelt a `linux` paranccsal, majd adjuk meg a kernel elérési útvonalát és a `root=` opciót, hogy megmondjuk a kernelnek, hol található az operációs rendszer gyökérfájlrendszere.

```
grub> linux /vmlinuz root=/dev/sda1
```

Töltsük be az initial RAM-lemezt az `initrd` paranccsal, amelyet az `initrd.img` fájl teljes elérési útvonala követ:

```
grub> initrd /initrd.img
```

Most indítsuk el a rendszert a `boot`-tal.

Bootolás a Rescue Shellből

Indítási hiba (boot failure) esetén a GRUB 2 betölthet egy rescue shellt, a korábban említett shell egyszerűsített változatát. Ezt a command promptból ismerhetjük fel, mely a `grub rescue>` formában jelenik meg.

A rendszer ebből a shellből történő indítása szinte teljesen megegyezik az előzőekben bemutatottakkal. Azonban be kell töltenünk néhány GRUB 2 modult, hogy működjön.

Miután kiderítettük, hogy melyik partíció a boot partíció (az `ls` paranccsal, ahogyan azt korábban bemutattuk), használjuk a `set prefix=` parancsot, majd a GRUB 2 fájlokat tartalmazó mappa teljes elérési útvonalát. Általában `/boot/grub`. A mi példánkban:

```
grub rescue> set prefix=(hd0,msdos1)/boot/grub
```

Most töltsük be a `normal` és `linux` modulokat az `insmod` paranccsal:

```
grub rescue> insmod normal
grub rescue> insmod linux
```

Ezután állítsuk be a boot partíciót a `set root=` paranccsal, ahogyan azt korábban már megtettük, töltsük be a linux kernelt (`linux`), az initial RAM lemezt (`initrd`) és próbáljunk meg bootolni a `boot` programmal.

GRUB Legacy

A GRUB Legacy telepítése egy működő rendszerből

A GRUB Legacy lemezre történő telepítéséhez egy futó rendszerből a `grub-install` segédprogramot fogjuk használni. Az alapparancs a `grub-install DEVICE`, ahol a `DEVICE` az a lemez, ahová a GRUB Legacy-t telepíteni szeretnénk. Egy példa erre a `/dev/sda`.

```
# grub-install /dev/sda
```

Vegyük figyelembe, hogy meg kell adnunk azt az eszközt, ahová a GRUB Legacy telepítve lesz, például `/dev/sda/`, *nem pedig a partíciót*, mint a `/dev/sda1`.

Alapértelmezés szerint a GRUB a szükséges fájlokat a megadott eszközön lévő `/boot` mappába másolja. Ha más mappába szeretnénk másolni őket, használja a `--boot-directory=` paramétert, amelyet a teljes elérési útvonal követ, ahová a fájlokat másolni kell.

A GRUB Legacy telepítése a GRUB Shellből

Ha valamilyen okból nem tudjuk elindítani a rendszert, és újra kell telepítenünk a GRUB Legacy-t, akkor ezt a GRUB Legacy bootlemezen lévő GRUB shellből tehetjük meg.

A GRUB shellből (nyomjuk meg a `c`-t a boot menüben, hogy elérjük a `grub>` promptot), az első lépés a boot eszköz beállítása, amely a `/boot` mappát tartalmazza. Például, ha ez a mappa az első lemez első partícióján található, a parancs a következő:

```
grub> root (hd0,0)
```

Ha nem tudjuk, hogy melyik eszközön található a `/boot` mappa, akkor a GRUB megkeresi nekünk a `find` parancs segítségével, az alábbi módon:

```
grub> find /boot/grub/stage1
 (hd0,0)
```

Ezután állítsuk be a boot partíciót a fentiek szerint, majd a `setup` paranccsal telepítsük a GRUB Legacy-t az MBR-re, és másoljuk a szükséges fájlokat a lemezre:

```
grub> setup (hd0)
```

Ha kész, indítsuk újra a rendszert, aminek már normális módon kell bootolnia.

A GRUB Legacy menübejegyzéseinek és beállításainak konfigurálása

A GRUB Legacy menü bejegyzéseit és beállításait a `/boot/grub/menu.lst` fájl tárolja. Ez egy egyszerű szöveges fájl a parancsok és paraméterek listájával, amely közvetlenül szerkeszthető a választott szövegszerkesztővel.

A `#`-val kezdődő sorok megjegyzésnek számítanak, az üres sorokat pedig figyelmen kívül hagyásra kerülnek.

Egy menübejegyzés legalább három parancsot tartalmaz. Az első, a `title`, az operációs rendszer címét állítja be a menü képernyőjén. A második, a `root`, megmondja a GRUB Legacy-nak, hogy melyik eszközről vagy partícióról indítson.

A harmadik bejegyzés, a `kernel`, megadja a kernel image teljes elérési útvonalát, amelyet be kell tölteni, amikor a megfelelő bejegyzés kiválasztásra kerül. Vegyük figyelembe, hogy ez az útvonal a `root` paraméterben megadott eszközhöz képest relatív.

Egy egyszerű példa:

```
# Ez a sor egy komment
title Linux disztribúcióm
root (hd0,0)
kernel /vmlinuz root=/dev/hda1
```

A GRUB 2-vel ellentétben a GRUB Legacyben mind a lemezek, mind a partíciók számozása nullától kezdődik. Tehát a `root (hd0,0)` parancs a boot partíciót az első lemez (`hd0`) első partíciójának (`0`) fogja beállítani.

A `root` utasítás elhagyható, ha a `kernel` parancsban az elérési út előtt adjuk meg a rendszerindító eszközt. A szintaxis ugyanaz, tehát:

```
kernel (hd0,0)/vmlinuz root=dev/hda1
```

megfelel ennek:

```
root (hd0,0)
kernel /vmlinuz root=/dev/hda1
```

Mindkettő az első lemez első partíciójának (`hd0,0`) gyökérmappájából (`/`) tölti be a `vmlinuz` fájlt.

A `root=/dev/hda1` paraméter a `kernel` parancs után megmondja a Linux kernelnek, hogy melyik partíciót használja gyökér fájlrendszerként. Ez egy Linux kernel paraméter, nem pedig egy GRUB Legacy parancs.

NOTE A kernelparaméterekről bővebben a https://www.kernel.org/doc/html/v4.14/admin-guide/kernel-parameters.html oldalon olvashatunk.

Előfordulhat, hogy meg kell adnunk az operációs rendszer kezdeti RAM lemezképének helyét az `initrd` paraméterrel. A fájl teljes elérési útját meg lehet adni, mint a `kernel` paraméterben, és az elérési út előtt megadhatunk egy eszközt vagy partíciót is, pl.:

```
# Ez a sor egy komment
title Linux disztribúcióm
root (hd0,0)
kernel /vmlinuz root=/dev/hda1
initrd /initrd.img
```

A GRUB Legacy moduláris felépítésű, ahol a modulok (általában a `/boot/grub/i386-pc`-ban tárolt `.mod` fájlok) betölthetők, hogy extra funkciókat adjanak hozzá, mint például a szokatlan hardverek, fájlrendszerek vagy új tömörítési algoritmusok támogatása.

A modulok betöltése a `module` paranccsal történik, amelyet a megfelelő `.mod` fájl teljes elérési útvonala követ. Ne feledjük, hogy a kernelekhez és az initrd imagekhez hasonlóan ez az elérési út a `root` parancsban megadott eszközhöz képest relatív.

Az alábbi példa betölti a `915resolution` modult, amely a framebuffer felbontásának helyes beállításához szükséges az Intel 800-as vagy 900-as sorozatú videó chipsetekkel rendelkező rendszereken.


```
module /boot/grub/i386-pc/915resolution.mod
```

Más operációs rendszerek láncbetöltése

A GRUB Legacy használható a nem támogatott operációs rendszerek, például a Windows betöltésére a *chainloading* (láncbetöltés) nevű folyamat segítségével. Először a GRUB Legacy töltődik be, majd a megfelelő opció kiválasztásakor a kívánt rendszer bootloadere töltődik be.

Egy tipikus bejegyzés a Windows láncbetöltéséhez az alábbi módon néz ki:

```
# Windows betöltése
title Windows XP
root (hd0,1)
makeactive
chainload +1
boot
```

Menjünk végig az egyes paramétereken. Mint korábban, a `root (hd0,1)` megadja azt az eszközt és partíciót, ahol a betölteni kívánt operációs rendszer bootloadere található. Ebben a példában az első lemez *második* partícióját.

`makeactive`

egy olyan flaget állít be, amely jelzi, hogy ez egy aktív partíció. Ez csak a DOS elsődleges partícióinál működik.

`chainload +1`

a GRUB-nak azt mondja, hogy töltse be a bootpartíció első szektorát. Általában itt találhatóak a bootloaderek.

`boot`

lefuttatja a bootloadert és betölti a megfelelő operációs rendszert.

Gyakorló feladatok

1. Mi a GRUB 2 konfigurációs fájljának alapértelmezett útvonala?

2. Milyen lépések szükségesek a GRUB 2 beállításainak módosításához?

3. Melyik fájlhoz kell hozzáadni az egyéni GRUB 2 menübejegyzéseket?

4. Hol tárolódnak a GRUB Legacy menübejegyzései?

5. Hogyan léphetünk be a GRUB 2 vagy GRUB Legacy menüből a GRUB Shell-be?

Gondolkodtató feladatok

1. Képzeljük el, hogy a felhasználó úgy konfigurálja a GRUB Legacy-t, hogy az első lemez második partíciójáról bootoljon. A következő egyéni menübejegyzést látjuk:

   ```
   title Az Linux Disztróm
   root (hd0,2)
   kernel /vmlinuz root=/dev/hda1
   initrd /initrd.img
   ```

   Azonban a rendszer nem fog bootolni. Mi a hiba?

2. Képzeljük el, hogy van egy `/dev/sda` néven azonosított lemezünk több partícióval. Melyik paranccsal lehet kideríteni, hogy melyik a boot partíció a rendszeren?

3. Melyik parancs használható egy partíció UUID-jének kiderítésére?

4. Képzeljük el az alábbi GRUB 2 bejegyzést

   ```
   menuentry "Default OS" {
       set root=(hd0,1)
       linux /vmlinuz root=/dev/sda1 ro quiet splash
       initrd /initrd.img
   }
   ```

   Módosítsuk úgy, hogy a rendszer a `5dda0af3-c995-481a-a6f3-46dcd3b6998d` UUID-vel rendelkező lemezről bootoljon

5. Hogyan lehet beállítani, hogy a GRUB 2 10 másodpercet várjon az alapértelmezett menüpont indítása előtt?

6. Milyen parancsokkal lehet a GRUB Legacy shellből telepíteni a GRUB-ot a második lemez első partíciójára?

Összefoglalás

Ebben a leckében megtanultuk

- Mi az a bootloader.
- A GRUB Legacy és a GRUB 2 közötti különbségek.
- Mi az a boot partíció, és mi a tartalma.
- Hogyan telepíthető a GRUB Legacy és a GRUB 2.
- Hogyan kell konfigurálni a GRUB Legacy-t és a GRUB 2-t.
- Hogyan adhatunk hozzá egyéni menübejegyzéseket a GRUB Legacy és a GRUB 2 rendszerhez.
- Hogyan léphetünk kapcsolatba a GRUB Legacy és a GRUB 2 menüképernyőjével és konzoljával.
- Hogyan indíthatunk rendszert a GRUB Legacy vagy GRUB 2 shellből vagy rescue shellből.

A következő parancsokról volt szó ebben a leckében:

- `grub-install`
- `update-grub`
- `grub-mkconfig`

Válaszok a gyakorló feladatokra

1. Mi a GRUB 2 konfigurációs fájljának alapértelmezett útvonala?

   `/boot/grub/grub.cfg`

2. Milyen lépések szükségesek a GRUB 2 beállításainak módosításához?

   Változtassuk meg az `/etc/default/grub` fájlt, majd frissítsük a konfigurációt az `update-grub` paranccsal.

3. Melyik fájlhoz kell hozzáadni az egyéni GRUB 2 menübejegyzéseket?

   `/etc/grub.d/40_custom`

4. Hol tárolódnak a GRUB Legacy menübejegyzései?

   `/boot/grub/menu.lst`

5. Hogyan léphetünk be a GRUB 2 vagy GRUB Legacy menüből a GRUB Shell-be?

   A `c` lenyomásával a menü képernyőn.

Válaszok a gondolkodtató feladatokra

1. Képzeljük el, hogy a felhasználó úgy konfigurálja a GRUB Legacy-t, hogy az első lemez második partíciójáról bootoljon. A következő egyéni menübejegyzést látjuk:

   ```
   title A Linux Disztróm
   root (hd0,2)
   kernel /vmlinuz root=/dev/hda1
   initrd /initrd.img
   ```

   Azonban a rendszer nem fog bootolni. Mi a hiba?

   A boot partíció a hibás. Ne feledjük, hogy a GRUB 2-vel ellentétben a GRUB Legacy a nullától számolja a partíciókat. Tehát az első lemez második partíciójára a helyes parancs a `root (hd0,1)`.

2. Képzeljük el, hogy van egy `/dev/sda` néven azonosított lemezünk több partícióval. Melyik paranccsal lehet kideríteni, hogy melyik a boot partíció a rendszeren?

   Használjuk az `fdisk -l /dev/sda` parancsot. A boot partíciót csillaggal (`*`) fogja jelölni a listában.

3. Melyik parancs használható egy partíció UUID-jének kiderítésére?

   Használjuk a `ls -la /dev/disk/by-uuid/` parancsot és keressük meg azt az UUID-t, ami a partícióra mutat.

4. Képzeljük el az alábbi GRUB 2 bejegyzést

   ```
   menuentry "Default OS" {
       set root=(hd0,1)
       linux /vmlinuz root=/dev/sda1 ro quiet splash
       initrd /initrd.img
   }
   ```

   Módosítsuk úgy, hogy a rendszer a `5dda0af3-c995-481a-a6f3-46dcd3b6998d` UUID-vel rendelkező lemezről bootoljon!

   Meg kell változtatni a `set root` utasítást. A lemez és a partíció megadása helyett adjuk meg a GRUB-nak, hogy keresse meg a kívánt UUID-vel rendelkező partíciót.

```
menuentry "Default OS" {
    search --set=root --fs-uuid 5dda0af3-c995-481a-a6f3-46dcd3b6998d
    linux /vmlinuz root=/dev/sda1 ro quiet splash
    initrd /initrd.img
}
```

5. Hogyan lehet beállítani, hogy a GRUB 2 10 másodpercet várjon az alapértelmezett menüpont indítása előtt?

   Adjuk hozzá a `GRUB_TIMEOUT=10` paramétert az `/etc/default/grub`-hoz.

6. Milyen parancsokkal lehet a GRUB Legacy shellből telepíteni a GRUB-ot a második lemez első partíciójára?

```
grub> root (hd1,0)
grub> setup (hd1)
```

102.3 Megosztott könyvtárak kezelése

Hivatkozás az LPI célkitűzésre

LPIC-1 v5, Exam 101, Objective 102.3

Súlyozás

1

Kulcsfontosságú ismeretek

- Megosztott könyvtárak azonosítása.
- A rendszerkönyvtárak tipikus helyének azonosítása.
- Megosztott könyvtárak betöltése.

A használt fájlok, kifejezések és segédprogramok listája

- `ldd`
- `ldconfig`
- `/etc/ld.so.conf`
- `LD_LIBRARY_PATH`

Linux Professional Institute

102.3 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	102 A Linux telepítése és a csomagkezelés
Fejezet:	102.3 Megosztott könyvtárak kezelése
Lecke:	1/1

Bevezetés

Ebben a leckében a *megosztott könyvtárakról* (shared libraries), más néven *megosztott objektumokról* (shared objects) lesz szó: olyan lefordított, újrafelhasználható kódrészletek, mint például függvények vagy osztályok, amelyeket különböző programok újra és újra felhasználnak.

Először is nézzük meg, hogy mik azok a megosztott könyvtárak, hogyan lehet őket azonosítani, és hol találhatók. Ezután azt látjuk majd, hogy hogyan kell beállítani a tárolási helyüket, majd pedig azt, hogy hogyan kereshetjük meg azokat a megosztott könyvtárakat, amelyektől egy adott program függ.

A megosztott könyvtárak koncepciója

Fizikai társaikhoz hasonlóan a szoftverkönyvtárak is kódgyűjtemények, amelyek célja, hogy sok különböző program tudja használni őket; ahogyan a fizikai könyvtárak is könyveket és más forrásokat őriznek, amelyeket sok különböző ember használhat.

Ahhoz, hogy egy program forráskódjából futtatható fájlt készítsünk, két fontos lépésre van szükség. Először a *compiler* (fordító) a forráskódot gépi kóddá alakítja, amelyet úgynevezett

objektumfájlokban (object files) tárol. Másodszor, a *linker* egyesíti az objektumfájlokat és *linkeli* őket könyvtárakhoz, hogy létrehozza a végső futtatható fájlt. Ez a linkelés történhet *statikusan* vagy *dinamikusan*. Attól függően, hogy melyik módszert választjuk, statikus könyvtárakról, vagy dinamikus linkelés esetén megosztott könyvtárakról beszélünk. Most nézzük meg a különbségeket!

Statikus könyvtárak

A statikus könyvtár a programmal a linkeléskor egyesül. A könyvtár kódjának egy példánya beágyazódik a programba, és annak részévé válik. Így a program a futás során nem függ a könyvtáraktól, mivel a program már tartalmazza a könyvtár kódját. A függőségek hiánya előnynek tekinthető, mivel nem kell aggódni amiatt, hogy a használt könyvtárak mindig rendelkezésre álljanak. Hátránya, hogy a statikusan linkelt programok nagyobbak.

Megosztott (vagy dinamikus) könyvtárak

Megosztott könyvtárak esetén a linker egyszerűen gondoskodik arról, hogy a program helyesen hivatkozzon a könyvtárakra. A linker azonban nem másol könyvtárkódot a programfájlba. Futtatási időben azonban a megosztott könyvtárnak rendelkezésre kell állnia, hogy a program függőségeit kielégítse. Ez egy gazdaságos megközelítés a rendszer erőforrásainak kezelésére, mivel segít csökkenteni a programfájlok méretét, és a könyvtárnak csak egy példánya töltődik be a memóriába, még akkor is, ha azt több program használja.

Megosztott objektumfájl-elnevezési konvenciók

A megosztott könyvtár neve, más néven *soname*, egy három elemből álló mintát követ:

- A könyvtár neve (általában `lib` prefix-el)
- `so` (ami a “shared object” rövidítése)
- A könyvtár verziószáma

Egy példa: `libpthread.so.0`

Ezzel szemben a statikus könyvtárak nevei `.a` végződéssel végződnek, például `libpthread.a`.

NOTE

Mivel a megosztott könyvtárakat tartalmazó fájloknak a program végrehajtásakor rendelkezésre kell állniuk, a legtöbb Linux rendszer tartalmaz megosztott könyvtárakat. Mivel a statikus könyvtárakra egy külön fájlban csak akkor van szükség, amikor a program linkelt, előfordulhat, hogy egy végfelhasználói rendszeren egyáltalán nincsenek.

A `glibc` (GNU C library) jó példa a megosztott könyvtárra. Debian GNU/Linux 9.9-es rendszerben

a fájl neve `libc.so.6`. Az ilyen, meglehetősen általános fájlnevek általában szimbolikus hivatkozások, amelyek a könyvtárat tartalmazó tényleges fájlra mutatnak, amelynek neve tartalmazza a pontos verziószámot. A `glibc` esetében ez a szimbolikus link így néz ki:

```
$ ls -l /lib/x86_64-linux-gnu/libc.so.6
lrwxrwxrwx 1 root root 12 feb  6 22:17 /lib/x86_64-linux-gnu/libc.so.6 -> libc-2.24.so
```

Ez a minta, hogy a megosztott könyvtárak egy adott verzió által elnevezett fájljaira általánosabb fájlnevekkel hivatkozunk, bevett gyakorlat.

További példák a megosztott könyvtárakra: `libreadline` (amely lehetővé teszi a felhasználók számára a parancssorok szerkesztését beírás közben, és támogatja az Emacs és a vi szerkesztési módokat is), `libcrypt` (amely titkosítással, hasheléssel és kódolással kapcsolatos funkciókat tartalmaz), vagy `libcurl` (amely egy többprotokollos fájlátviteli könyvtár).

A megosztott könyvtárak leggyakoribb helyei a Linux rendszerben a következők:

- `/lib`
- `/lib32`
- `/lib64`
- `/usr/lib`
- `/usr/local/lib`

NOTE A megosztott könyvtárak koncepciója nem kizárólag a Linuxra jellemző. A Windowsban például DLL-nek hívják őket, ami a *dynamic linked libraries* rövidítése.

A megosztott könyvtárak elérési útjainak konfigurálása

A dinamikusan linkelt programokban található hivatkozásokat a dinamikus linker (`ld.so` vagy `ld-linux.so`) oldja fel a program futtatásakor. A dinamikus linker több könyvtárban is keres mappákat. Ezeket a mappákat a *library path* adja meg. A könyvtárak elérési útvonalát a `/etc` mappában, nevezetesen a `/etc/ld.so.conf` fájlban, illetve manapság már gyakrabban a `/etc/ld.so.conf.d` mappában található fájlokban állítjuk be. Az előbbi általában csak egyetlen `include` sort tartalmaz az utóbbiban található `*.conf` fájlok számára:

```
$ cat /etc/ld.so.conf
include /etc/ld.so.conf.d/*.conf
```

A `/etc/ld.so.conf.d` mappa `*.conf` fájlokat tartalmaz:

```
$ ls /etc/ld.so.conf.d/
libc.conf  x86_64-linux-gnu.conf
```

Ezeknek a `*.conf` fájloknak tartalmazniuk kell a megosztott könyvtárak mappáinak abszolút elérési útvonalát:

```
$ cat /etc/ld.so.conf.d/x86_64-linux-gnu.conf
# Multiarch support
/lib/x86_64-linux-gnu
/usr/lib/x86_64-linux-gnu
```

Az `ldconfig` parancs gondoskodik ezen konfigurációs fájlok beolvasásáról, a fent említett szimbolikus linkek létrehozásáról, amelyek segítenek az egyes könyvtárak megtalálásában, és végül a cache fájl `/etc/ld.so.cache` frissítéséről. Így az `ldconfig`-t minden alkalommal el kell indítani, amikor konfigurációs fájlokat adunk hozzá vagy frissítünk.

Az `ldconfig` hasznos kapcsolói:

`-v`, **`--verbose`**

Megjeleníti a könyvtárak verziószámait, az egyes könyvtárak nevét és a létrehozott hivatkozásokat:

```
$ sudo ldconfig -v
/usr/local/lib:
/lib/x86_64-linux-gnu:
    libnss_myhostname.so.2 -> libnss_myhostname.so.2
    libfuse.so.2 -> libfuse.so.2.9.7
    libidn.so.11 -> libidn.so.11.6.16
    libnss_mdns4.so.2 -> libnss_mdns4.so.2
    libparted.so.2 -> libparted.so.2.0.1
    (...)
```

Láthatjuk például, hogy a `libfuse.so.2` a tényleges megosztott objektumfájlhoz, a `libfuse.so.2.9.7`-hez van linkelve.

`-p`, **`--print-cache`**

Az aktuális gyorsítótárban tárolt könyvtárak és könyvtárjelöltek listájának kinyomtatása:

```
$ sudo ldconfig -p
1094 libs found in the cache `/etc/ld.so.cache'
    libzvbi.so.0 (libc6,x86-64) => /usr/lib/x86_64-linux-gnu/libzvbi.so.0
    libzvbi-chains.so.0 (libc6,x86-64) => /usr/lib/x86_64-linux-gnu/libzvbi-chains.so.0
    libzmq.so.5 (libc6,x86-64) => /usr/lib/x86_64-linux-gnu/libzmq.so.5
    libzeitgeist-2.0.so.0 (libc6,x86-64) => /usr/lib/x86_64-linux-gnu/libzeitgeist-
2.0.so.0
    (...)
```

Vegyük észre, hogy a gyorsítótár a linkek teljes soname-jét használja:

```
$ sudo ldconfig -p |grep libfuse
      libfuse.so.2 (libc6,x86-64) => /lib/x86_64-linux-gnu/libfuse.so.2
```

Ha hosszan listázzuk a `/lib/x86_64-linux-gnu/libfuse.so.2`-t, akkor megtaláljuk a hivatkozást a tényleges megosztott objektumfájlra, a `libfuse.so.2.9.7`-re, amely ugyanebben a mappában található:

```
$ ls -l /lib/x86_64-linux-gnu/libfuse.so.2
lrwxrwxrwx 1 root root 16 Aug 21  2018 /lib/x86_64-linux-gnu/libfuse.so.2 ->
libfuse.so.2.9.7
```

NOTE

Mivel írási hozzáférést igényel az `/etc/ld.so.cache` (tulajdonosa a root), vagy root-nak kell lennünk, vagy a `sudo`-t kell használnunk az `ldconfig` meghívásához. Az `ldconfig` kapcsolókkal kapcsolatos további információkért lsd. a manual oldalát.

A fent leírt konfigurációs fájlok mellett a `LD_LIBRARY_PATH` környezeti változót is használhatjuk a megosztott könyvtárak új elérési útvonalainak ideiglenes hozzáadására. Kettősponttal elválasztott (`:`) mappákból áll, ahol a könyvtárakat keressük. Ha például a `/usr/local/mylib` könyvtár elérési útvonalát szeretnénk hozzáadni az aktuális shell munkamenetben, akkor beírhatjuk a következőt:

```
$ LD_LIBRARY_PATH=/usr/local/mylib
```

Most ellenőrizhetjük az értékét:

```
$ echo $LD_LIBRARY_PATH
```

```
/usr/local/mylib
```

Ahhoz, hogy hozzáadjuk az `/usr/local/mylib`-t a shell munkamenet könyvtári elérési útvonalához, és ahhoz, hogy exportálhassuk a shellből indított összes gyermekfolyamatba, az alábbit kell beírni:

```
$ export LD_LIBRARY_PATH=/usr/local/mylib
```

Az `LD_LIBRARY_PATH` környezeti változó eltávolításához írjuk be az alábbit:

```
$ unset LD_LIBRARY_PATH
```

A változtatások állandósításához írjuk be a

```
export LD_LIBRARY_PATH=/usr/local/mylib
```

a sort a Bash egyik inicializáló szkriptjébe, mint például az `/etc/bash.bashrc` vagy a `~/.bashrc`.

NOTE A `LD_LIBRARY_PATH` a megosztott könyvtárak számára az, ami a `PATH` a végrehajtható programok számára. A környezeti változókról és a shell konfigurációjáról további információkat a témához tartozó leckékben találhatunk.

Egy futtatható program függőségeinek keresése

Egy adott program által igényelt megosztott könyvtárak kereséséhez használjuk az `ldd` parancsot, amelyet a program abszolút elérési útvonala követ. A kimenet megmutatja a megosztott könyvtárfájl elérési útvonalát, valamint azt a hexadecimális memóriacímet, ahová az betöltődött:

```
$ ldd /usr/bin/git
    linux-vdso.so.1 =>  (0x00007ffcbb310000)
    libpcre.so.3 => /lib/x86_64-linux-gnu/libpcre.so.3 (0x00007f18241eb000)
    libz.so.1 => /lib/x86_64-linux-gnu/libz.so.1 (0x00007f1823fd1000)
    libresolv.so.2 => /lib/x86_64-linux-gnu/libresolv.so.2 (0x00007f1823db6000)
    libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f1823b99000)
    librt.so.1 => /lib/x86_64-linux-gnu/librt.so.1 (0x00007f1823991000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f18235c7000)
    /lib64/ld-linux-x86-64.so.2 (0x00007f182445b000)
```

Hasonlóképpen, az `ldd`-t használjuk egy megosztott objektum függőségeinek keresésére:

```
$ ldd /lib/x86_64-linux-gnu/libc.so.6
    /lib64/ld-linux-x86-64.so.2 (0x00007fbfed578000)
    linux-vdso.so.1 (0x00007fffb7bf5000)
```

Az `-u` (vagy `--unused`) opcióval az `ldd` kiírja a nem használt közvetlen függőségeket (ha vannak):

```
$ ldd -u /usr/bin/git
Unused direct dependencies:
    /lib/x86_64-linux-gnu/libz.so.1
    /lib/x86_64-linux-gnu/libpthread.so.0
    /lib/x86_64-linux-gnu/librt.so.1
```

A nem használt függőségek oka a linker által a bináris program készítésekor használt opciókhoz kapcsolódik. Bár a programnak nincs szüksége egy nem használt könyvtárra, az mégis linkelésre került, és az objektumfájlról szóló információban `NEEDED`-ként lett megjelölve. Ezt olyan parancsokkal vizsgálhatjuk meg, mint a `readelf` vagy az `objdump`, amelyeket hamarosan a gondolkodtató feladatokban fogunk használni.

Gyakorló feladatok

1. Osszuk fel részekre a következő megosztott könyvtárneveket:

Teljes fájlnév	**Könyvtár neve**	***so* suffix**	**Verziószám**
`linux-vdso.so.1`			
`libprocps.so.6`			
`libdl.so.2`			
`libc.so.6`			
`libsystemd.so.0`			
`ld-linux-x86-64.so.2`			

2. Fejlesztettünk egy szoftvert és egy új megosztott könyvtárat szeretnénk hozzáadni a rendszerünkhöz (`/opt/lib/mylib`). Ennek abszolút elérési útját egy `mylib.conf` nevű fájlba írjuk.
   - Melyik mappában kell elhelyeznünk ezt a fájlt?
   - Milyen parancsot kell futtatnunk, hogy a módosítások teljeskörűen érvénybe lépjenek?

3. Milyen parancsot kellene használnunk a `kill`-hez szükséges megosztott könyvtárak listázására?

Gondolkodtató feladatok

1. Az `objdump` egy parancssori segédprogram, amely megjeleníti az objektumfájlok információit. Ellenőrizzük a `which objdump` segítségével, hogy telepítve van-e a rendszerünkben! Ha nincs, telepítsük fel!
   - Használjuk az `objdump`-ot a `-p` (vagy `--private-headers`) kapcsolóval és a `grep`-pel a `glibc` függőségek kiírásához:
   - Használjuk az `objdump`-t a `-p` (vagy `--private-headers`) kapcsolóval és a `grep`-pel a `glibc` soname kiírásához:
   - Használjuk az `objdump`-t a `-p` (vagy `--private-headers`) kapcsolóval és a `grep`-pel a Bash függőségek kiírásához:

Összefoglalás

Ebben a leckében megtanultuk:

- Mi a megosztott (vagy dinamikus) könyvtár (library).
- A megosztott és a statikus könyvtárak közötti különbségeket.
- A megosztott könyvtárak neveit (*sonames*).
- A megosztott könyvtárak előnyben részesített helyeit a Linux rendszerben, mint például a `/lib` vagy az `/usr/lib`.
- A dinamikus linker `ld.so` (vagy `ld-linux.so`) célját.
- A megosztott könyvtárak elérési útvonalainak beállítása az `/etc/` állományok segítségével, mint például az `ld.so.conf` vagy az `ld.so.conf.d` mappában található állományok.
- Hogyan konfigurálhatjuk a megosztott könyvtárak elérési útvonalait az `LD_LIBRARY_PATH` környezeti változó segítségével.
- Hogyan keressük meg a végrehajtható és megosztott könyvtárak függőségeit.

A leckében használt parancsok:

`ls`

Mappa tartalmának kilistázása.

`cat`

Fájlok összefűzése és standard outputra való kiíratása.

`sudo`

A parancsot a rendszergazdai jogosultságokkal rendelkező superuser hajtsa végre.

`ldconfig`

A dinamikus linker futásidejű hozzárendeléseinek konfigurálása.

`echo`

Egy környezeti változó értékének kiíratása.

`export`

A környezeti változó értékének exportálása a gyerek shellekbe.

`unset`

Környezeti változó eltávolítása.

`ldd`

Egy program megosztott objektumfüggőségének kiírása.

`readelf`

ELF fájlokkal kapcsolatos információk megjelenítése (az ELF a *executable and linkable format* (végrehajtható és linkelhető formátum) rövidítése).

`objdump`

Objektumfájlok információinak kiírása.

Válaszok a gyakorló feladatokra

1. Osszuk fel részekre a következő megosztott könyvtárneveket:

Teljes fájlnév	**Könyvtár neve**	***so* suffix**	**Verziószám**
`linux-vdso.so.1`	`linux-vdso`	`so`	`1`
`libprocps.so.6`	`libprocps`	`so`	`6`
`libdl.so.2`	`libdl`	`so`	`2`
`libc.so.6`	`libc`	`so`	`6`
`libsystemd.so.0`	`libsystemd`	`so`	`0`
`ld-linux-x86-64.so.2`	`ld-linux-x86-64`	`so`	`2`

2. Fejlesztettünk egy szoftvert és egy új megosztott könyvtárat szeretnénk hozzáadni a rendszerünkhöz (`/opt/lib/mylib`). Ennek abszolút elérési útját egy `mylib.conf` nevű fájlba írjuk.
   - Melyik mappában kell elhelyeznünk ezt a fájlt?

     `/etc/ld.so.conf.d`

   - Milyen parancsot kell futtatnunk, hogy a módosítások teljeskörűen érvénybe lépjenek?

     `ldconfig`

3. Milyen parancsot kellene használnunk a `kill`-hez szükséges megosztott könyvtárak listázására?

   `ldd /bin/kill`

Válaszok a gondolkodtató feladatokra

1. Az `objdump` egy parancssori segédprogram, amely megjeleníti az objektumfájlok információit. Ellenőrizzük a `which objdump` segítségével, hogy telepítve van-e a rendszerünkben! Ha nincs, telepítsük fel!
   - Használjuk az `objdump`-ot a `-p` (vagy `--private-headers`) kapcsolóval és a `grep`-pel a `glibc` függőségek kiírásához:

     ```
     objdump -p /lib/x86_64-linux-gnu/libc.so.6 | grep NEEDED
     ```

   - Használjuk az `objdump`-t a `-p` (vagy `--private-headers`) kapcsolóval és a `grep`-pel a `glibc` soname kiírásához:

     ```
     objdump -p /lib/x86_64-linux-gnu/libc.so.6 | grep SONAME
     ```

   - Használjuk az `objdump`-t a `-p` (vagy `--private-headers`) kapcsolóval és a `grep`-pel a Bash függőségek kiírásához:

     ```
     objdump -p /bin/bash | grep NEEDED
     ```

102.4 A Debian csomagkezelő használata

Hivatkozás az LPI célkitűzésre

LPIC-1 v5, Exam 101, Objective 102.4

Súlyozás

3

Kulcsfontosságú ismeretek

- Debian bináris csomagok telepítése, frissítése és eltávolítása.
- Olyan csomagok keresése, amelyek bizonyos fájlokat vagy könyvtárakat tartalmaznak, amelyek telepítve vannak vagy nincsenek.
- Olyan csomaginformációk beszerzése, mint a verzió, tartalom, függőségek, csomagintegritás és telepítési állapot (telepítve van-e a csomag vagy sem).
- Az apt ismerete.

A használt fájlok, kifejezések és segédprogramok listája

- `/etc/apt/sources.list`
- `dpkg`
- `dpkg-reconfigure`
- `apt-get`
- `apt-cache`

102.4 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	102 A Linux telepítése és a csomagkezelés
Fejezet:	102.4 A Debian csomagkezelő használata
Lecke:	1/1

Bevezetés

Réges-régen, amikor a Linux még gyerekcipőben járt, a szoftverek terjesztésének legelterjedtebb módja egy tömörített fájl (általában egy `.tar.gz` archívum) volt a forráskóddal, amelyet az ember saját maga csomagolt ki és fordított le.

A szoftverek mennyiségének és összetettségének növekedésével azonban egyértelművé vált az előre lefordított szoftverek terjesztésének szükségessége. Végülis nem mindenki rendelkezett az idő és a számítási teljesítmény tekintetében elegendő erőforrással ahhoz, hogy olyan nagy projekteket fordítson le, mint a Linux kernel vagy az X Server.

Hamarosan egyre nagyobb erőfeszítéseket tettek arra, hogy szabványosítsák a “package”-k (szoftvercsomagok) terjesztésének módját, és megszülettek az első csomagkezelők. Ezek az eszközök nagyban megkönnyítették a szoftverek telepítését, konfigurálását vagy eltávolítását a rendszerből.

Ezek egyike a Debian csomagformátum (`.deb`) és a hozzá tartozó csomagkezelő eszköz (`dpkg`) volt. Ma már széles körben használják őket nemcsak magán a Debianon, hanem az abból leszármazottakon is, mint például az Ubuntu és a belőle kialakultak.

Egy másik, Debian-alapú rendszereken népszerű csomagkezelő eszköz a *Advanced Package Tool* (`apt`), amely a csomagok telepítésének, karbantartásának és eltávolításának számos aspektusának egyszerűsítésével könnyíti meg a munkát.

Ebben a leckében meg fogjuk tanulni, hogyan használjuk az `dpkg` és az `apt` programokat a szoftverek beszerzésére, telepítésére, karbantartására és eltávolítására egy Debian-alapú Linux rendszeren.

A Debian Package Tool (dpkg)

The *Debian Package* (`dpkg`) tool is the essential utility to install, configure, maintain and remove software packages on Debian-based systems. The most basic operation is to install a `.deb` package, which can be done with:

```
# dpkg -i PACKAGENAME
```

Where `PACKAGENAME` is the name of the `.deb` file you want to install.

A csomagok frissítéseit ugyanígy kezeljük. Egy csomag telepítése előtt a `dpkg` ellenőrzi, hogy létezik-e már egy korábbi verzió a rendszerben. Ha igen, akkor a csomagot frissíti az új verzióra. Ha nem, akkor egy friss példány kerül telepítésre.

A függőségek kezelése

Gyakran előfordul, hogy egy csomag rendeltetésszerű működése másoktól függ. Egy képszerkesztőnek például szüksége lehet könyvtárakra a JPEG-fájlok megnyitásához, vagy egy másik segédprogram felhasználói felületének szüksége lehet egy olyan widget eszköztárra, mint a Qt vagy a GTK.

A `dpkg` ellenőrizni fogja, hogy ezek a függőségek telepítve vannak-e a rendszerünkön, és ha nem, akkor nem telepíti a csomagot. Ebben az esetben a `dpkg` felsorolja, hogy mely csomagok hiányoznak. A függőségeket azonban önmagában *nem tudja* megoldani. A felhasználó feladata, hogy megkeresse a megfelelő függőségeket tartalmazó `.deb` csomagokat és telepítse azokat.

Az alábbi példában a felhasználó megpróbálja telepíteni az OpenShot videószerkesztő csomagot, de néhány függőség hiányzik:

```
# dpkg -i openshot-qt_2.4.3+dfsg1-1_all.deb
(Reading database ... 269630 files and directories currently installed.)
Preparing to unpack openshot-qt_2.4.3+dfsg1-1_all.deb ...
Unpacking openshot-qt (2.4.3+dfsg1-1) over (2.4.3+dfsg1-1) ...
```

```
dpkg: dependency problems prevent configuration of openshot-qt:
 openshot-qt depends on fonts-cantarell; however:
  Package fonts-cantarell is not installed.
 openshot-qt depends on python3-openshot; however:
  Package python3-openshot is not installed.
 openshot-qt depends on python3-pyqt5; however:
  Package python3-pyqt5 is not installed.
 openshot-qt depends on python3-pyqt5.qtsvg; however:
  Package python3-pyqt5.qtsvg is not installed.
 openshot-qt depends on python3-pyqt5.qtwebkit; however:
  Package python3-pyqt5.qtwebkit is not installed.
 openshot-qt depends on python3-zmq; however:
  Package python3-zmq is not installed.

dpkg: error processing package openshot-qt (--install):
 dependency problems - leaving unconfigured
Processing triggers for mime-support (3.60ubuntu1) ...
Processing triggers for gnome-menus (3.32.0-1ubuntu1) ...
Processing triggers for desktop-file-utils (0.23-4ubuntu1) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for man-db (2.8.5-2) ...
Errors were encountered while processing:
 openshot-qt
```

Mint fentebb látható, az OpenShot függ a `fonts-cantarell`, `python3-openshot`, `python3-pyqt5`, `python3-pyqt5.qtsvg`, `python3-pyqt5.qtwebkit` és `python3-zmq` csomagoktól. Ezek mindegyikét telepíteni kell az OpenShot sikeres telepítéséhez.

Csomagok eltávolítása

Egy csomag eltávolításához adjuk át a `-r` paramétert a `dpkg`-nak, utána pedig a csomag nevét. Például a következő parancs eltávolítja a rendszerből az `unrar` csomagot:

```
# dpkg -r unrar
(Reading database ... 269630 files and directories currently installed.)
Removing unrar (1:5.6.6-2) ...
Processing triggers for man-db (2.8.5-2) ...
```

Az eltávolítási művelet függőségi ellenőrzést is végez, és egy csomag csak akkor távolítható el, ha minden más, tőle függő csomag is eltávolításra kerül. Ha ezt megpróbáljuk, az alábbi hibaüzenethez hasonlót kapunk:

```
# dpkg -r p7zip
dpkg: dependency problems prevent removal of p7zip:
 winetricks depends on p7zip; however:
  Package p7zip is to be removed.
 p7zip-full depends on p7zip (= 16.02+dfsg-6).

dpkg: error processing package p7zip (--remove):
 dependency problems - not removing
Errors were encountered while processing:
 p7zip
```

Több csomag nevét is megadhatjuk a `dpkg -r` parancsnak, így az összes csomag egyszerre lesz eltávolítva.

Amikor egy csomagot eltávolítanak, a megfelelő konfigurációs fájlok a rendszerben maradnak. Ha *mindent* el akarunk távolítani, ami a csomaghoz tartozik, használjuk a `-P` (purge) opciót az `-r` helyett.

NOTE

A `dpkg` kényszerítheti a csomag telepítését vagy eltávolítását, még akkor is, ha a függőségek nem teljesülnek. Ezt a a `--force` paraméter hozzáadásával érhetjük el, mint például a `dpkg -i --force PACKAGENAME`. Ez azonban nagy valószínűséggel a telepített csomagot, vagy akár a rendszert is hibás állapotban hagyja. *Ne használjuk* a `--force` paramétert, hacsak nem vagyunk teljesen biztos abban, hogy mit csinálunk.

Csomaginformációk megszerzése

Ha információt szeretnénk kapni egy `.deb` csomagról, például a verziójáról, architektúrájáról, karbantartójáról, függőségeiről és így tovább, használjuk a `dpkg` parancsot a `-I` paraméterrel, amelyet a vizsgálni kívánt csomag fájlneve követ:

```
# dpkg -I google-chrome-stable_current_amd64.deb
 new Debian package, version 2.0.
 size 59477810 bytes: control archive=10394 bytes.
    1222 bytes,    13 lines      control
   16906 bytes,   457 lines   *  postinst             #!/bin/sh
   12983 bytes,   344 lines   *  postrm               #!/bin/sh
    1385 bytes,    42 lines   *  prerm                #!/bin/sh
 Package: google-chrome-stable
 Version: 76.0.3809.100-1
 Architecture: amd64
```

```
 Maintainer: Chrome Linux Team <chromium-dev@chromium.org>
 Installed-Size: 205436
 Pre-Depends: dpkg (>= 1.14.0)
 Depends: ca-certificates, fonts-liberation, libappindicator3-1, libasound2 (>= 1.0.16),
libatk-bridge2.0-0 (>= 2.5.3), libatk1.0-0 (>= 2.2.0), libatspi2.0-0 (>= 2.9.90), libc6 (>=
2.16), libcairo2 (>= 1.6.0), libcups2 (>= 1.4.0), libdbus-1-3 (>= 1.5.12), libexpat1 (>=
2.0.1), libgcc1 (>= 1:3.0), libgdk-pixbuf2.0-0 (>= 2.22.0), libglib2.0-0 (>= 2.31.8),
libgtk-3-0 (>= 3.9.10), libnspr4 (>= 2:4.9-2~), libnss3 (>= 2:3.22), libpango-1.0-0 (>=
1.14.0), libpangocairo-1.0-0 (>= 1.14.0), libuuid1 (>= 2.16), libx11-6 (>= 2:1.4.99.1),
libx11-xcb1, libxcb1 (>= 1.6), libxcomposite1 (>= 1:0.3-1), libxcursor1 (>> 1.1.2),
libxdamage1 (>= 1:1.1), libxext6, libxfixes3, libxi6 (>= 2:1.2.99.4), libxrandr2 (>=
2:1.2.99.3), libxrender1, libxss1, libxtst6, lsb-release, wget, xdg-utils (>= 1.0.2)
 Recommends: libu2f-udev
 Provides: www-browser
 Section: web
 Priority: optional
 Description: The web browser from Google
  Google Chrome is a browser that combines a minimal design with sophisticated technology to
make the web faster, safer, and easier.
```

Telepített csomagok és csomagtartalmak listázása

A rendszerünkre telepített összes csomag listáját a `--get-selections` opcióval kaphatjuk meg, mint a `dpkg --get-selections` parancsban. Egy adott csomag által telepített összes fájl listáját is megkaphatjuk, ha a `-L PACKAGENAME` paramétert adjuk meg a `dpkg`-nak, ahogy ezt alább láthatjuk:

```
# dpkg -L unrar
/.
/usr
/usr/bin
/usr/bin/unrar-nonfree
/usr/share
/usr/share/doc
/usr/share/doc/unrar
/usr/share/doc/unrar/changelog.Debian.gz
/usr/share/doc/unrar/copyright
/usr/share/man
/usr/share/man/man1
/usr/share/man/man1/unrar-nonfree.1.gz
```

Annak megállapítása, hogy melyik csomagban van egy adott fájl

Néha szükség lehet arra, hogy megtudjuk, melyik csomag tartalmaz egy adott fájlt a rendszerben. Ezt a `dpkg-query` segédprogrammal nézhetjük meg, amelyet az `-S` paraméter és a kérdéses fájl elérési útvonala követ:

```
# dpkg-query -S /usr/bin/unrar-nonfree
unrar: /usr/bin/unrar-nonfree
```

Telepített csomagok átkonfigurálása

Amikor egy csomag telepítése megtörténik, van egy konfigurációs lépés, az úgynevezett *post-install*, ahol egy szkript fut le, amely beállít mindent, ami a szoftver futtatásához szükséges, például a jogosultságokat, a konfigurációs fájlok elhelyezését stb. Ez néhány kérdést is feltehet a felhasználónak, hogy beállítsa a szoftver futtatásával kapcsolatos preferenciákat.

Néha egy sérült vagy rosszul formázott konfigurációs fájl miatt előfordulhat, hogy vissza szeretnénk állítani egy csomag beállításait a “friss” állapotba, vagy meg szeretnénk változtatni a kezdeti konfigurációs kérdésekre adott válaszokat. Ehhez futtassuk a `dpkg-reconfigure` segédprogramot a csomag nevével.

Ez a program biztonsági mentést készít a régi konfigurációs fájlokról, kicsomagolja az újakat a megfelelő mappákba, és lefuttatja a csomag által biztosított *post-install* szkriptet, mintha a csomagot először telepítettük volna. Próbáljuk meg a `tzdata` csomag újrakonfigurálását a következő példával:

```
# dpkg-reconfigure tzdata
```

Advanced Package Tool (apt)

Az *Advanced Package Tool* (APT) egy olyan csomagkezelő rendszer, amely egy eszközkészletet tartalmaz, és jelentősen leegyszerűsíti a csomagok telepítését, frissítését, eltávolítását és kezelését. Az APT olyan funkciókat biztosít, mint a fejlett keresési lehetőségek és az automatikus függőség-feloldás.

Az APT nem “helyettesíti” a `dpkg`-t. Gondolhatunk rá úgy, mint egy “front end”-re, amely leegyszerűsíti a műveleteket és kitölti a `dpkg` funkcionalitásbeli hiányosságait, mint például a függőségek feloldása.

Az APT a telepíthető csomagokat tartalmazó szoftver repositorykkal együttműködve működik.

Ilyen repository lehet egy helyi vagy távoli szerver, vagy (ritkábban) akár egy CD-ROM lemez is.

A Linux disztribúciók, mint például a Debian és az Ubuntu, saját repositorykat üzemeltetnek, de fejlesztők vagy felhasználói csoportok egyéb repository-kat tarthatnak fent, hogy a disztribúció fő repositoryjában nem elérhető szoftvereket biztosítsanak.

Számos segédprogram működik együtt az APT-vel, a legfontosabbak a következők:

`apt-get`
: csomagok letöltésére, telepítésére, frissítésére vagy eltávolítására szolgál a rendszerből.

`apt-cache`
: a csomagindexben történő műveletek, például keresések elvégzésére szolgál.

`apt-file`
: a csomagokon belüli fájlok keresésére szolgál.

Létezik egy “barátságosabb” segédprogram is, amelynek neve egyszerűen `apt`, és amely az `apt-get` és az `apt-cache` leggyakrabban használt opcióit egyesíti egy segédprogramban. Az `apt` sok parancsa megegyezik az `apt-get` parancsaival, így sok esetben felcserélhetők. Mivel azonban előfordulhat, hogy az `apt` nincs telepítve a rendszerre, ajánlott megtanulni az `apt-get` és az `apt-cache` használatát.

NOTE Az `apt` és az `apt-get` hálózati kapcsolatot igényelhet, mivel előfordulhat, hogy a csomagokat és a csomagindexeket egy távoli kiszolgálóról kell letölteni.

A csomagindex frissítése

A szoftverek APT-vel történő telepítése vagy frissítése előtt ajánlott először frissíteni a csomagindexet, hogy az új és frissített csomagokra vonatkozó információkhoz hozzájussunk. Ezt az `apt-get` paranccsal, majd az `update` paraméterrel lehet megtenni:

```
# apt-get update
Ign:1 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:2 https://repo.skype.com/deb stable InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu disco InRelease
Hit:4 http://repository.spotify.com stable InRelease
Hit:5 http://dl.google.com/linux/chrome/deb stable Release
Hit:6 http://apt.pop-os.org/proprietary disco InRelease
Hit:7 http://ppa.launchpad.net/system76/pop/ubuntu disco InRelease
Hit:8 http://us.archive.ubuntu.com/ubuntu disco-security InRelease
Hit:9 http://us.archive.ubuntu.com/ubuntu disco-updates InRelease
```

```
Hit:10 http://us.archive.ubuntu.com/ubuntu disco-backports InRelease
Reading package lists... Done
```

TIP Az `apt-get update` helyett használható az `apt update` is.

Csomagok telepítése és törlése

A csomagindex frissítése után most már telepíthetünk egy csomagot. Ezt az `apt-get install` paranccsal lehet megtenni, amelyet a telepíteni kívánt csomag neve követ:

```
# apt-get install xournal
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  xournal
0 upgraded, 1 newly installed, 0 to remove and 75 not upgraded.
Need to get 285 kB of archives.
After this operation, 1041 kB of additional disk space will be used.
```

Egy csomag eltávolításához pedig használjuk az `apt-get remove` parancsot, amelyet a csomag neve követ:

```
# apt-get remove xournal
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  xournal
0 upgraded, 0 newly installed, 1 to remove and 75 not upgraded.
After this operation, 1041 kB disk space will be freed.
Do you want to continue? [Y/n]
```

Jegyezzük meg, hogy a csomagok telepítésekor vagy eltávolításakor az APT automatikusan feloldja a függőségeket. Ez azt jelenti, hogy a telepítendő csomaghoz szükséges további csomagok is telepítésre kerülnek, és az eltávolítandó csomagtól függő csomagok is eltávolításra kerülnek. Az APT mindig megmutatja, hogy mi lesz telepítve vagy eltávolítva, mielőtt megkérdezné, hogy folytatni szeretnénk-e:

```
# apt-get remove p7zip
```

```
Reading package lists... Done
Building dependency tree
The following packages will be REMOVED:
  android-libbacktrace android-libunwind android-libutils
  android-libziparchive android-sdk-platform-tools fastboot p7zip p7zip-full
0 upgraded, 0 newly installed, 8 to remove and 75 not upgraded.
After this operation, 6545 kB disk space will be freed.
Do you want to continue? [Y/n]
```

Vegyük figyelembe, hogy amikor egy csomagot eltávolítunk, a megfelelő konfigurációs fájlok a rendszerben maradnak. A csomag *és* a konfigurációs fájlok eltávolításához a `remove` helyett használjuk a `purge` paramétert, vagy a `remove` paramétert a `--purge` kapcsolóval:

```
# apt-get purge p7zip
```

vagy

```
# apt-get remove --purge p7zip
```

TIP Használható az `apt install` vagy `apt remove` is.

Törött függőségek javítása

Lehetséges, hogy egy rendszeren “törött függőségek” (broken dependencies) vannak. Ez azt jelenti, hogy egy vagy több telepített csomag más csomagoktól függ, amelyek nem lettek telepítve, vagy már nincsenek jelen. Ez történhet egy APT hiba, vagy egy manuálisan telepített csomag miatt.

Ennek megoldásához használjuk az `apt-get install -f` parancsot. Ez megpróbálja `javítani` a hibás csomagokat a hiányzó függőségek telepítésével, biztosítva, hogy az összes csomag ismét konzisztens legyen.

TIP Használható az `apt install -f` is.

Csomag upgrade-elése

Az APT segítségével a telepített csomagokat automatikusan upgrade-elhetjük a repositorykból elérhető legújabb verziókra. Ez az `apt-get upgrade` paranccsal történik. A parancs futtatása előtt először frissítsük a csomagindexet az `apt-get update` paranccsal:

```
# apt-get update
```

```
Hit:1 http://us.archive.ubuntu.com/ubuntu disco InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu disco-security InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu disco-updates InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu disco-backports InRelease
Reading package lists... Done

# apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages have been kept back:
  gnome-control-center
The following packages will be upgraded:
  cups cups-bsd cups-client cups-common cups-core-drivers cups-daemon
  cups-ipp-utils cups-ppdc cups-server-common firefox-locale-ar (...)

74 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
Need to get 243 MB of archives.
After this operation, 30.7 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

A kimenet alján található összefoglaló megmutatja, hogy hány csomag kerül frissítésre, telepítésre, eltávolításra vagy visszatartásra, valamint kiírja a teljes letöltési méretet és azt, hogy mennyi extra lemezterületre lesz szükség a művelet elvégzéséhez. A frissítés befejezéséhez lenyomjuk az `Y` -t, majd megvárjuk, amíg az `apt-get` befejezi a műveletet.

Egyetlen csomag upgrade-eléséhez csak futtassuk az `apt-get upgrade` parancsot, amelyet a csomag neve követ. A `dpkg`-hoz hasonlóan az `apt-get` először azt ellenőrzi, hogy a csomagnak van-e telepítve egy korábbi verziója. Ha igen, a csomagot a repositoryban elérhető legújabb verzióra upgrade-eli. Ha nem, akkor egy friss példány kerül telepítésre.

TIP Használható az `apt upgrade` és az `apt update` is.

A lokális gyorsítótár

Amikor telepítünk vagy frissítünk egy csomagot, a csomag telepítése előtt a megfelelő `.deb` fájl letöltődik egy helyi gyorsítótárba (local cache). Alapértelmezés szerint ez a mappa a `/var/cache/apt/archives`. A részben letöltött fájlok a `/var/cache/apt/archives/partial/` mappába másolódnak.

A csomagok telepítése és frissítése során a gyorsítótár elég nagyra nőhet. A hely visszaszerzéséhez kiüríthetjük a gyorsítótárat az `apt-get clean` paranccsal. Ez eltávolítja a

`/var/cache/apt/archives` és a `/var/cache/apt/archives/partial/` mappák tartalmát.

TIP Használható az `apt clean` is.

Csomagok keresése

Az `apt-cache` segédprogrammal műveleteket végezhetünk a csomagindexen, például megkereshetünk egy adott csomagot, vagy listázhatjuk, hogy mely csomagok tartalmaznak egy adott fájlt.

A kereséshez használjuk az `apt-cache search` parancsot, amelyet egy keresési minta követ. A kimenet minden olyan csomag listája lesz, amely tartalmazza a mintát, akár a csomag nevében, akár a leírásában, akár a fájlokban.

```
# apt-cache search p7zip
liblzma-dev - XZ-format compression library - development files
liblzma5 - XZ-format compression library
forensics-extra - Forensics Environment - extra console components (metapackage)
p7zip - 7zr file archiver with high compression ratio
p7zip-full - 7z and 7za file archivers with high compression ratio
p7zip-rar - non-free rar module for p7zip
```

A fenti példában a `liblzma5 - XZ-format compression library` bejegyzés nem felel meg a mintának. Ha azonban a `show` paraméterrel megjelenítjük a csomag teljes információját, beleértve a leírást is, akkor ott megtaláljuk a mintát:

```
# apt-cache show liblzma5
Package: liblzma5
Architecture: amd64
Version: 5.2.4-1
Multi-Arch: same
Priority: required
Section: libs
Source: xz-utils
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Jonathan Nieder <jrnieder@gmail.com>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 259
Depends: libc6 (>= 2.17)
Breaks: liblzma2 (<< 5.1.1alpha+20110809-3~)
Filename: pool/main/x/xz-utils/liblzma5_5.2.4-1_amd64.deb
```

```
Size: 92352
MD5sum: 223533a347dc76a8cc9445cfc6146ec3
SHA1: 8ed14092fb1caecfebc556fda0745e1e74ba5a67
SHA256: 01020b5a0515dbc9a7c00b464a65450f788b0258c3fbb733ecad0438f5124800
Homepage: https://tukaani.org/xz/
Description-en: XZ-format compression library
 XZ is the successor to the Lempel-Ziv/Markov-chain Algorithm
 compression format, which provides memory-hungry but powerful
 compression (often better than bzip2) and fast, easy decompression.
 .
 The native format of liblzma is XZ; it also supports raw (headerless)
 streams and the older LZMA format used by lzma. (For 7-Zip's related
 format, use the p7zip package instead.)
```

A keresési mintában használhatunk *reguláris kifejezéseket* is, amik nagyon összetett (és pontos) kereséseket tesznek lehetővé. Ez a téma azonban nem tartozik ennek a leckének a témakörébe.

TIP Használható az `apt search` az `apt-cache search` helyett és az `apt show` az `apt-cache show` helyett.

A források listája

Az APT egy forráslistát használ ahhoz, hogy tudja, honnan szerezzen be csomagokat. Ezt a listát a `sources.list` fájlban tárolja, ami az `/etc/apt` könyvtárban található. Ez a fájl szerkeszthető közvetlenül egy szövegszerkesztővel, mint például `vi`, `pico` vagy `nano`, vagy grafikus segédprogramokkal, mint például az `aptitude` vagy a `synaptic`.

Egy tipikus sor a `sources.list` fájlban így néz ki:

```
deb http://us.archive.ubuntu.com/ubuntu/ disco main restricted universe multiverse
```

A szintaxis az is archívum típusa, az URL, a disztribúció és egy vagy több komponens, ahol:

Archívum típusa

A repository tartalmazhat kész szoftvereket tartalmazó csomagokat (bináris csomagok, `deb` típus) vagy a szoftver forráskódját tartalmazó csomagokat (forráscsomagok, `deb-src` típus). A fenti példa bináris csomagokat biztosít.

URL

A repository URL-je.

Disztribúció

Annak a disztribúciónak a neve (vagy kódneve), amelyhez a csomagok rendelkezésre állnak. Egy repository több disztribúcióhoz is tartalmazhat csomagokat. A fenti példában a `disco` az Ubuntu 19.04 *Disco Dingo* kódneve.

Komponensek

Minden komponens csomagok halmazát jelenti. Ezek a komponensek különböző Linux disztribúciókban eltérőek lehetnek. Például az Ubuntu és leszármazottai esetében ezek a következők:

`main`

hivatalosan támogatott, nyílt forráskódú csomagokat tartalmaz.

`restricted`

hivatalosan támogatott, zárt forráskódú szoftvereket tartalmaz, mint például a grafikus kártyák eszközmeghajtóit.

`universe`

közösség által karbantartott nyílt forráskódú szoftvereket tartalmaz.

`multiverse`

nem támogatott, zárt forráskódú vagy szabadalommal terhelt szoftvereket tartalmaz.

Debian esetén a fő komponensek:

`main`

a *Debian Free Software Guidelines* (DFSG) szabványnak megfelelő csomagokból áll, amelyek a működéshez nem támaszkodnak ezen a területen kívüli szoftverekre. Az itt szereplő csomagok a Debian disztribúció részének tekintendők.

`contrib`

DFSG-kompatibilis csomagokat tartalmaz, amelyek azonban más csomagoktól függenek, amelyek nem a `main`-ben vannak.

`non-free`

olyan csomagokat tartalmaz, amelyek nem felelnek meg a DFSG-nek.

`security`

biztonsági frissítéseket tartalmaz.

`backports`

a `main` csomagok újabb verzióit tartalmazza. A Debian stabil verzióinak fejlesztési ciklusa elég hosszú (kb. két év), és ez biztosítja, hogy a felhasználók a legfrissebb csomagokat kapják meg anélkül, hogy a `main` core repositoryt kellene módosítaniuk.

NOTE A *Debian Free Software Guidelines*-ról többet ezen a linken tudhatunk meg: https://www.debian.org/social_contract#guidelines

Egy olyan, új repository hozzáadásához, ahonnan csomagokat szerezhetünk be, egyszerűen adjuk hozzá a megfelelő sort (általában a repository fenntartója adja meg) a `sources.list` végéhez, mentsük el a fájlt, és töltsük be újra a csomagindexet az `apt-get update` segítségével. Ezután az új repository-ban lévő csomagok telepíthetővé válnak az `apt-get install` segítségével.

Ne felejtsük el, hogy a `#` karakterrel kezdődő sorok kommentnek számítanak, és figyelmen kívül maradnak.

Az `/etc/apt/sources.list.d` mappa

Az `/etc/apt/sources.list.d` mappán belül további, az APT által használt repositorykat tartalmazó fájlokat adhatunk hozzá anélkül, hogy a fő `/etc/apt/sources.list` fájlt módosítani kellene. Ezek egyszerű szöveges fájlok, a fent leírt szintaxissal és a `.list` kiterjesztéssel.

Az alábbiakban az `/etc/apt/sources.list.d/buster-backports.list` nevű fájl tartalma látható:

```
deb http://deb.debian.org/debian buster-backports main contrib non-free
deb-src http://deb.debian.org/debian buster-backports main contrib non-free
```

A csomag tartalmának kilistázása és fájlok megtalálása

Az `apt-file` nevű segédprogrammal több műveletet is végezhetünk a csomagindexben, például listázhatjuk egy csomag tartalmát, vagy megkereshetjük azt a csomagot, amely egy adott fájlt tartalmaz. Ez a segédprogram nem biztos, hogy alapértelmezés szerint telepítve van a rendszeren. Ebben az esetben általában az `apt-get` segítségével telepíthetjük:

```
# apt-get install apt-file
```

A telepítés után frissítenünk kell az `apt-file` gyorsítótárát:

```
# apt-file update
```

Ez általában csak néhány másodpercet vesz igénybe. Ezután az `apt-file` kész a használatra.

Egy csomag tartalmának listázásához használjuk a `list` paramétert, amelyet a csomag neve követ:

```
# apt-file list unrar
unrar: /usr/bin/unrar-nonfree
unrar: /usr/share/doc/unrar/changelog.Debian.gz
unrar: /usr/share/doc/unrar/copyright
unrar: /usr/share/man/man1/unrar-nonfree.1.gz
```

TIP Használható az `apt list` az `apt-file list` helyett.

Az összes csomagban kereshetünk egy fájlt a `search` paraméterrel, amelyet a fájl neve követ. Ha például tudni szeretnénk, hogy melyik csomagban található a `libSDL2.so` nevű fájl, akkor a következő módon kereshetjük meg:

```
# apt-file search libSDL2.so
libsdl2-dev: /usr/lib/x86_64-linux-gnu/libSDL2.so
```

A válasz a `libsdl2-dev` csomag, amely az `/usr/lib/x86_64-linux-gnu/libSDL2.so` fájlt tartalmazza.

A különbség az `apt-file search` és a `dpkg-query` között az, hogy az `apt-file search` a nem telepített csomagok között is keres, míg a `dpkg-query` csak a telepített csomaghoz tartozó fájlokat tudja megjeleníteni.

Gyakorló feladatok

1. Milyen paranccsal telepíthetjük a `package.deb` nevű csomagot a `dpkg` használatával?

2. A `dpkg-query` segítségével keressük meg, hogy melyik csomag tartalmaz egy `7zr.1.gz` nevű fájlt!

3. El lehet távolítani az `unzip` nevű csomagot a rendszerből a `dpkg -r unzip` segítségével, ha a `file-roller` csomag függ tőle? Ha nem, akkor mi lenne a helyes módszer?

4. Az `apt-file` használatával hogyan lehet megtudni, hogy melyik csomag tartalmazza az `unrar` fájlt?

5. Az `apt-cache` használatának esetén mi a parancs a `gimp` csomag információinak megjelenítésére?

Gondolkodtató feladatok

1. Vegyünk egy repositoryt a `xenial` disztribúció Debian forráscsomagjaival, amelyet a `http://us.archive.ubuntu.com/ubuntu/` címen tárolunk, valamint egyet a `universe` komponens csomagjaival. Mi lenne a megfelelő sor, amelyet hozzá kellene adni az `/etc/apt/sources.list`-hez?

2. Egy program fordítása közben egy hibaüzenettel találkozunk, amely arról szól, hogy a `zzip-io.h` fejlécfájl nincs jelen a rendszerben. Hogyan tudjuk kideríteni, hogy melyik csomag biztosítja ezt a fájlt?

3. Hogyan lehet figyelmen kívül hagyni egy függőségi figyelmeztetést és eltávolítani egy csomagot a `dpkg` segítségével, még akkor is, ha vannak olyan csomagok a rendszerben, amelyek függnek tőle?

4. Hogyan kaphatunk több információt a `midori` nevű csomagról az `apt` segítségével?

5. A csomagok telepítése vagy frissítése előtt melyik parancsot kell használni annak biztosítására, hogy a csomagindex naprakész legyen?

Összefoglalás

Ebben a leckében megtanultuk:

- Hogyan használjuk a `dpkg`-t a csomagok telepítésére és eltávolítására.
- Hogyan listázzuk a telepített csomagokat és a csomagok tartalmát.
- Hogyan lehet egy telepített csomagot újrakonfigurálni.
- Mi az az `apt`, és hogyan lehet vele csomagokat telepíteni, upgrade-elni és eltávolítani.
- Hogyan használjuk az `apt-cache`-t a csomagok keresésére.
- Hogyan működik az `/etc/apt/sources.list` fájl.
- Hogyan használjuk az `apt-file`-t egy csomag tartalmának megjelenítésére, vagy hogyan találjuk meg, hogy melyik csomag tartalmaz egy adott fájlt.

A következő parancsokról volt szó:

`dpkg -i`
: Egy csomag vagy szóközökkel elválasztott csomaglista telepítése.

`dpkg -r`
: Egy csomag vagy szóközökkel elválasztott csomaglista eltávolítása.

`dpkg -I`
: Megvizsgál egy csomagot, részleteket szolgáltat a telepített szoftverekről és a szükséges függőségekről.

`dpkg --get-selections`
: Minden olyan csomagot felsorol, amelyet az `dpkg` telepített a rendszerre.

`dpkg -L`
: Megjelenít egy listát minden olyan fájlról, amelyet egy adott csomag telepít.

`dpkg-query`
: A megadott fájlnévvel ez a parancs kiírja a fájlt telepítő csomagot.

`dpkg-reconfigure`
: Ez a parancs újra lefuttatja a csomagok *post-install* szkriptjét, hogy a rendszergazda elvégezhesse a csomag telepítésének konfigurációs módosításait.

`apt-get update`

Ez a parancs frissíti a helyi csomagindexet, hogy az megfeleljen a `/etc/apt/` mappában található konfigurált repositorykban elérhető csomagoknak.

`apt-get install`

Ez a parancs letölti a csomagot egy távoli repositoryból és telepíti azt a függőségekkel együtt; használható egy már letöltött Debian csomag telepítésére is.

`apt-get remove`

Ez a parancs eltávolítja a megadott csomag(ok)at a rendszerből.

`apt-cache show`

A `dpkg -I` parancshoz hasonlóan ez a parancs is használható egy adott csomag részleteinek megjelenítésére.

`apt-cache search`

Ez a parancs a helyi APT gyorsítótár adatbázisában keres egy adott csomagot.

`apt-file update`

Ez a parancs frissíti a csomagok gyorsítótárát, hogy az `apt-file` parancs lekérdezhesse annak tartalmát.

`apt-file search`

Ez a parancs megkeresi, hogy egy adott fájl melyik csomagban található. A parancs a mintát tartalmazó összes csomag listáját adja vissza.

`apt-file list`

Ez a parancs egy csomag tartalmának listázására szolgál, akárcsak az `dpkg -L` parancs.

Válaszok a gyakorló feladatokra

1. Milyen paranccsal telepíthetjük a `package.deb` nevű csomagot a `dpkg` segítségével?

   Adjuk át a `-i` paramétert a `dpkg`-nak:

   ```
   # dpkg -i package.deb
   ```

2. A `dpkg-query` segítségével keressük meg, hogy melyik csomag tartalmaz egy `7zr.1.gz` nevű fájlt!

   Adjuk át a `-S` paramétert a `dpkg-query`-nek:

   ```
   # dpkg-query -S 7zr.1.gz
   ```

3. El lehet távolítani az `unzip` nevű csomagot a rendszerből a `dpkg -r unzip` segítségével, ha a `file-roller` csomag függ tőle? Ha nem, akkor mi lenne a helyes módszer?

   Nem. A `dpkg` nem oldja fel a függőségeket és nem engedi a csomag eltávolítását, ha egy másik csomag függ tőle. Ebben a példában először eltávolítjuk a `file-roller`-t (feltételezve, hogy semmi nem függ tőle) és utána az `unzip`-et, vagy mindkettőt egyszerre az alábbi módon:

   ```
   # dpkg -r unzip file-roller
   ```

4. Az `apt-file` használatával hogyan lehet megtudni, hogy melyik csomag tartalmazza az `unrar` fájlt?

   Használjuk a `search` paramétert az útvonallal (vagy fájlnévvel):

   ```
   # apt-file search /usr/bin/unrar
   ```

5. Az `apt-cache` használatának esetén mi a parancs a `gimp` csomag információinak megjelenítésére?

   Használjuk a `show` paramétert és a csomag nevét:

   ```
   # apt-cache show gimp
   ```

Válaszok a gondolkodtató feladatokra

1. Vegyünk egy repositoryt a `xenial` disztribúció Debian forráscsomagjaival, amelyet a `http://us.archive.ubuntu.com/ubuntu/` címen tárolunk, valamint egyet a `universe` komponens csomagjaival. Mi lenne a megfelelő sor, amelyet hozzá kellene adni az `/etc/apt/sources.list`-hez?

   A forráscsomagok `deb-src` típusúak, így a sornak a következőnek kell lennie:

   ```
   deb-src http://us.archive.ubuntu.com/ubuntu/ xenial universe
   ```

   Ezt a sort a `.list` fájlban is beilleszthetjük a `/etc/apt/sources.list.d/` állományba. A neve rajtunk múlik, de legyen leíró, például `xenial_sources.list`.

2. Egy program fordítása közben egy hibaüzenettel találkozunk, amely arról szól, hogy a `zzip-io.h` fejlécfájl nincs jelen a rendszerben. Hogyan tudjuk kideríteni, hogy melyik csomag biztosítja ezt a fájlt?

   Az `apt-file search` segítségével megkereshetjük, hogy melyik csomag tartalmaz olyan fájlt, amely nincs meg a rendszerben:

   ```
   # apt-file search zzip-io.h
   ```

3. Hogyan lehet figyelmen kívül hagyni egy függőségi figyelmeztetést és eltávolítani egy csomagot a `dpkg` segítségével, még akkor is, ha vannak olyan csomagok a rendszerben, amelyek függnek tőle?

   A `--force` paraméter használható, de ezt *soha* nem szabad megtenni, hacsak nem tudjuk pontosan, mit csinálunk, mivel nagy a kockázata annak, hogy a rendszer inkonzisztens vagy “broken” állapotban marad.

4. Hogyan kaphatunk több információt a `midori` nevű csomagról az `apt` segítségével?

   Használjuk az `apt-cache show`-t a csomag nevével:

   ```
   # apt-cache show midori
   ```

5. A csomagok telepítése vagy frissítése előtt melyik parancsot kell használni annak biztosítására, hogy a csomagindex naprakész legyen?

Az `apt-get update`-t kell használnunk. Ez letölti a legújabb csomagindexeket a `/etc/apt/sources.list` fájlban vagy az `/etc/apt/sources.list.d/` könyvtárban leírt repositorykból.

102.5 Az RPM és a YUM csomagkezelő használata

Hivatkozás az LPI célkitűzésre

LPIC-1 v5, Exam 101, Objective 102.5

Súlyozás

3

Kulcsfontosságú ismeretek

- Csomagok telepítése, újratelepítése, frissítése és eltávolítása RPM, YUM és Zypper segítségével.
- Információk beszerzése az RPM csomagokról, például a verzióról, állapotról, függőségekről, integritásról és aláírásokról.
- Annak meghatározása, hogy egy csomag milyen fájlokat biztosít, valamint megtalálni, hogy egy adott fájl melyik csomagból származik.
- A dnf ismerete.

A használt fájlok, kifejezések és segédprogramok listája

- `rpm`
- `rpm2cpio`
- `/etc/yum.conf`
- `/etc/yum.repos.d/`
- `yum`
- `zypper`

102.5 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	102 A Linux telepítése és a csomagkezelés
Fejezet:	102.5 Az RPM és a YUM csomagkezelő használata
Lecke:	1/1

Bevezetés

Réges-régen, amikor a Linux még gyerekcipőben járt, a szoftverek terjesztésének legelterjedtebb módja egy tömörített fájl (általában egy `.tar.gz` archívum) volt a forráskóddal, amelyet az ember saját maga csomagolt ki és fordított le.

A szoftverek mennyiségének és összetettségének növekedésével azonban egyértelművé vált az előre lefordított szoftverek terjesztésének szükségessége. Végülis nem mindenki rendelkezett az idő és a számítási teljesítmény tekintetében elegendő erőforrással ahhoz, hogy olyan nagy projekteket fordítson le, mint a Linux kernel vagy az X Server.

Hamarosan egyre nagyobb erőfeszítéseket tettek arra, hogy szabványosítsák a “package”-k (szoftvercsomagok) terjesztésének módját, és megszülettek az első csomagkezelők. Ezek az eszközök nagyban megkönnyítették a szoftverek telepítését, konfigurálását vagy eltávolítását a rendszerből.

Ezek egyike a Red Hat által kifejlesztett *RPM Package Manager* és a hozzá tartozó eszköz (`rpm`). Ma már nemcsak a Red Hat Enterprise Linuxon (RHEL), hanem annak leszármazottain, mint a

Fedora, CentOS és Oracle Linux, más disztribúciókon, mint az openSUSE, sőt más operációs rendszereken, például az IBM AIX-jén is széles körben használják.

A Red Hat kompatibilis disztribúciókban népszerű csomagkezelő eszközök a `yum` (YellowDog Updater Modified), a `dnf` (Dandified YUM) és a `zypper`, amelyek a csomagok telepítésének, karbantartásának és eltávolításának számos aspektusát leegyszerűsítik, így a csomagkezelés sokkal egyszerűbbé válik.

Ebben a leckében megtanuljuk, hogyan használjuk az `rpm`, `yum`, `dnf` és `zypper` csomagkezelőket a szoftverek beszerzésére, telepítésére, kezelésére és eltávolítására egy Linux rendszerben.

NOTE

Annak ellenére, hogy ugyanazt a csomagformátumot használják, a disztribúciók között belső különbségek vannak, így egy kifejezetten openSUSE-hez készült csomag nem biztos, hogy működik egy RHEL-rendszeren, és fordítva. Amikor csomagokat keresünk, mindig ellenőrizzük a kompatibilitást, és ha lehetséges, próbáljunk meg az adott disztribúcióra szabott csomagot találni.

Az RPM csomagkezelő (rpm)

Az RPM csomagkezelő (`rpm`) a szoftvercsomagok kezelésének alapvető eszköze a Red Hat-alapú (vagy a belőle származtatott) rendszereken.

Csomagok telepítése, frissítése és eltávolítása

A legalapvetőbb művelet egy csomag telepítése, amit a következő módon lehet elvégezni:

```
# rpm -i PACKAGENAME
```

Ahol a `PACKAGENAME` a telepíteni kívánt `.rpm` csomag neve.

Ha egy csomag korábbi verziója van a rendszeren, akkor a `-U` paraméter segítségével frissíthetünk egy újabb verzióra:

```
# rpm -U PACKAGENAME
```

Ha nincs telepítve a `PACKAGENAME` korábbi verziója, akkor egy új példány kerül telepítésre. Ha ezt el akarjuk kerülni, és *csak* egy *telepített* csomagot szeretnénk frissíteni, használjuk a `-F` opciót.

Mindkét művelethez hozzáadhatjuk a `-v` paramétert, hogy részletes (verbose) kimenetet kapjunk (több információ jelenik meg a telepítés során), és a `-h` paramétert, hogy a telepítés

előrehaladásának követéséhez vizuális segédeszközként kettőskeresztek (#) jelenjenek meg. Több paraméter kombinálható egybe, így az `rpm -i -v -h` ugyanaz, mint az `rpm -ivh`.

Egy telepített csomag eltávolításához adjuk meg az `-e` paramétert (mint a “erase”, azaz törlés) az `rpm`-nek, majd az eltávolítani kívánt csomag nevét:

```
# rpm -e wget
```

Ha egy telepített csomag függ az eltávolítandó csomagtól, hibaüzenetet kapunk:

```
# rpm -e unzip
error: Failed dependencies:
    /usr/bin/unzip is needed by (installed) file-roller-3.28.1-2.el7.x86_64
```

A művelet elvégzéséhez először el kell távolítanunk azokat a csomagokat, amelyek az eltávolítandó csomagtól függenek (a fenti példában a `file-roller`). Több csomag nevét is megadhatjuk az `rpm -e` parancsnak, ha egyszerre több csomagot szeretnénk eltávolítani.

A függőségek kezelése

Gyakran előfordul, hogy egy csomag rendeltetésszerű működése másoktól függ. Egy képszerkesztőnek például szüksége lehet könyvtárakra a JPEG-fájlok megnyitásához, vagy egy másik segédprogram felhasználói felületének szüksége lehet egy olyan widget eszköztárra, mint a Qt vagy a GTK.s

Az `rpm` ellenőrzi, hogy ezek a függőségek telepítve vannak-e a rendszerre, és ha nem, akkor nem telepíti a csomagot. Ebben az esetben az `rpm` felsorolja, hogy mi hiányzik. A függőségeket azonban magától *nem* tudja feloldani.

Az alábbi példában a felhasználó megpróbált telepíteni egy csomagot a GIMP képszerkesztőhöz, de néhány függőség hiányzott:

```
# rpm -i gimp-2.8.22-1.el7.x86_64.rpm
error: Failed dependencies:
    babl(x86-64) >= 0.1.10 is needed by gimp-2:2.8.22-1.el7.x86_64
    gegl(x86-64) >= 0.2.0 is needed by gimp-2:2.8.22-1.el7.x86_64
    gimp-libs(x86-64) = 2:2.8.22-1.el7 is needed by gimp-2:2.8.22-1.el7.x86_64
    libbabl-0.1.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
    libgegl-0.2.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
    libgimp-2.0.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
    libgimpbase-2.0.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
```

```
    libgimpcolor-2.0.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
    libgimpconfig-2.0.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
    libgimpmath-2.0.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
    libgimpmodule-2.0.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
    libgimpthumb-2.0.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
    libgimpui-2.0.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
    libgimpwidgets-2.0.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
    libmng.so.1()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
    libwmf-0.2.so.7()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
    libwmflite-0.2.so.7()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
```

A felhasználó feladata, hogy megkeresse a megfelelő függőségeket tartalmazó `.rpm` csomagokat és telepítse őket. Az olyan csomagkezelők, mint a `yum`, a `zypper` és a `dnf` rendelkeznek olyan eszközökkel, amelyek meg tudják mondani, hogy egy adott fájlt melyik csomag biztosít. Ezeket a lecke későbbi részében tárgyaljuk.

Telepített csomagok listázása

A rendszeren lévő összes telepített csomag listájának megtekintéséhez használjuk az `rpm -qa` parancsot (`query all`).

```
# rpm -qa
selinux-policy-3.13.1-229.el7.noarch
pciutils-libs-3.5.1-3.el7.x86_64
redhat-menus-12.0.2-8.el7.noarch
grubby-8.28-25.el7.x86_64
hunspell-en-0.20121024-6.el7.noarch
dejavu-fonts-common-2.33-6.el7.noarch
xorg-x11-drv-dummy-0.3.7-1.el7.1.x86_64
libevdev-1.5.6-1.el7.x86_64
[...]
```

Csomaginformációk megszerzése

Ha olyan információkat szeretnénk kapni egy *telepített* csomagról, mint például a verziószám, architektúra, telepítési dátum, a csomag készítője (packager), összefoglaló stb., használjuk az `rpm`-t a `-qi` (“query info”) paraméterekkel, amelyet a csomag neve követ. Például:

```
# rpm -qi unzip
Name        : unzip
Version     : 6.0
```

```
Release     : 19.el7
Architecture: x86_64
Install Date: Sun 25 Aug 2019 05:14:39 PM EDT
Group       : Applications/Archiving
Size        : 373986
License     : BSD
Signature   : RSA/SHA256, Wed 25 Apr 2018 07:50:02 AM EDT, Key ID 24c6a8a7f4a80eb5
Source RPM  : unzip-6.0-19.el7.src.rpm
Build Date  : Wed 11 Apr 2018 01:24:53 AM EDT
Build Host  : x86-01.bsys.centos.org
Relocations : (not relocatable)
Packager    : CentOS BuildSystem <http://bugs.centos.org>
Vendor      : CentOS
URL         : http://www.info-zip.org/UnZip.html
Summary     : A utility for unpacking zip files
Description :
The unzip utility is used to list, test, or extract files from a zip
archive. Zip archives are commonly found on MS-DOS systems. The zip
utility, included in the zip package, creates zip archives. Zip and
unzip are both compatible with archives created by PKWARE(R)'s PKZIP
for MS-DOS, but the programs' options and default behaviors do differ
in some respects.

Telepítsük az unzip csomagot, ha fájlokat kell listázni, tesztelni vagy kicsomagolni egy
zip-archívumból.
```

Ha egy *telepített* csomagban található fájlok listáját szeretnénk megkapni, használjuk a `-ql` paramétereket (“query list”), amelyet a csomag neve követ:

```
# rpm -ql unzip
/usr/bin/funzip
/usr/bin/unzip
/usr/bin/unzipsfx
/usr/bin/zipgrep
/usr/bin/zipinfo
/usr/share/doc/unzip-6.0
/usr/share/doc/unzip-6.0/BUGS
/usr/share/doc/unzip-6.0/LICENSE
/usr/share/doc/unzip-6.0/README
/usr/share/man/man1/funzip.1.gz
/usr/share/man/man1/unzip.1.gz
/usr/share/man/man1/unzipsfx.1.gz
/usr/share/man/man1/zipgrep.1.gz
```

```
/usr/share/man/man1/zipinfo.1.gz
```

Ha olyan csomagról szeretnénk információt vagy fájllistát kapni, amely *nem* lett telepítve, csak adjuk hozzá a fenti parancsokhoz a `-p` paramétert, majd az RPM fájl nevét (`FILENAME`). Így az `rpm -qi PACKAGENAME`-ből `rpm -qip FILENAME` lesz, az `rpm -ql PACKAGENAME`-ből pedig `rpm -qlp FILENAME`, ahogy az alábbiakban látható.

```
# rpm -qip atom.x86_64.rpm
Name        : atom
Version     : 1.40.0
Release     : 0.1
Architecture: x86_64
Install Date: (not installed)
Group       : Unspecified
Size        : 570783704
License     : MIT
Signature   : (none)
Source RPM  : atom-1.40.0-0.1.src.rpm
Build Date  : sex 09 ago 2019 12:36:31 -03
Build Host  : b01bbeaf3a88
Relocations : /usr
URL         : https://atom.io/
Summary     : A hackable text editor for the 21st Century.
Description :
A hackable text editor for the 21st Century.
```

```
# rpm -qlp atom.x86_64.rpm
/usr/bin/apm
/usr/bin/atom
/usr/share/applications/atom.desktop
/usr/share/atom
/usr/share/atom/LICENSE
/usr/share/atom/LICENSES.chromium.html
/usr/share/atom/atom
/usr/share/atom/atom.png
/usr/share/atom/blink_image_resources_200_percent.pak
/usr/share/atom/content_resources_200_percent.pak
/usr/share/atom/content_shell.pak

(a listázás folytatódik)
```

Annak megállapítása, hogy melyik csomagban található egy adott fájl

Ha meg akarjuk tudni, hogy egy fájl melyik telepített csomag tulajdonában van, használjuk a `-qf` ("query file") paramétert, amelyet a fájl teljes elérési útvonala követ:

```
# rpm -qf /usr/bin/unzip
unzip-6.0-19.el7.x86_64
```

A fenti példában a `/usr/bin/unzip` fájl az `unzip-6.0-19.el7.x86_64` csomaghoz tartozik.

YellowDog Updater Modified (YUM)

A `yum` eredetileg *Yellow Dog Updater* (YUP) néven készült, a Yellow Dog Linux disztribúció csomagkezelője. Idővel más RPM-alapú rendszerek, például a Fedora, CentOS, Red Hat Enterprise Linux és Oracle Linux csomagjainak kezelésére is továbbfejlesztették.

Funkcionálisan hasonló a Debian-alapú rendszerek `apt` segédprogramjához, képes csomagok keresésére, telepítésére, frissítésére és eltávolítására, valamint a függőségek automatikus kezelésére. A `yum` használható egyetlen csomag telepítésére, vagy egy egész rendszer egyszerre történő frissítésére.

Csomagok keresése

Ahhoz, hogy telepíteni tudjunk egy csomagot, tudnunk kell a nevét. Ehhez kereshetünk a `yum search PATTERN` paranccsal, ahol a `PATTERN` a keresett csomag neve. Az eredmény azon csomagok listája, amelyek neve vagy összefoglalója tartalmazza a megadott keresési mintát. Ha például szükségünk van egy segédprogramra a 7Zip tömörített fájlok (`.7z` kiterjesztéssel) kezelésére, akkor használhatjuk a következőt:

```
# yum search 7zip
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.ufscar.br
 * epel: mirror.globo.com
 * extras: mirror.ufscar.br
 * updates: mirror.ufscar.br
============================ N/S matchyutr54ed: 7zip ============================
p7zip-plugins.x86_64 : Additional plugins for p7zip
p7zip.x86_64 : Very high compression ratio file archiver
p7zip-doc.noarch : Manual documentation and contrib directory
p7zip-gui.x86_64 : 7zG - 7-Zip GUI version
```



```
  Csak a nevek és az összefoglalók egyezését nézi, minden máshoz használjuk a "search all"-
t.
```

Csomagok telepítése, frissítése és eltávolítása

Egy csomag telepítéséhez a `yum` használatával használjuk a `yum install PACKAGENAME` parancsot, ahol a `PACKAGENAME` a csomag neve. A `yum` lekérdezi a csomagot és a hozzá tartozó függőségeket egy online repositoryból és mindent telepít a rendszerre.

```
# yum install p7zip
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.ufscar.br
 * epel: mirror.globo.com
 * extras: mirror.ufscar.br
 * updates: mirror.ufscar.br
Resolving Dependencies
--> Running transaction check
---> Package p7zip.x86_64 0:16.02-10.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package        Arch            Version               Repository     Size
================================================================================
Installing:
 p7zip          x86_64          16.02-10.el7          epel          604 k

Transaction Summary
================================================================================
Install  1 Package

Total download size: 604 k
Installed size: 1.7 M
Is this ok [y/d/N]:
```

Egy telepített csomag frissítéséhez használjuk a `yum update PACKAGENAME` parancsot, ahol a `PACKAGENAME` a frissíteni kívánt csomag neve. Például:

```
# yum update wget
```

```
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.ufscar.br
 * epel: mirror.globo.com
 * extras: mirror.ufscar.br
 * updates: mirror.ufscar.br
Resolving Dependencies
--> Running transaction check
---> Package wget.x86_64 0:1.14-18.el7 will be updated
---> Package wget.x86_64 0:1.14-18.el7_6.1 will be an update
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package     Arch          Version                   Repository      Size
================================================================================
Updating:
 wget        x86_64        1.14-18.el7_6.1           updates        547 k

Transaction Summary
================================================================================
Upgrade  1 Package

Total download size: 547 k
Is this ok [y/d/N]:
```

Ha nem adunk meg csomagnevet, akkor minden olyan csomagot frissíthetünk, amelyhez van elérhető frissítés.

Ha ellenőrizni szeretnénk, hogy elérhető-e frissítés egy adott csomaghoz, használjuk a `yum check-update PACKAGENAME` parancsot. A korábbiakhoz hasonlóan, ha elhagyjuk a csomag nevét, a `yum` a rendszerben lévő összes telepített csomag frissítését ellenőrizni fogja.

Egy telepített csomag eltávolításához használjuk a `yum remove PACKAGENAME` parancsot, ahol a `PACKAGENAME` az eltávolítani kívánt csomag neve.

Annak megtalálása, hogy melyik csomag biztosít egy adott fájlt

Az előző példában bemutattunk egy kísérletet a `gimp` képszerkesztő telepítésére, ami nem teljesült függőségek miatt nem sikerült. Az `rpm` azonban megmutatja, hogy mely fájlok hiányoznak, de nem sorolja fel az azokat biztosító csomagok nevét.

Például az egyik hiányzó függőség a `libgimpui-2.0.so.0` volt. Ha meg akarjuk nézni, hogy melyik csomag biztosítja, használhatjuk a `yum whatprovides` parancsot a keresett fájl nevével:

```
# yum whatprovides libgimpui-2.0.so.0
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.ufscar.br
 * epel: mirror.globo.com
 * extras: mirror.ufscar.br
 * updates: mirror.ufscar.br
2:gimp-libs-2.8.22-1.el7.i686 : GIMP libraries
Repo        : base
Matched from:
Provides    : libgimpui-2.0.so.0
```

A válasz a `gimp-libs-2.8.22-1.el7.i686`. Ezután a csomagot a `yum install gimp-libs` paranccsal telepíthetjük.

Ez a rendszerben már meglévő fájlok esetében is működik. Ha például tudni szeretnénk, hogy honnan származik az `/etc/hosts` fájl, használhatjuk a következőt:

```
# yum whatprovides /etc/hosts
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.ufscar.br
 * epel: mirror.globo.com
 * extras: mirror.ufscar.br
 * updates: mirror.ufscar.br
setup-2.8.71-10.el7.noarch : A set of system configuration and setup files
Repo        : base
Matched from:
Filename    : /etc/hosts
```

A válasz: `setup-2.8.71-10.el7.noarch`.

Információk megszerzése egy csomagról

Ha információt szeretnénk kapni egy csomagról, például a verziójáról, architektúrájáról, leírásáról, méretéről és egyebekről, használjuk a `yum info PACKAGENAME` parancsot, ahol a `PACKAGENAME` a csomag neve, amelyről információt szeretnénk kapni:

```
# yum info firefox
Last metadata expiration check: 0:24:16 ago on Sat 21 Sep 2019 02:39:43 PM -03.
Installed Packages
Name         : firefox
Version      : 69.0.1
Release      : 3.fc30
Architecture : x86_64
Size         : 268 M
Source       : firefox-69.0.1-3.fc30.src.rpm
Repository   : @System
From repo    : updates
Summary      : Mozilla Firefox Web browser
URL          : https://www.mozilla.org/firefox/
License      : MPLv1.1 or GPLv2+ or LGPLv2+
Description  : Mozilla Firefox is an open-source web browser, designed
             : for standards compliance, performance and portability.
```

A repositoryk menedzselése

A `yum` esetében a repositoryk (“repos”) az `/etc/yum.repos.d/` mappában találhatók. Minden egyes repositoryt egy `.repo` fájl képvisel, például a `CentOS-Base.repo`.

További, extra repositorykat a felhasználó adhat hozzá egy `.repo` fájl hozzáadásával a fent említett mappában, vagy az `/etc/yum.conf` fájl végén. A repositoryk hozzáadásának vagy kezelésének ajánlott módja azonban a `yum-config-manager` eszköz használata.

Repoistory hozzáadásához használjuk a `--add-repo` paramétert, amelyet a `.repo` fájl URL-je követ.

```
# yum-config-manager --add-repo https://rpms.remirepo.net/enterprise/remi.repo
Loaded plugins: fastestmirror, langpacks
adding repo from: https://rpms.remirepo.net/enterprise/remi.repo
grabbing file https://rpms.remirepo.net/enterprise/remi.repo to /etc/yum.repos.d/remi.repo
repo saved to /etc/yum.repos.d/remi.repo
```

Az összes elérhető repository listájának lekérdezéséhez használjuk a `yum repolist all` parancsot. Egy ehhez hasonló kimenetet fogunk kapni:

```
# yum repolist all
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
```

```
 * base: mirror.ufscar.br
 * epel: mirror.globo.com
 * extras: mirror.ufscar.br
 * updates: mirror.ufscar.br
repo id                      repo name                    status
updates/7/x86_64             CentOS-7 - Updates           enabled:  2,500
updates-source/7             CentOS-7 - Updates Sources   disabled
```

A `disabled` repositoryk figyelmen kívül maradnak a szoftver telepítésekor vagy frissítésekor. Egy repository engedélyezéséhez vagy letiltásához használjauk a `yum-config-manager` segédprogramot, majd a repository azonosítóját.

A fenti kimeneten a repository azonosítója minden sor első oszlopában (`repo id`) szerepel. Csak az első `/` előtti részt használjuk, így a `CentOS-7 - Updates` repo azonosítója `updates`, és nem `updates/7/x86_64`.

```
# yum-config-manager --disable updates
```

A fenti parancs kikapcsolja az `updates` repot. Újra bekapcsolásához használjuk a következőt:

```
# yum-config-manager --enable updates
```

NOTE

A Yum a letöltött csomagokat és a hozzájuk tartozó metaadatokat egy gyorsítótárban (általában `/var/cache/yum`) tárolja. Ahogy a rendszer frissül és új csomagok települnek, ez a gyorsítótár elég nagyra nőhet. A gyorsítótár kitakarításához és a lemezterület visszanyeréséhez használhatjuk a `yum clean` parancsot és azt, hogy mit kell kitakarítani. A leghasznosabb paraméterek a `packages` (`yum clean packages`) a letöltött csomagok törléséhez és a `metadata` (`yum clean metadata`) a kapcsolódó metaadatok törléséhez. További információkért lsd. a `yum` kézikönyv oldalát (`man yum` parancs).

DNF

A `dnf` a Fedora által használt csomagkezelő eszköz, és a `yum` egy forkja. Mint ilyen, sok parancsa és paramétere hasonló. Ez a szakasz csak egy gyors áttekintést ad a `dnf`-ről.

Csomagok keresése

`dnf search PATTERN`, ahol a `PATTERN` az, amit keresünk. Például a `dnf search unzip` megjelenít minden olyan csomagot, amik tartalmazzák az `unzip`-et a nevükben vagy a

leírásukban.

Információk megszerzése egy csomagról

`dnf info PACKAGENAME`

Csomagok telepítése

`dnf install PACKAGENAME`, ahol a `PACKAGENAME` a telepíteni kívánt csomag neve. A nevet kereséssel megtalálhatjuk.

Csomagok eltávolítása

`dnf remove PACKAGENAME`

Csomagok frissítése

`dnf upgrade PACKAGENAME` egy csomag frissítéséhez. A csomagnév kihagyásával az összes csomagot frissíthetjük.

Annak megtalálása, hogy melyik csomag biztosít egy adott fájlt

`dnf provides FILENAME`

A rendszerben telepített összes csomag listájának lekérdezése

`dnf list --installed`

Egy csomag tartalmának kilistázása

`dnf repoquery -l PACKAGENAME`

NOTE A `dnf` beépített súgórendszerrel rendelkezik, amely minden egyes parancshoz további információkat (például extra paramétereket) mutat. Használatához írjuk be a `dnf help`-et és a parancsot, például `dnf help install`.

Repositoryk menedzselése

A `yum`-hoz és `zypper`-hez hasonlóan a `dnf` is repositorykkal (repok) dolgozik. Minden disztribúció rendelkezik az alapértelmezett repositoryk listájával, és a rendszergazdák szükség szerint hozzáadhatnak vagy eltávolíthatnak repositorykat.

Az összes elérhető repository listájának lekérdezéséhez használjuk a `dnf repolist` parancsot. Csak az engedélyezett repositoryk listázásához adjuk hozzá az `--enabled` opciót, csak a letiltott tárolók listázásához adjuk hozzá a `--disabled` opciót.

```
# dnf repolist
Last metadata expiration check: 0:20:09 ago on Sat 21 Sep 2019 02:39:43 PM -03.
```

```
repo id                    repo name                                    status
*fedora                    Fedora 30 - x86_64                           56,582
*fedora-modular            Fedora Modular 30 - x86_64                      135
*updates                   Fedora 30 - x86_64 - Updates                 12,774
*updates-modular           Fedora Modular 30 - x86_64 - Updates             145
```

Egy repository hozzáadásához használjuk a `dnf config-manager --add_repo URL` parancsot, ahol az `URL` a repository teljes URL címe. A repository engedélyezéséhez használjuk a `dnf config-manager --set-enabled REPO_ID` parancsot.

Hasonlóképpen, egy repository letiltásához használjuk a `dnf config-manager --set-disabled REPO_ID` parancsot. Mindkét esetben a `REPO_ID` a repository egyedi azonosítója, amelyet a `dnf repolist` segítségével kaphatunk meg. A hozzáadott repositoryk alapértelmezés szerint engedélyezve vannak.

A repositoryk a `.repo` fájlokban tárolódnak az `/etc/yum.repos.d/` mappában, pontosan ugyanazzal a szintaxissal, mint a `yum` esetében.

Zypper

A `zypper` a SUSE Linux és az OpenSUSE rendszeren használt csomagkezelő eszköz. Funkcióit tekintve hasonló az `apt` és a `yum` rendszerekhez, képes telepíteni, frissíteni és eltávolítani csomagokat a rendszerből, automatikus függőség-feloldással.

A csomagindex frissítése

Más csomagkezelő eszközökhöz hasonlóan a `zypper` is csomagokat és metaadatokat tartalmazó repositorykkal dolgozik. Ezeket a metaadatokat időről időre frissíteni kell, hogy a segédprogram tudjon a legújabb elérhető csomagokról. A frissítéshez egyszerűen írjuk be a következőt:

```
# zypper refresh
Repository 'Non-OSS Repository' is up to date.
Repository 'Main Repository' is up to date.
Repository 'Main Update Repository' is up to date.
Repository 'Update Repository (Non-Oss)' is up to date.
All repositories have been refreshed.
```

A `zypper` rendelkezik egy automatikus frissítési funkcióval, amely repositorynként engedélyezhető, ami azt jelenti, hogy egyes repositoryk automatikusan frissülnek egy lekérdezés vagy csomagtelepítés előtt, míg másokat kézzel kell frissíteni. Ennek a funkciónak a vezérlését hamarosan megtanuljuk.

Csomagok keresése

Egy csomag kereséséhez használjuk a `search` (vagy `se`) operátort a csomag nevével:

```
# zypper se gnumeric
Loading repository data...
Reading installed packages...

S | Name           | Summary                            | Type
--+----------------+------------------------------------+--------
  | gnumeric       | Spreadsheet Application            | package
  | gnumeric-devel | Spreadsheet Application            | package
  | gnumeric-doc   | Documentation files for Gnumeric   | package
  | gnumeric-lang  | Translations for package gnumeric | package
```

A keresés operátorral a rendszerben lévő összes telepített csomag listáját is lekérdezhetjük. Ehhez használjuk a `-i` paramétert csomagnév nélkül, mint `zypper se -i`.

Ha látni szeretnénk, hogy egy adott csomag telepítve van-e, adjuk hozzá a csomag nevét a fenti parancshoz. Például a következő parancs a telepített csomagok között keres olyanokat, amelyek nevében szerepel a “firefox”:

```
# zypper se -i firefox
Loading repository data...
Reading installed packages...

S | Name                                | Summary                 | Type
--+-------------------------------------+-------------------------+--------
i | MozillaFirefox                      | Mozilla Firefox Web B-> | package
i | MozillaFirefox-branding-openSUSE    | openSUSE branding of -> | package
i | MozillaFirefox-translations-common | Common translations f-> | package
```

Ha csak a *nem telepített* csomagok között szeretnénk keresni, adjuk hozzá a `-u` paramétert a `se` operátorhoz.

Csomagok telepítése, frissítése és eltávolítása

Egy szoftvercsomag telepítéséhez használjuk az `install` (vagy `in`) operátort, amelyet a csomag neve követ. Például így:

```
# zypper in unrar
```



```
zypper in unrar
Loading repository data...
Reading installed packages...
Resolving package dependencies...

The following NEW package is going to be installed:
  unrar

1 new package to install.
Overall download size: 141.2 KiB. Already cached: 0 B. After the operation, additional 301.6
KiB will be used.
Continue? [y/n/v/...? shows all options] (y): y
Retrieving package unrar-5.7.5-lp151.1.1.x86_64
                                    (1/1), 141.2 KiB (301.6 KiB unpacked)
Retrieving: unrar-5.7.5-lp151.1.1.x86_64.rpm .....................[done]
Checking for file conflicts: .....................................[done]
(1/1) Installing: unrar-5.7.5-lp151.1.1.x86_64 ...................[done]
```

A `zypper` arra is használható, hogy egy RPM csomagot telepítsen a lemezre, miközben megpróbálja a függőségeit a repositorykból származó csomagok segítségével kielégíteni. Ehhez a csomag neve helyett csak a csomag teljes elérési útját kell megadni, például `zypper in /home/john/newpackage.rpm`.

A rendszerre telepített csomagok frissítéséhez használjuk a `zypper update` parancsot. A telepítési folyamathoz hasonlóan ez is megjeleníti a telepítendő/frissítendő csomagok listáját, mielőtt megkérdezné, hogy folytatni akarjuk-e a telepítést.

Ha csak az elérhető frissítéseket szeretnénk listázni, anélkül, hogy bármit is telepítenénk, használhatjuk a `zypper list-updates` parancsot.

Egy csomag eltávolításához használjuk a `remove` (vagy `rm`) operátort, amelyet a csomag neve követ:

```
# zypper rm unrar
Loading repository data...
Reading installed packages...
Resolving package dependencies...

The following package is going to be REMOVED:
  unrar

1 package to remove.
After the operation, 301.6 KiB will be freed.
```

```
Continue? [y/n/v/...? shows all options] (y): y
(1/1) Removing unrar-5.7.5-lp151.1.1.x86_64 .....................[done]
```

Ne feledjük, hogy egy csomag eltávolítása minden más, tőle függő csomagot is eltávolít. Például:

```
# zypper rm libgimp-2_0-0
Loading repository data...
Warning: No repositories defined. Operating only with the installed resolvables. Nothing can
be installed.
Reading installed packages...
Resolving package dependencies...

The following 6 packages are going to be REMOVED:
  gimp gimp-help gimp-lang gimp-plugins-python libgimp-2_0-0
  libgimpui-2_0-0

6 packages to remove.
After the operation, 98.0 MiB will be freed.
Continue? [y/n/v/...? shows all options] (y):
```

Annak megtalálása, hogy melyik csomag tartalmaz egy adott fájlt

Ha meg szeretnénk nézni, hogy mely csomagok tartalmaznak egy adott fájlt, használjuk a keresés operátort, amelyet a `--provides` paraméter és a fájl neve (vagy a teljes elérési útvonal) követ. Ha például tudni akarjuk, hogy mely csomagok tartalmazzák a `/usr/lib64/'-ben található` `libgimpmodule-2.0.so.0` fájlt, akkor a következőt használjuk:

```
# zypper se --provides /usr/lib64/libgimpmodule-2.0.so.0
Loading repository data...
Reading installed packages...

S | Name          | Summary                                          | Type
--+---------------+--------------------------------------------------+--------
i | libgimp-2_0-0 | The GNU Image Manipulation Program - Libra-> | package
```

Csomaginformációk megszerzése

A csomaghoz tartozó metaadatok megtekintéséhez használjuk az `info` operátort, amelyet a csomag neve követ. Ez megadja az eredeti repositoryt, a csomag nevét, verzióját, architektúráját, gyártóját, telepített méretét, azt, hogy telepítve van-e vagy sem, állapotát (ha naprakész), a

forráscsomagot és egy leírást.

```
# zypper info gimp
Loading repository data...
Reading installed packages...

Information for package gimp:
 ----------------------------
Repository     : Main Repository
Name           : gimp
Version        : 2.8.22-lp151.4.6
Arch           : x86_64
Vendor         : openSUSE
Installed Size : 29.1 MiB
Installed      : Yes (automatically)
Status         : up-to-date
Source package : gimp-2.8.22-lp151.4.6.src
Summary        : The GNU Image Manipulation Program
Description    :
    The GIMP is an image composition and editing program, which can be
    used for creating logos and other graphics for Web pages. The GIMP
    offers many tools and filters, and provides a large image
    manipulation toolbox, including channel operations and layers,
    effects, subpixel imaging and antialiasing, and conversions, together
    with multilevel undo. The GIMP offers a scripting facility, but many
    of the included scripts rely on fonts that we cannot distribute.
```

Repositoryk menedzselése

A `zypper` használható a repositoryk menedzselésére is. A rendszerben jelenleg regisztrált összes repository listájának megtekintéséhez használjuk a `zypper repos` parancsot:

```
# zypper repos
Repository priorities are without effect. All enabled repositories share the same priority.

#  | Alias                    | Name                                | Enabled | GPG Check |
Refresh
---+--------------------------+-------------------------------------+---------+-----------+
--------
 1 | openSUSE-Leap-15.1-1     | openSUSE-Leap-15.1-1                | No      | ----      |
----
 2 | repo-debug               | Debug Repository                    | No      | ----      |
```

```
----
 3 | repo-debug-non-oss        | Debug Repository (Non-OSS)         | No      | ----      |
----
 4 | repo-debug-update         | Update Repository (Debug)          | No      | ----      |
----
 5 | repo-debug-update-non-oss | Update Repository (Debug, Non-OSS) | No      | ----      |
----
 6 | repo-non-oss              | Non-OSS Repository                 | Yes     | (r ) Yes  |
Yes
 7 | repo-oss                  | Main Repository                    | Yes     | (r ) Yes  |
Yes
 8 | repo-source               | Source Repository                  | No      | ----      |
----
 9 | repo-source-non-oss       | Source Repository (Non-OSS)        | No      | ----      |
----
10 | repo-update               | Main Update Repository             | Yes     | (r ) Yes  |
Yes
11 | repo-update-non-oss       | Update Repository (Non-Oss)        | Yes     | (r ) Yes  |
Yes
```

Az `Enabled` oszlopban láthatjuk, hogy egyes tárolók engedélyezve vannak, míg mások nem. Ezt megváltoztathatjuk a `modifyrepo` operátorral, amelyet az `-e` (enable - engedélyezés) vagy `-d` (disable - letiltás) paraméter és a repository alias (a fenti kimenet második oszlopa) követ.

```
# zypper modifyrepo -d repo-non-oss
Repository 'repo-non-oss' has been successfully disabled.

# zypper modifyrepo -e repo-non-oss
Repository 'repo-non-oss' has been successfully enabled.
```

Korábban említettük, hogy a `zypper` rendelkezik egy *auto refresh* képességgel, amely repositorynként engedélyezhető. Ha engedélyezve van, ez a flag a `zypper`-t egy frissítési művelet futtatására készteti (ugyanúgy, mint a `zypper refresh` futtatása), mielőtt a megadott repoval dolgozna. Ez a `modifyrepo` operátor `-f` és `-F` paramétereivel szabályozható:

```
# zypper modifyrepo -F repo-non-oss
Autorefresh has been disabled for repository 'repo-non-oss'.

# zypper modifyrepo -f repo-non-oss
Autorefresh has been enabled for repository 'repo-non-oss'.
```

Repositoryk hozzáadása és eltávolítása

Új repository `zypper`-hez való hozzáadásához használjuk az `addrepo` operátort, amelyet a repository URL-címe és neve követ, az alábbiak szerint:

```
# zypper addrepo http://packman.inode.at/suse/openSUSE_Leap_15.1/ packman
Adding repository 'packman' .....................................[done]
Repository 'packman' successfully added

URI         : http://packman.inode.at/suse/openSUSE_Leap_15.1/
Enabled     : Yes
GPG Check   : Yes
Autorefresh : No
Priority    : 99 (default priority)

Repository priorities are without effect. All enabled repositories share the same priority.
```

Repository hozzáadásakor az automatikus frissítéseket az `-f` paraméterrel engedélyezhetjük. A hozzáadott repositoryk alapértelmezés szerint engedélyezve vannak, de hozzáadáskor letilthatjuk a `-d` paraméterrel.

Egy repository eltávolításához használjuk a `removerepo` operátort, amelyet a repository neve (Alias) követ. A fenti példában hozzáadott repository eltávolításához a parancs a következő:

```
# zypper removerepo packman
Removing repository 'packman' ...................................[done]
Repository 'packman' has been removed.
```

Gyakorló feladatok

1. Hogyan telepíthetjük az `rpm` segítségével egy Red Hat Enterprise Linux rendszeren a `file-roller-3.28.1-2.el7.x86_64.rpm` csomagot, amely a telepítés során megjelenít egy progress bart?

2. Az `rpm` segítségével derítsük ki, hogy melyik csomag tartalmazza a `/etc/redhat-release` fájlt!

3. Hogyan használnánk a `yum`-ot a rendszerben lévő összes csomag frissítésének ellenőrzésére?

4. A `zypper` használatával hogyan tudnánk letiltani a repo-extras nevű repositoryt?

5. Ha van egy `.repo` fájl, amely leír egy új repositoryt, hova kell ezt a fájlt tenni, hogy a DNF felismerje?

Gondolkodtató feladatok

1. Hogyan tudnánk a `zypper` segítségével kideríteni, hogy melyik csomagban van a `/usr/sbin/swapon` fájl?

2. Hogyan kaphatunk egy listát a rendszerben lévő összes telepített csomagról a `dnf` segítségével?

3. A `dnf` használatával milyen paranccsal adhatunk hozzá egy `https://www.example.url/home:reponame.repo` címen található repositoryt a rendszerhez?

4. Hogyan lehet a `zypper` segítségével ellenőrizni, hogy az `unzip` csomag telepítve van-e?

5. A `yum` segítségével derítsük ki, hogy melyik csomagban van a `/bin/wget` fájl!

Összefoglalás

Ebben a leckében megtanultuk:

- Hogyan használjuk az `rpm`-et csomagok telepítésére, frissítésére és eltávolítására.
- Hogyan használjuk a `yum`, a `zypper` és a `dnf` eszközöket.
- Hogyan szerezhetünk információkat egy csomagról.
- Hogyan kaphatunk listát a csomagok tartalmáról.
- Hogyan lehet megtudni, hogy egy fájl melyik csomagból származik.
- Hogyan lehet repositorykat listázni, hozzáadni, eltávolítani, engedélyezni vagy letiltani.

A következő parancsokról volt szó:

- `rpm`
- `yum`
- `dnf`
- `zypper`

Válaszok a gyakorló feladatokra

1. Hogyan telepíthetjük az `rpm` segítségével egy Red Hat Enterprise Linux rendszeren a `file-roller-3.28.1-2.el7.x86_64.rpm` csomagot, amely a telepítés során megjelenít egy progress bart?

   Használjuk a `-i` paramétert a csomag telepítéséhez és a `-h` kapcsolót a "kettőskeresztek (hash mark)" megjelenítéséhez a telepítés során. Tehát a válasz: `rpm -ih file-roller-3.28.1-2.el7.x86_64.rpm`.

2. Az `rpm` segítségével derítsük ki, hogy melyik csomag tartalmazza a `/etc/redhat-release` fájlt!

   Egy fájlról kérdezünk le információkat, ezért használjuk a `-qf` paramétert: `rpm -qf /etc/redhat-release`.

3. Hogyan használnánk a `yum`-ot a rendszerben lévő összes csomag frissítésének ellenőrzésére?

   Használjuk a `check-update` műveletet csomagnév *nélkül*: `yum check-update`.

4. A `zypper` használatával hogyan tudnánk letiltani a repo-extras nevű repositoryt?

   Használjuk a `modifyrepo` műveletet a repo paramétereinek megváltoztatásához és a `-d` paramétert a letiltásához: `zypper modifyrepo -d repo-extras`.

5. Ha van egy `.repo` fájl, amely leír egy új repositoryt, hova kell ezt a fájlt tenni, hogy a DNF felismerje?

   A DNF `.repo` fájljait ugyanarra a helyre kell tenni, ahova a YUM is teszi, az `/etc/yum.repos.d/`-be.

Válaszok a gondolkodtató feladatokra

1. Hogyan tudnánk a `zypper` segítségével kideríteni, hogy melyik csomagban van a `/usr/sbin/swapon` fájl?

   Használjuk a `se` (search - keresés) operátort és a `--provides` paramétert: `zypper se --provides /usr/sbin/swapon`.

2. Hogyan kaphatunk egy listát a rendszerben lévő összes telepített csomagról a `dnf` segítségével?

   Használjuk a `list` operátort, majd az `--installed` paramétert: `dnf list --installed`.

3. A `dnf` használatával milyen paranccsal adhatunk hozzá egy `https://www.example.url/home:reponame.repo` címen található repositoryt a rendszerhez?

   A repositorykkal való munka "konfigurációváltás", ezért használjuk a `config-manager` és az `--add_repo` paramétert: `dnf config-manager --add_repo https://www.example.url/home:reponame.repo`.

4. Hogyan lehet a `zypper` segítségével ellenőrizni, hogy az `unzip` csomag telepítve van-e?

   Végre kell hajtanunk egy keresést (`se`) a telepített (`-i`) csomagokon: `zypper se -i unzip`.

5. A `yum` segítségével derítsük ki, hogy melyik csomagban van a `/bin/wget` fájl!

   Ahhoz, hogy kitaláljuk, mi biztosít egy fájlt, használjuk a `whatprovides`-t és a fájlnevet: `yum whatprovides /bin/wget`.

102.6 A Linux, mint virtualization guest

Hivatkozás az LPI célkitűzésre

LPIC-1, Exam 101, Objective 102.6

Súlyozás

1

Kulcsfontosságú ismeretek

- A virtuális gépek és konténerek általános koncepciójának megértése
- Az IaaS-felhőben található virtuális gépek közös elemeinek, például a számítási példányoknak, a blokkos tárolásnak és a hálózatnak a megértése.
- A Linux-rendszerek egyedi tulajdonságainak megértése, amelyeket meg kell változtatni, ha egy rendszert klónoznak vagy sablonként használnak.
- Annak megértése, hogy hogyan használnak rendszerképeket virtuális gépek, felhőpéldányok és konténerek telepítéséhez.
- A Linuxot virtualizációs termékkel integráló Linux-bővítmények megértése
- A cloud-init ismerete

A használt fájlok, kifejezések és segédprogramok listája

- Virtual machine
- Linux container
- Application container
- Guest drivers
- SSH host keys
- D-Bus machine id

102.6 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	102 A Linux telepítése és a csomagkezelés
Fejezet:	102.6 A Linux, mint virtualization guest (virtualizációs vendég)
Lecke:	1/1

Bevezetés

A Linux egyik nagy erőssége a sokoldalúsága. Ennek a sokoldalúságnak az egyik aspektusa az a képesség, hogy a Linuxot más operációs rendszerek vagy egyedi alkalmazások teljesen elszigetelt és biztonságos környezetben történő elhelyezésére használhatjuk. Ez a lecke a virtualizáció és a konténertechnológiák (container technologies) fogalmaira összpontosít, valamint néhány technikai részletre, amelyeket figyelembe kell venni, amikor virtuális gépet telepítünk egy felhőplatformon.

A virtualizáció áttekintése

A virtualizáció egy olyan technológia, amely lehetővé teszi, hogy egy *hypervisor*-nak nevezett szoftverplatformon olyan folyamatokat futtassunk, amelyek egy teljesen emulált számítógépes rendszert tartalmaznak. A hypervisor felelős az egyes virtuális gépek által használható fizikai hardver erőforrásainak kezeléséért. Ezeket a virtuális gépeket a hypervisor *vendégeinek* (guests) nevezzük. Egy virtuális gép a fizikai számítógép számos aspektusát emulálja szoftveresen, például a rendszer BIOS-át és a merevlemez vezérlőit. A virtuális gépek gyakran használnak egyedi fájlként tárolt merevlemez image-ket, és a hypervisor szoftveren keresztül hozzáférnek a

gazdagép (host machine) RAM-jához és CPU-jához. A hypervisor szétválasztja a vendégeknek a gazdarendszer hardveres erőforrásaihoz való hozzáférését, így lehetővé teszi, hogy egyetlen gazdarendszeren több operációs rendszer is fusson.

A Linuxhoz általában használt hypervisorok a következők:

Xen

A Xen egy nyílt forráskódú, Type-1 hypervisor, ami azt jelenti, hogy működéséhez nem támaszkodik egy alapul szolgáló operációs rendszerre. Az ilyen típusú hypervisort *bare-metal hypervisornak* nevezik, mivel a számítógép közvetlenül a hypervisorba tud bootolni.

KVM

A Kernel Virtual Machine egy Linux kernelmodul a virtualizációhoz. A KVM mind a Type-1, mind a Type-2 hypervisorok közé tartozik, mert bár működéséhez egy általános Linux operációs rendszerre van szüksége, egy futó Linux telepítésbe integrálódva tökéletesen képes hypervisorként működni. A KVM-mel telepített virtuális gépek létrehozásához és kezeléséhez a `libvirt` daemont és a kapcsolódó szoftveres segédprogramokat használják.

VirtualBox

Egy népszerű asztali alkalmazás, amely megkönnyíti a virtuális gépek létrehozását és kezelését. Az Oracle VM VirtualBox átível a platformokon, mert Linuxon, macOS-en és Microsoft Windowson is működik. Mivel a VirtualBox futtatásához egy mögöttes operációs rendszerre van szükség, ezért ez egy Type-2 hypervisor.

Egyes hypervisorok lehetővé teszik a virtuális gépek dinamikus áthelyezését. A virtuális gép egyik hypervisor telepítésből a másikba történő áthelyezésének folyamatát *migrációnak* nevezzük, és az ezzel kapcsolatos technikák a hypervisor-megvalósításokban különböznek. Egyes áttelepítések csak akkor végezhetők el, amikor a vendégrendszer teljesen leállt, míg más áttelepítések a vendég futása közben is elvégezhetők (ezt nevezzük *live migrációnak*). Ezek a technikák hasznosak lehetnek a hypervisorok karbantartási időszakai alatt, vagy a rendszer rugalmassága érdekében, amikor egy hypervisor működésképtelenné válik, és a vendéget át lehet helyezni egy működő hypervisorra.

A virtuális gépek típusai

A virtuális gépeknek három fő típusa van: a *teljesen virtualizált* (fully virtualized) vendég, a *paravirtualizált* (paravirtualized) vendég és a *hibrid* (hybrid) vendég.

Teljesen virtualizált

Minden olyan utasításnak, amelyet a vendég operációs rendszer várhatóan végrehajt, képesnek kell lennie a teljes mértékben virtualizált operációs rendszer telepítésén belül is futni. Ennek

oka az, hogy a vendégen belül nem települnek további szoftverillesztőprogramok, amelyek az utasításokat lefordítanák akár a szimulált, akár a valós hardverre. A teljesen virtualizált vendég olyan vendég, ahol a vendég (vagy HardwareVM) nem tudja, hogy az egy futó virtuális gép példánya. Ahhoz, hogy ez a fajta virtualizáció x86-alapú hardveren megvalósulhasson, az Intel VT-x vagy AMD-V CPU-bővítményeket engedélyezni kell a hypervisort telepítő rendszeren. Ezt a BIOS vagy az UEFI firmware konfigurációs menüjéből lehet megtenni.

Paravirtualizált

A paravirtualizált vendég (vagy PVM) olyan vendég, ahol a vendég operációs rendszer tisztában van azzal, hogy ő egy futó virtuális gép példánya. Az ilyen típusú vendégek egy módosított kernelt és speciális illesztőprogramokat (úgynevezett *vendégillesztőprogramokat* (guest drivers)) használnak, amelyek segítenek a vendég operációs rendszernek a hypervisor szoftver- és hardvererőforrásainak használatában. A paravirtualizált vendég teljesítménye gyakran jobb, mint a teljesen virtualizált vendégé, az ilyen szoftverillesztők által biztosított előnynek köszönhetően.

Hibrid

A paravirtualizálás és a teljes virtualizáció kombinálható, így a nem módosított operációs rendszerek közel natív I/O teljesítményt kaphatnak, ha paravirtualizált illesztőprogramokat használnak a teljesen virtualizált operációs rendszereken. A paravirtualizált illesztőprogramok tároló- és hálózati eszközillesztőket tartalmaznak, amelyek fokozott lemez- és hálózati I/O-teljesítményt nyújtanak.

A virtualizációs platformok gyakran kínálnak csomagolt vendégillesztőket (packaged guest drivers) a virtualizált operációs rendszerekhez. A KVM a *Virtio* projektből származó illesztőprogramokat használja, míg az Oracle VM VirtualBox a letölthető ISO CD-ROM image fájlból elérhető *Guest Extensions*-t használja.

Példa a `libvirt` virtuális gépre

Egy olyan virtuális gépet fogunk megvizsgálni, amelyet a `libvirt` kezel, és a KVM hypervisort használja. Egy virtuális gép gyakran egy fájlcsoportból áll, elsősorban egy XML-fájlból, amely *meghatározza* a virtuális gépet (például a hardverkonfigurációt, a hálózati csatlakoztathatóságot, a megjelenítési képességeket és egyebeket), valamint egy kapcsolódó merevlemez image fájlból, amely az operációs rendszer és a szoftver telepítését tartalmazza.

Először is kezdjük el megvizsgálni egy virtuális gép és a hozzá tartozó hálózati környezet XML konfigurációs fájljának példáját:

```
$ ls /etc/libvirt/qemu
total 24
```

```
drwxr-xr-x 3 root root 4096 Oct 29 17:48 networks
-rw------- 1 root root 5667 Jun 29 17:17 rhel8.0.xml
```

NOTE

A mappa elérési útvonalának `qemu` része a KVM-alapú virtuális gépek alapjául szolgáló szoftverre utal. A QEMU projekt a hypervisor számára biztosít szoftvert a virtuális gép által használt hardvereszközök emulálásához, mint például a lemezvezérlők, a gazda CPU-hoz való hozzáférés, a hálózati kártya emulációja és így tovább.

Vegyük észre, hogy van egy `networks` nevű mappa. Ez a mappa olyan (szintén XML-t használó) definíciós fájlokat tartalmaz, amelyek olyan hálózati konfigurációkat hoznak létre, amelyeket a virtuális gépek használhatnak. Ez a hypervisor csak egy hálózatot használ, ezért csak egy definíciós fájlja van, amely egy virtuális hálózati szegmens konfigurációját tartalmazza, amelyet ezek a rendszerek használni fognak.

```
$ ls -l /etc/libvirt/qemu/networks/
total 8
drwxr-xr-x 2 root root 4096 Jun 29 17:15 autostart
-rw------- 1 root root  576 Jun 28 16:39 default.xml
$ sudo cat /etc/libvirt/qemu/networks/default.xml
<!--
WARNING: THIS IS AN AUTO-GENERATED FILE. CHANGES TO IT ARE LIKELY TO BE
OVERWRITTEN AND LOST. Changes to this xml configuration should be made using:
  virsh net-edit default
or other application using the libvirt API.
-->

<network>
  <name>default</name>
  <uuid>55ab064f-62f8-49d3-8d25-8ef36a524344</uuid>
  <forward mode='nat'/>
  <bridge name='virbr0' stp='on' delay='0'/>
  <mac address='52:54:00:b8:e0:15'/>
  <ip address='192.168.122.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.122.2' end='192.168.122.254'/>
    </dhcp>
  </ip>
</network>
```

Ez a definíció tartalmaz egy Class C magánhálózatot és egy emulált hardvereszközt, amely a hálózat routereként működik. A hypervisor számára egy DHCP-kiszolgáló implementációval

használható IP-címek tartománya is rendelkezésre áll, amelyeket az ezt a hálózatot használó virtuális gépekhez lehet hozzárendelni. Ez a hálózati konfiguráció a *hálózati címfordítást* (Network Address Translation, NAT) is használja a csomagok más hálózatokra, például a hypervisor LAN-jára történő továbbítására.

Most egy Red Hat Enterprise Linux 8 virtuális gép definíciós fájljára fordítjuk figyelmünket. (A külön figyelmet érdemlő részek félkövér szöveggel vannak szedve):

```
$ sudo cat /etc/libvirt/qemu/rhel8.0.xml
<!--
WARNING: THIS IS AN AUTO-GENERATED FILE. CHANGES TO IT ARE LIKELY TO BE
OVERWRITTEN AND LOST. Changes to this xml configuration should be made using:
  virsh edit rhel8.0
or other application using the libvirt API.
-->

<domain type='kvm'>
  <name>rhel8.0</name>
  <uuid>fadd8c5d-c5e1-410e-b425-30da7598d0f6</uuid>
  <metadata>
    <libosinfo:libosinfo xmlns:libosinfo="http://libosinfo.org/xmlns/libvirt/domain/1.0">
      <libosinfo:os id="http://redhat.com/rhel/8.0"/>
    </libosinfo:libosinfo>
  </metadata>
  <memory unit='KiB'>4194304</memory>
  <currentMemory unit='KiB'>4194304</currentMemory>
  <vcpu placement='static'>2</vcpu>
  <os>
    <type arch='x86_64' machine='pc-q35-3.1'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <vmport state='off'/>
  </features>
  <cpu mode='host-model' check='partial'>
    <model fallback='allow'/>
  </cpu>
  <clock offset='utc'>
    <timer name='rtc' tickpolicy='catchup'/>
    <timer name='pit' tickpolicy='delay'/>
    <timer name='hpet' present='no'/>
```



```
  </clock>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <pm>
    <suspend-to-mem enabled='no'/>
    <suspend-to-disk enabled='no'/>
  </pm>
  <devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='qcow2'/>
      <source file='/var/lib/libvirt/images/rhel8'/>
      <target dev='vda' bus='virtio'/>
      <address type='pci' domain='0x0000' bus='0x04' slot='0x00' function='0x0'/>
    </disk>
    <controller type='usb' index='0' model='qemu-xhci' ports='15'>
      <address type='pci' domain='0x0000' bus='0x02' slot='0x00' function='0x0'/>
    </controller>
    <controller type='sata' index='0'>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1f' function='0x2'/>
    </controller>
    <controller type='pci' index='0' model='pcie-root'/>
    <controller type='pci' index='1' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='1' port='0x10'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'
multifunction='on'/>
    </controller>
    <controller type='pci' index='2' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='2' port='0x11'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x1'/>
    </controller>
    <controller type='pci' index='3' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='3' port='0x12'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x2'/>
    </controller>
    <controller type='pci' index='4' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='4' port='0x13'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x3'/>
    </controller>
```

```
    <controller type='pci' index='5' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='5' port='0x14'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x4'/>
    </controller>
    <controller type='pci' index='6' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='6' port='0x15'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x5'/>
    </controller>
    <controller type='pci' index='7' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='7' port='0x16'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x6'/>
    </controller>
    <controller type='virtio-serial' index='0'>
      <address type='pci' domain='0x0000' bus='0x03' slot='0x00' function='0x0'/>
    </controller>
    <interface type='network'>
      <mac address='52:54:00:50:a7:18'/>
      <source network='default'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000' bus='0x01' slot='0x00' function='0x0'/>
    </interface>
    <serial type='pty'>
      <target type='isa-serial' port='0'>
        <model name='isa-serial'/>
      </target>
    </serial>
    <console type='pty'>
      <target type='serial' port='0'/>
    </console>
    <channel type='unix'>
      <target type='virtio' name='org.qemu.guest_agent.0'/>
      <address type='virtio-serial' controller='0' bus='0' port='1'/>
    </channel>
    <channel type='spicevmc'>
      <target type='virtio' name='com.redhat.spice.0'/>
      <address type='virtio-serial' controller='0' bus='0' port='2'/>
    </channel>
    <input type='tablet' bus='usb'>
      <address type='usb' bus='0' port='1'/>
    </input>
    <input type='mouse' bus='ps2'/>
```

```
    <input type='keyboard' bus='ps2'/>
    <graphics type='spice' autoport='yes'>
      <listen type='address'/>
      <image compression='off'/>
    </graphics>
    <sound model='ich9'>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1b' function='0x0'/>
    </sound>
    <video>
      <model type='virtio' heads='1' primary='yes'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x0'/>
    </video>
    <redirdev bus='usb' type='spicevmc'>
      <address type='usb' bus='0' port='2'/>
    </redirdev>
    <redirdev bus='usb' type='spicevmc'>
      <address type='usb' bus='0' port='3'/>
    </redirdev>
    <memballoon model='virtio'>
      <address type='pci' domain='0x0000' bus='0x05' slot='0x00' function='0x0'/>
    </memballoon>
    <rng model='virtio'>
      <backend model='random'>/dev/urandom</backend>
      <address type='pci' domain='0x0000' bus='0x06' slot='0x00' function='0x0'/>
    </rng>
  </devices>
</domain>
```

Ez a fájl számos olyan hardverbeállítást határoz meg, amelyet ez a vendég használ, például a hozzárendelt RAM mennyiségét, a hypervisor CPU-magjainak számát, amelyekhez a vendég hozzáférhet, a vendéghez tartozó merevlemez image fájlt (a `disk` stanza alatt), a megjelenítési képességeit (a SPICE protokollon keresztül), a vendég hozzáférését az USB-eszközökhöz, valamint az emulált billentyűzet- és egérbevitelt.

Példa virtuális gép lemeztárhelyére

A virtuális gép merevlemezének image-e a `/var/lib/libvirt/images/rhel8` címen található. Itt van maga a lemez image ezen a hypervisoron:

```
$ sudo ls -lh /var/lib/libvirt/images/rhel8
-rw------- 1 root root 5.5G Oct 25 15:57 /var/lib/libvirt/images/rhel8
```

A lemez image jelenlegi mérete mindössze 5,5 GB helyet foglal el a hypervisoron. A vendég operációs rendszere azonban 23,3 GB méretű lemezt lát, amint azt a következő parancs kimenete is mutatja a futó virtuális gépen belül:

```
$ lsblk
NAME          MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda           252:0    0 23.3G  0 disk
├─vda1        252:1    0    1G  0 part /boot
└─vda2        252:2    0 22.3G  0 part
  ├─rhel-root 253:0    0   20G  0 lvm  /
  └─rhel-swap 253:1    0  2.3G  0 lvm  [SWAP]
```

Ez a vendéghez használt lemezszolgáltatás típusának köszönhető. A virtuális gépek többféle lemez imaget használhatnak, de a két elsődleges típus a következő:

COW

Az írás közbeni másolás (copy-on-write, más néven *thin-provisioning* vagy *sparse images*) egy olyan módszer, amelynek során egy lemezfájl előre meghatározott felső mérethatárral jön létre. A lemez image mérete csak akkor növekszik, amikor új adatok kerülnek a lemezre. Az előző példához hasonlóan a vendég operációs rendszer látja az előre meghatározott 23,3 GB-os lemezkorlátot, de csak 5,5 GB adatot írt a lemezfájlba. A példa virtuális gépéhez használt lemezkép formátum a `qcow2`, ami egy QEMU COW fájlformátum.

RAW

A *raw* vagy *full* lemeztípus egy olyan fájl, amelynek minden hely előre ki van osztva (pre-allocated). Például egy 10 GB-os nyers lemezképfájl 10 GB tényleges lemezterületet foglal el a hypervisoron. Ez a lemezstílus teljesítménybeli előnyökkel jár, mivel az összes szükséges lemezterület már rendelkezésre áll, így az alapul szolgáló hypervisor csak adatokat írhat a lemezre anélkül, hogy a teljesítményt megterhelné a lemez image ellenőrzése, hogy az még nem érte-e el a korlátot, és a fájl méretének növelése, ha új adatok íródnak rá.

Vannak más virtualizációs kezelőplatformok, például a *Red Hat Enterprise Virtualization* és az *oVirt*, amelyek képesek fizikai lemezeket használni a virtuális gépek operációs rendszerének háttértárolójaként. Ezek a rendszerek tárolóhálózati (Storage Area Network, SAN) vagy hálózatra csatolt tárolóeszközöket (Network Attached Storage, NAS) használhatnak az adatok írásához, és a hypervisor nyomon követi, hogy melyik tárolóhely melyik virtuális géphez tartozik. Ezek a tárolórendszerek olyan technológiákat használhatnak, mint a logikai kötetkezelés (Logical Volume Management, LVM), hogy szükség szerint növeljék vagy csökkentsék a virtuális gépek lemezes tárolóhelyeinek méretét, és hogy segítsék a tárhely pillanatfelvételeinek (snapshot) létrehozását és kezelését.

Virtális gépek templatejei

Mivel a virtuális gépek jellemzően csak fájlok, amelyek egy hypervisoron futnak, könnyen létrehozhatók *templatek* (sablonok), amelyek testreszabhatók az adott telepítési forgatókönyvekhez. Gyakran előfordul, hogy egy virtuális gépen egy alapvető operációs rendszer és néhány előre meghatározott hitelesítési konfigurációs beállítás van a későbbi rendszerindítások megkönnyítése érdekében. Ez csökkenti az új rendszer létrehozásához szükséges időt azáltal, hogy csökkenti a gyakran ismétlődő munkák mennyiségét, például az alapcsomagok telepítését és a helyi beállításokat.

Ez a virtuális gép-sablon később átmásolható egy új vendégrendszerbe. Ebben az esetben az új vendéget átneveznénk, új MAC-címet generálnánk a hálózati interfészéhez, és egyéb módosításokat végeznénk a tervezett felhasználástól függően.

A D-Bus Machine ID

Sok Linux-telepítés a telepítéskor generált gépazonosító számot, az úgynevezett *D-Bus machine ID* -t (gépazonosítót) használja. Ha azonban egy virtuális gépet *klónozunk*, hogy más virtuális gépek telepítéséhez sablonként használjuk, akkor új D-Bus machine ID-t kell létrehozni annak biztosítására, hogy a hypervisor rendszererőforrásai a megfelelő vendégrendszerhez kerüljenek.

A következő paranccsal ellenőrizhetjük, hogy létezik-e D-Bus machine ID a futó rendszerhez:

```
$ dbus-uuidgen --ensure
```

Ha nem jelenik meg hibaüzenet, akkor a rendszer rendelkezik azonosítóval. Az aktuális D-Bus machine ID megtekintéséhez futtassuk a következőt:

```
$ dbus-uuidgen --get
17f2e0698e844e31b12ccd3f9aa4d94a
```

A megjelenő szöveges karakterlánc az aktuális azonosítószám. Egy hypervisoron futó két Linux rendszer nem rendelkezhet ugyanazzal a D-Bus machine ID-vel.

A D-Bus machine ID a `/var/lib/dbus/machine-id` mappában található, és szimbolikusan linkelve van az `/etc/machine-id` mappához. Ennek az azonosítószámnak a megváltoztatása egy futó rendszeren nem javasolt, mivel a rendszer instabilitása és összeomlása valószínűsíthető. Ha két virtuális gépnek ugyanaz a D-Bus machine ID-je van, az alábbi eljárást követve hozhatunk létre egy újat:

```
$ sudo rm -f /etc/machine-id
$ sudo dbus-uuidgen --ensure=/etc/machine-id
```

Abban az esetben, ha a `/var/lib/dbus/machine-id` nem egy szimbolikus link a `/etc/machine-id`-re, akkor a `/var/lib/dbus/machine-id`-t el kell távolítani.

Virtuális gépek telepítése a felhőbe

Számos IaaS (*Infrastructure as a Service*, szolgáltatott infrastruktúra) szolgáltató áll rendelkezésre, amelyek hypervisor rendszereket futtatnak, és virtuális vendég-image-ket tudnak telepíteni egy szervezet számára. Gyakorlatilag ezek a szolgáltatók mindegyike rendelkezik olyan eszközökkel, amelyek lehetővé teszik a rendszergazdák számára, hogy különböző Linux-disztribúciókon alapuló egyéni virtuális gépeket készítsenek, telepítsenek és konfiguráljanak. E vállalatok közül sokan rendelkeznek olyan rendszerekkel is, amelyek lehetővé teszik az ügyfél szervezetén belül létrehozott virtuális gépek telepítését és migrációját.

Egy Linux-rendszer IaaS-környezetben történő telepítésének értékelésekor a rendszergazdának tisztában kell lennie néhány kulcsfontosságú elemmel:

Computing Instances

“Computing instances” (számítási példányok) alapján számolja fel a használati díjakat, vagyis hogy mennyi CPU-időt használ a felhőalapú infrastruktúra. Annak gondos megtervezése, hogy az alkalmazásoknak mennyi feldolgozási időre lesz ténylegesen szükségük, segít abban, hogy a felhőmegoldás költségei kezelhetőek maradjanak.

A számítási példányok gyakran a felhőkörnyezetben rendelkezésre bocsátott virtuális gépek számára is utalnak. Az egyszerre futó rendszerek minél több példánya szintén befolyásolja, hogy mennyi CPU-időért kell fizetnie a szervezetnek.

Block Storage

A felhőszolgáltatók különböző szintű blokk-tárolókkal is rendelkeznek, amelyeket egy szervezet használhat. Egyes ajánlatok egyszerűen a fájlok webalapú hálózati tárolására szolgálnak, más ajánlatok pedig a felhőből biztosított virtuális gépek számára a fájlok tárolására szolgáló külső tárolókra vonatkoznak.

Az ilyen ajánlatok költségei a felhasznált tárhely mennyiségétől és a szolgáltató adatközpontjaiban lévő tárhely sebességétől függően változnak. A gyorsabb tárhely-hozzáférés általában többe kerül, és ezzel szemben a "nyugvó" adatok (mint az archív tárolás) gyakran nagyon olcsók.

Networking

A felhőmegoldások szolgáltatójával való együttműködés egyik fő eleme a virtuális hálózat konfigurálása. Sok IaaS-szolgáltató rendelkezik valamilyen webalapú segédprogrammal, amelyet a különböző hálózati útvonalak, alhálózatok és tűzfal-konfigurációk megtervezéséhez és végrehajtásához lehet használni. Egyesek még DNS-megoldásokat is biztosítanak, hogy az internetre néző rendszerekhez nyilvánosan elérhető FQDN (*teljesen minősített tartománynevek* (fully qualified domain names)) rendelhetők legyenek. Olyan "hibrid" megoldások is rendelkezésre állnak, amelyek VPN (*virtuális magánhálózat* (Virtual Private Network)) segítségével összekapcsolják a meglévő, helyben lévő hálózati infrastruktúrát a felhőalapú infrastruktúrával, így összekapcsolva a kettőt.

Vendégek biztonságos elérése a felhőben

A legelterjedtebb módszer egy távoli virtuális vendég elérésére egy felhőplatformon az OpenSSH szoftver használata. A felhőben található Linux rendszerben az OpenSSH kiszolgáló futna, míg a rendszergazda egy előre megosztott kulcsokkal rendelkező OpenSSH klienst használna a távoli eléréshez.

Egy adminisztrátor a következő parancsot futtatná

```
$ ssh-keygen
```

és követi a promptokat egy nyilvános és egy privát SSH-kulcspár létrehozásához. A privát kulcs a rendszergazda helyi rendszerén marad (a `~/.ssh/` fájlban tárolva), a nyilvános kulcs pedig átmásolódik a távoli felhőrendszerbe, pontosan ugyanezt a módszert használjuk, amikor egy vállalati LAN-on lévő hálózati gépekkel dolgozunk.

Az adminisztrátor a következő parancsot futtatná:

```
$ ssh-copy-id -i <public_key> user@cloud_server
```

Ez a most generált kulcspár nyilvános SSH-kulcsát másolja át a távoli felhőkiszolgálóra. A nyilvános kulcsot a felhőkiszolgáló `~/.ssh/authorized_keys` fájljába rögzíti, és beállítja a megfelelő jogosultságokat a fájlhoz.

NOTE

Ha csak egy nyilvános kulcsfájl van a `~/.ssh/` mappában, akkor az `-i` kapcsoló elhagyható, mivel az `ssh-copy-id` parancs alapértelmezés szerint a mappában lévő nyilvános kulcsfájlt fogja használni (általában a `.pub` kiterjesztéssel végződő fájlt).

Egyes felhőszolgáltatók automatikusan generálnak egy kulcspárt, amikor egy új Linux-rendszert biztosítanak. A rendszergazdának ezután le kell töltenie a felhőszolgáltatótól az új rendszerhez tartozó privát kulcsot, és azt a helyi rendszerén kell tárolnia. Ne feledjük, hogy az SSH kulcsok engedélyei a magánkulcsok esetében a `0600`, a nyilvános kulcsok esetében pedig a `0644` kell, hogy legyenek.

Felhőrendszerek előzetes konfigurálása

Egy hasznos eszköz, amely leegyszerűsíti a felhőalapú virtuális gépek telepítését, a `cloud-init` segédprogram. Ez a parancs, a hozzá tartozó konfigurációs fájlokkal és az előre definiált virtuális gépek image-ivel együtt egy gyártófüggetlen módszer egy Linux guest telepítésére az IaaS szolgáltatók sokaságában. A YAML (*YAML Ain't Markup Language*) egyszerű szöveges fájlok segítségével a rendszergazda előre konfigurálhatja a hálózati beállításokat, a szoftvercsomagok kiválasztását, az SSH-kulcsok konfigurálását, a felhasználói fiókok létrehozását, a helyi beállításokat, valamint számtalan más opciót az új rendszerek gyors kiépítéséhez.

Egy új rendszer első indítása során a `cloud-init` beolvassa a beállításokat a YAML konfigurációs fájlokból és alkalmazza azokat. Ezt a folyamatot csak a rendszer kezdeti beállításakor kell alkalmazni, és megkönnyíti az új rendszerek sokaságának telepítését egy felhőszolgáltató platformjára.

A `cloud-init` segítségével használt YAML fájl szintaxisát *cloud-config*-nak hívják. Íme egy `cloud-config` példafájl:

```
#cloud-config
timezone: Africa/Dar_es_Salaam
hostname: test-system

# Update the system when it first boots up
apt_update: true
apt_upgrade: true

# Install the Nginx web server
packages:
 - nginx
```

Figyeljük meg, hogy a felső sorban nincs szóköz a hash szimbólum (`#`) és a `cloud-config` kifejezés között!

NOTE

A `cloud-init` nem csak a virtuális gépekhez használható. A `cloud-init` eszközkészlet a konténerek (például az LXD Linux konténerek) telepítés előtti

előkonfigurálására is használható.

Konténerek

A konténertechnológia bizonyos szempontból hasonlít a virtuális gépekhez, ahol egy izolált környezetet kapunk egy alkalmazás egyszerű telepítéséhez. Míg a virtuális gép esetében egy teljes számítógépet emulálnak, a konténer csak annyi szoftvert használ, amennyi az alkalmazás futtatásához szükséges. Ily módon sokkal kevesebb az overhead.

A konténerek nagyobb rugalmasságot tesznek lehetővé, mint a virtuális gépek. Egy alkalmazáskonténer ugyanúgy átvihető egyik állomásról a másikra, mint ahogy egy virtuális gép is átvihető egyik hypervisorról a másikra. Néha azonban egy virtuális gépet ki kell kapcsolni, mielőtt migrálni lehetne, míg egy konténer esetében az alkalmazás mindig fut, amíg migrálódik. A konténerek megkönnyítik az alkalmazások új verzióinak a meglévő verzióval párhuzamos telepítését is. Ahogy a felhasználók lezárják a munkameneteiket a futó konténerekkel, a konténereket a konténer-összehangoló szoftver automatikusan eltávolíthatja a rendszerből, és az új verzióval helyettesítheti őket, így csökkentve az állásidőt.

NOTE Számos konténertechnológia áll rendelkezésre Linuxhoz, például a *Docker*, *Kubernetes*, *LXD/LXC*, *systemd-nspawn*, *OpenShift* és még sok más. Egy konténer-szoftvercsomag pontos implementálása meghaladja az LPIC-1 vizsga kereteit.

A konténerek a Linux kernel *kontrollcsoportok* (control groups, ismertebb nevén *cgroups*) mechanizmusát használják. A cgroup az egyik lehetőség arra, hogy a rendszer erőforrásait, például a memóriát, a processzoridőt, valamint a lemez- és hálózati sávszélességet egy-egy alkalmazás számára felosszuk. A rendszergazda közvetlenül használhatja a cgroupokat arra, hogy a rendszer erőforrás-korlátjait beállítsa egy alkalmazás vagy alkalmazáscsoportok számára, amelyek egyetlen cgroupon belül létezhetnek. Lényegében ezt teszi a renszergazda számára a konténerszoftver, valamint olyan eszközöket biztosít, amelyek megkönnyítik a cgroups kezelését és telepítését.

NOTE Jelenleg a cgroups ismerete nem szükséges az LPIC-1 vizsga teljesítéséhez. A cgroup fogalmát azért említjük itt, hogy a jelölt legalább némi háttérismerettel rendelkezzen arról, hogyan különül el egy alkalmazás a rendszer erőforrásainak kihasználása érdekében.

Gyakorló feladatok

1. Milyen CPU-bővítésekre van szükség egy x86 alapú hardverplatformon, amely teljesen virtualizált vendégeket futtat?

2. Egy kritikus fontosságú kiszolgáló telepítése, amely a leggyorsabb teljesítményt igényli, valószínűleg milyen típusú virtualizációt használ?

3. Két, ugyanabból a sablonból klónozott és D-Bus-t használó virtuális gép teljesítménye hibásan működik. Mindkettőnek különálló hosztneve és hálózati konfigurációs beállításai vannak. Milyen paranccsal lehetne meghatározni, hogy az egyes virtuális gépek különböző D-Bus machine ID-kkal rendelkeznek-e?

Gondolkodtató feladatok

1. Futtassuk a következő parancsot, hogy ellenőrizzük, hogy a rendszeren már engedélyezve vannak-e a CPU-bővítések egy virtuális gép futtatásához (az eredmények a CPU-tól függően változhatnak):

   ```
   grep --color -E "vmx|svm" /proc/cpuinfo
   ```

   A kimenettől függően a `vmx` (Intel VT-x engedélyezett CPU-k esetén) vagy az `svm` (AMD SVM engedélyezett CPU-k esetén) van kiemelve. Ha nem kapunk eredményt, nézzük meg a BIOS vagy az UEFI firmware utasításait a virtualizáció engedélyezéséről a processzor számára.

2. Ha processzor támogatja a virtualizációt, keressük meg a disztribúció dokumentációját a KVM hypervisor futtatásához!
   - Telepítsük a KVM hypervisor futtatásához szükséges csomagokat!
   - Ha grafikus asztali környezetet használunk, ajánlott telepíteni a `virt-manager` alkalmazást is, ami egy grafikus front-end, amely egy KVM telepítésekor használható. Ez segíti a virtuális gépek telepítését és kezelését!
   - Töltsük le az általunk választott Linux disztribúció ISO image-ét és a disztribúció dokumentációját követve hozzunk létre egy új virtuális gépet az ISO segítségével!

Összefoglalás

Ebben a leckében a virtuális gépek és a konténerek koncepcionális alapjait ismertettük, valamint azt, hogy ezek a technológiák hogyan használhatók Linux alatt.

Röviden ismertettük az alábbi parancsokat:

`dbus-uuidgen`

A rendszer DBus ID-jának ellenőrzésére és megtekintésére szolgál.

`ssh-keygen`

Nyilvános és privát SSH kulcspár generálására szolgál távoli, felhőalapú rendszerek eléréséhez.

`ssh-copy-id`

Egy rendszer nyilvános SSH-kulcsának távoli rendszerre való másolásához használatos, a távoli hitelesítés megkönnyítése érdekében.

`cloud-init`

Virtuális gépek és konténerek felhőkörnyezetbe történő konfigurálásának és telepítésének segítésére szolgál.

Válaszok a gyakorló feladatokra

1. Milyen CPU-bővítésekre van szükség egy x86 alapú hardverplatformon, amely teljesen virtualizált vendégeket futtat?

   Intel CPU-k esetén VT-x, AMD CPU-k esetén pedig AMD-V

2. Egy kritikus fontosságú kiszolgáló telepítése, amely a leggyorsabb teljesítményt igényli, valószínűleg milyen típusú virtualizációt használ?

   A paravirtualizációt - például a Xen-t - vendég operációs rendszerként használó operációs rendszer jobban kihasználhatja a rendelkezésére álló hardveres erőforrásokat a hypervisorral való együttműködésre tervezett szoftverillesztőprogramok használatával.

3. Két, ugyanabból a sablonból klónozott és D-Bus-t használó virtuális gép teljesítménye hibásan működik. Mindkettőnek különálló hosztneve és hálózati konfigurációs beállításai vannak. Milyen paranccsal lehetne meghatározni, hogy az egyes virtuális gépek különböző D-Bus gépazonosítókkal rendelkeznek-e?

   `dbus-uuidgen --get`

Válaszok a gondolkodtató feladatokra

1. Futtassuk a következő parancsot, hogy ellenőrizzük, hogy a rendszeren már engedélyezve vannak-e a CPU-bővítések egy virtuális gép futtatásához (az eredmények a CPU-tól függően változhatnak): `grep --color -E "vmx|svm" /proc/cpuinfo`. A kimenettől függően a `vmx` (Intel VT-x engedélyezett CPU-k esetén) vagy az `svm` (AMD SVM engedélyezett CPU-k esetén) van kiemelve. Ha nem kapunk eredményt, nézzük meg a BIOS vagy az UEFI firmware utasításait a virtualizáció engedélyezéséről a processzor számára.

   Az eredmények az általunk használt CPU-tól függően változnak. Itt egy példa egy Intel CPU-val rendelkező számítógép kimeneti eredményére, amelynek virtualizációs kiterjesztései engedélyezve vannak az UEFI firmware-ben:

```
$ grep --color -E "vmx|svm" /proc/cpuinfo
flags       : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc
art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni
pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1
sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm
3dnowprefetch cpuid_fault epb invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow vnmi
flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm
mpx rdseed adx smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat
pln pts hwp hwp_notify hwp_act_window hwp_epp md_clear flush_l1d
```

2. Ha processzor támogatja a virtualizációt, keressük meg a disztribúció dokumentációját a KVM hipervizor futtatásához!
   - Telepítsük a KVM hipervizor futtatásához szükséges csomagokat!

     Ez a disztribúciótól függően változik, de itt található néhány kiindulási pont:

     Ubuntu — https://help.ubuntu.com/lts/serverguide/libvirt.html

     Fedora — https://docs.fedoraproject.org/en-US/quick-docs/getting-started-with-virtualization/

     Arch Linux — https://wiki.archlinux.org/index.php/KVM

   - Ha grafikus asztali környezetet használunk, ajánlott telepíteni a `virt-manager` alkalmazást is, ami egy grafikus front-end, amely egy KVM telepítésekor használható. Ez segíti a virtuális gépek telepítését és kezelését!

     Ez is a disztribúciótól függ. Ubuntu esetén így fog kinézni:

```
$ sudo apt install virt-manager
```

- Töltsük le az általunk választott Linux disztribúció ISO image-ét és a disztribúció dokumentációját követve hozzunk létre egy új virtuális gépet az ISO segítségével!

  Ezt a feladatot a `virt-manager` csomag könnyedén megoldja. A virtuális gép azonban a parancssorból is létrehozható a `virt-install` paranccsal. Próbáljuk ki mindkét módszert, hogy megértsük a virtuális gépek telepítésének módját.

Linux Professional Institute

Témakör 103: GNU és Unix parancsok

103.1 Munka a parancssorban

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 101, Objective 103.1

Súlyozás

4

Kulcsfontosságú ismeretek

- Egyetlen shell parancs és egysoros parancssorozatok használata alapvető feladatok elvégzéséhez a parancssorban.
- A shell környezet használata és módosítása, beleértve a környezeti változók definiálását, hivatkozást és exportálását.
- Parancselőzmények használata és szerkesztése.
- Parancsok meghívása a meghatározott elérési útvonalon belül és kívül.

A használt fájlok, kifejezések és segédprogramok listája

- `bash`
- `echo`
- `env`
- `export`
- `pwd`
- `set`
- `unset`
- `type`
- `which`

- `man`
- `uname`
- `history`
- `.bash_history`
- Quoting

103.1 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	103 GNU és Unix parancsok
Fejezet:	103.1 Munka a parancssorban
Lecke:	1/2

Bevezetés

A Linux adminisztráció és a Bash shell világába újonnan belépők gyakran elveszettnek érzik magukat a GUI felület megnyugtató kényelme nélkül. Hozzászoktak ahhoz, hogy a jobb egérgombbal kattintva hozzáférnek a grafikus fájlkezelő segédprogramok által elérhetővé tett vizuális jelzésekhez és kontextuális információkhoz. Ezért fontos, hogy gyorsan megtanuljuk és elsajátítsuk a parancssori eszközök viszonylag kis készletét, amelyeken keresztül azonnal hozzáférhetünk a régi GUI által kínált összes adathoz — és még annál is többhöz.

Rendszerinformációk megszerzése

Miközben a parancssor villogó téglalapját nézzük, az első kérdésünk valószínűleg az lesz, hogy “Hol vagyok?”. Pontosabban: “Hol vagyok most a Linux fájlrendszerében, és ha mondjuk létrehoznék egy új fájlt, hol lenne az?”. Amit itt keresünk, az a *present work directory* (aktuális mappa), és a `pwd` parancs megmondja, amit tudni akarunk:

```
$ pwd
/home/frank
```

Tegyük fel, hogy Frank jelenleg be van jelentkezve a rendszerbe, és a home mappájában van: `/home/frank/`. Ha Frank a `touch` paranccsal létrehoz egy üres fájlt anélkül, hogy más helyet adna meg a fájlrendszerben, a fájl a `/home/frank/` mappában jön létre. A mappa tartalmának listázása az `ls` segítségével megmutatja nekünk az új fájlt:

```
$ touch newfile
$ ls
newfile
```

A fájlrendszerben lévő pozíciónkon kívül gyakran szükségünk lesz a Linux rendszerre vonatkozó információkra is. Ilyen lehet például a disztribúció pontos kiadási száma vagy az éppen betöltött Linux kernel verziója. Az `uname` eszköz az, amire itt szükségünk lesz. Különösen a `-a` (“all”) kapcsolóval használt `uname`.

```
$ uname -a
Linux base 4.18.0-18-generic #19~18.04.1-Ubuntu SMP Fri Apr 5 10:22:13 UTC 2019 x86_64
x86_64 x86_64 GNU/Linux
```

Itt az `uname` azt mutatja, hogy Frank gépére a Linux kernel 4.18.0 verziója van telepítve, és Ubuntu 18.04-et futtat 64 bites (`x86_64`) CPU-n.

Parancsinformációk megszerzése

Gyakran találkozunk olyan dokumentációval, amely olyan Linux-parancsokról szól, amelyeket még nem ismerünk. Maga a parancssor mindenféle hasznos információt kínál arról, hogy mit csinálnak a parancsok, és hogyan lehet őket hatékonyan használni. Talán a leghasznosabb információkat a `man` rendszer számos fájljában találjuk.

A Linux fejlesztők általában `man` fájlokat írnak, és ezeket az általuk készített segédprogramokkal együtt terjesztik. A `man` fájlok erősen strukturált dokumentumok, amelyek tartalma intuitív módon, szabványos szakaszcímekkel van felosztva. Ha beírjuk a `man` parancsot, majd a parancs nevét, akkor a parancs nevét, egy rövid használati összefoglalót, egy részletesebb leírást, valamint néhány fontos történelmi és licencelési hátteret tartalmazó információt kapunk. Íme egy példa:

```
$ man uname
UNAME(1)            User Commands           UNAME(1)
NAME
   uname - print system information
SYNOPSIS
   uname [OPTION]...
```

```
DESCRIPTION
   Print certain system information.  With no OPTION, same as -s.
   -a, --all
      print  all information, in the following order, except omit -p
      and -i if unknown:
   -s, --kernel-name
      print the kernel name
   -n, --nodename
      print the network node hostname
   -r, --kernel-release
      print the kernel release
   -v, --kernel-version
      print the kernel version
   -m, --machine
      print the machine hardware name
   -p, --processor
      print the processor type (non-portable)
   -i, --hardware-platform
      print the hardware platform (non-portable)
   -o, --operating-system
      print the operating system
   --help display this help and exit
   --version
      output version information and exit
AUTHOR
   Written by David MacKenzie.
REPORTING BUGS
   GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
   Report uname translation bugs to
   <http://translationproject.org/team/>
COPYRIGHT
   Copyright©2017 Free Software Foundation, Inc. License  GPLv3+: GNU
   GPL version 3 or later <http://gnu.org/licenses/gpl.html>.
   This is free software: you are free to change and redistribute it.
   There is NO WARRANTY, to the extent permitted by law.
SEE ALSO
   arch(1), uname(2)
   Full documentation at: <http://www.gnu.org/software/coreutils/uname>
   or available locally via: info '(coreutils) uname invocation'
GNU coreutils 8.28       January 2018             UNAME(1)
```

A `man` csak akkor működik, ha pontos parancsnevet adunk meg neki. Ha azonban nem vagyunk biztosak a keresett parancs nevében, akkor az `apropos` paranccsal kereshetünk a `man` lapnevek és

leírások között. Feltételezve például, hogy nem emlékszünk arra, hogy az `uname` az, ami megadja az aktuális Linux kernel verzióját, átadhatjuk a `kernel` szót a `apropos` parancsnak. Valószínűleg sok kimeneti sort fogunk kapni, de ezek között kell szerepelnie az alábbiaknak:

```
$ apropos kernel
systemd-udevd-kernel.socket (8) - Device event managing daemon
uname (2)            - get name and information about current kernel
urandom (4)          - kernel random number source devices
```

Ha nincs szükségünk egy parancs teljes dokumentációjára, a `type` használatával gyorsan megkaphatjuk a parancsra vonatkozó alapvető adatokat. Ebben a példában a `type` segítségével egyszerre négy különböző parancsot kérdezhetünk le. Az eredményekből kiderül, hogy a `cp` (“copy”) egy program, amely a `/bin/cp` állományban található, és, hogy a `kill` (a futó folyamat állapotának megváltoztatása) egy shell builtin (beépülő) — ami azt jelenti, hogy valójában magának a Bash shellnek a része:

```
$ type uname cp kill which
uname is hashed (/bin/uname)
cp is /bin/cp
kill is a shell builtin
which is /usr/bin/which
```

Vegyük észre, hogy a `uname` amellett, hogy egy átlagos bináris parancs, mint a `cp`, "hashed." `is. Ez azért van, mert Frank nemrég használta a `uname` parancsot, és a rendszer hatékonyságának növelése érdekében a parancsot egy hash-táblába helyezte, hogy a következő futtatáskor könnyebben elérhető legyen. Ha a rendszerindítás után futtatná a `type uname`-t, Frank azt találná, hogy a `type` ismét egy szabályos bináris állományként írja le a `uname`-t.

NOTE A hash-tábla kitakarításának gyorsabb módja a `hash -d` parancs futtatása.

Néha — különösen, ha automatizált szkriptekkel dolgozunk — szükségünk lesz valami egyszerűbb információforrásra egy parancsról. A `which` parancs, amit az előző `type` parancsunk követett le nekünk, nem ad vissza mást, mint a parancs abszolút helyét. Ez a példa mind a `uname`, mind a `which` parancsot megkeresi.

```
$ which uname which
/bin/uname
/usr/bin/which
```

NOTE Ha a “builtin” parancsokkal kapcsolatos információkat szeretnénk megjeleníteni,

használhatjuk a `help` parancsot.

Parancselőzmények használata

Gyakran gondosan utánanézünk egy parancs megfelelő használatának, és sikeresen futtatjuk azt az opciók és argumentumok bonyolult sorával együtt. De mi történik néhány héttel később, amikor ugyanazt a parancsot kell futtatnunk ugyanazokkal az opciókkal és argumentumokkal, de nem emlékszünk a részletekre? Ahelyett, hogy a kutatást a nulláról kellene újra kezdenünk, gyakran képesek leszünk az eredeti parancsot visszaállítani a `history` segítségével.

A `history` beírása az utoljára használt parancsokat adja vissza, amelyek közül a legutóbbiak jelennek meg utoljára. Könnyen kereshetünk ezekben a parancsokban, ha egy adott sztringet adunk a `grep` parancshoz. Ez a példa minden olyan parancsot megkeres, amely a `bash_history` szöveget tartalmazza:

```
$ history | grep bash_history
1605 sudo find /home -name ".bash_history" | xargs grep sudo
```

Itt egyetlen parancsot kapunk vissza a sorszámával együtt, ami az 1605.

És ha már a `bash_history` szóba került , ez valójában egy rejtett fájl neve, amit a felhasználó home mappájában találhatunk meg. Mivel ez egy rejtett fájl (ezt a fájlnevet megelőző pont jelöli), csak akkor lesz látható, ha a mappa tartalmát az `ls` használatával, a `-a` argumentum segítségével listáztatjuk ki:

```
$ ls /home/frank
newfile
$ ls -a /home/frank
.  ..  .bash_history  .bash_logout  .bashrc  .profile  .ssh  newfile
```

Mi van a `.bash_history` fájlban? Nézzük meg a sajátunkat: a legutóbbi száz és száz parancsunkat fogjuk látni. Meglepődve láthatjuk azonban azt, hogy a *legutóbbi* parancsaink közül néhány hiányzik. Ennek az az oka, hogy míg a dinamikus `history` adatbázisba azonnal bekerülnek, addig a parancstörténet legújabb kiegészítései nem kerülnek be a `.bash_history` fájlba, amíg ki nem lépünk a munkamenetből.

A `history` tartalmát kihasználva sokkal gyorsabbá és hatékonyabbá tehetjük a parancssor használatát a billentyűzet felfelé és lefelé mutató kurzorbillentyűinek használatával. A felfelé billentyű többszöri lenyomásával a parancssor feltöltődik a legutóbbi parancsokkal. Amikor eljutunk ahhoz, amelyet másodszor is végre szeretnénk hajtani, az Enter lenyomásával

futtathatjuk azt. Ez megkönnyíti a parancsok felidézését és ha szükséges, módosítását többször is egy shell munkamenet során.

Gyakorló feladatok

1. A `man` használatával határozzuk meg, hogyan utasítsuk az `apropos`-t egy rövid parancs kiadására úgy, hogy csak egy rövid használati utasítást adjon ki, majd kilépjen!

2. A `man` rendszer segítségével határozzuk meg, hogy a `grep` parancshoz milyen szerzői jogi licenc van rendelve!

Gondolkodtató feladatok

1. Határozzuk meg a számítógépen használt hardverarchitektúra és Linux kernel verzióját könnyen olvasható kimeneti formátumban!

2. Írassuk ki a dinamikus `history` adatbázis és a `.bash_history` fájl utolsó húsz sorát, hogy összehasonlíthassuk őket!

3. Az `apropos` eszközzel azonosítsuk azt a `man` oldalt, ahol megtaláljuk azt a parancsot, amivel megadhatjuk, hogy a csatlakoztatott fizikai blokkeszköz méretét megabájt vagy gigabájt helyett bájtban jelenítse meg!

Összefoglalás

Ebben a leckében megtanultuk:

- Hogyan szerezhetünk információt a fájlrendszer helyéről és az operációs rendszer szoftvercsomagjáról.
- Hogyan találhatunk segítséget a parancsok használatához.
- Hogyan lehet azonosítani a fájlrendszer helyét és a parancs binárisok típusát.
- Hogyan találhatjuk meg és használhatjuk újra a korábban végrehajtott parancsokat. A leckében a következő parancsokról volt szó:

`pwd`
: Az aktuális mappa elérési útvonalát írja ki.

`uname`
: Kiírja a rendszer hardverarchitektúráját, a Linux kernel verzióját, a disztribúciót és a disztribúció kiadását.

`man`
: A parancsok használatát dokumentáló súgófájlok elérése.

`type`
: Kiírja egy vagy több parancs fájlrendszerbeli helyét és típusát.

`which`
: Egy parancs fájlrendszerbeli helyének kiírása.

`history`
: A korábban már végrehajtott parancsok kiíratása vagy újrafelhasználása.

Válaszok a gyakorló feladatokra

1. A `man` használatával határozzuk meg, hogyan utasítsuk az `apropos`-t egy rövid parancs kiadására úgy, hogy csak egy rövid használati utasítást adjon ki, majd kilépjen!

   Futtassuk a `man apropos` parancsot, majd görgessünk le az “Options” szekcióig, amíg meg nem találjuk a `--usage` bekezdést!

2. A `man` rendszer segítségével határozzuk meg, hogy a `grep` parancshoz milyen szerzői jogi licenc van rendelve!

   Futtassuk a `man grep` parancsot és görgessünk le a dokumentum “Copyright” szakaszáig. Vegyük észre, hogy a program a Free Software Foundation egyik szerzői jogát használja!

Válaszok a gondolkodtató feladatokra

1. Határozzuk meg a számítógépen használt hardverarchitektúra és Linux kernel verzióját könnyen olvasható kimeneti formátumban!

   Futtassuk a `man uname` parancsot, olvassuk át a "Description" szakaszt és találjuk meg azokat az argumentumokat, amelyek csak az általunk elvárt eredményeket jelenítik meg. Figyeljük meg, hogy a `-v` a kernel verzióját adja meg, az `-i` pedig a hardverplatformot!

   ```
   $ man uname
   $ uname -v
   $ uname -i
   ```

2. Írassuk ki a dinamikus `history` adatbázis és a `.bash_history` fájl utolsó húsz sorát, hogy összehasonlíthassuk őket!

   ```
   $ history 20
   $ tail -n 20 .bash_history
   ```

3. Az `apropos` eszközzel azonosítsuk azt a `man` oldalt, ahol megtaláljuk azt a parancsot, amivel megadhatjuk, hogy a csatlakoztatott fizikai blokkeszköz méretét megabájt vagy gigabájt helyett bájtban jelenítse meg!

   Az egyik lehetőség az, hogy futtatjuk az `apropos`-t a `block` sztringgel, átolvassuk az eredményeket, megjegyezzük, hogy az `lsblk` blokkos eszközöket listáz (és ezért ez lenne a legvalószínűbb eszköz a mi igényeinkhez), futtatjuk a `man lsblk`-t, végiggörgetjük a "Description" részt, és megjegyezzük, hogy a `-b` megjeleníti az eszköz méretét bájtban. Végül futtassuk az `lsblk -b` parancsot, hogy lássuk, mi jön ki!

   ```
   $ apropos block
   $ man lsblk
   $ lsblk -b
   ```

Linux Professional Institute

103.1 Lecke 2

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	103 GNU és Unix parancsok
Fejezet:	103.1 Munka a parancssorban
Lecke:	2/2

Bevezetés

Az operációs rendszer környezete tartalmazza azokat az alapvető eszközöket — mint például a parancssori shelleket és néha a GUI-t --, amelyekre szükségünk lesz ahhoz, hogy elvégezzünk dolgokat. De a környezet a parancsikonok és az előre beállított értékek katalógusával is rendelkezik. Itt fogjuk megtanulni, hogyan kell ezeket az értékeket felsorolni, meghívni és kezelni.

A környezeti változóink megtalálása

Hogyan azonosítjuk tehát az egyes környezeti változók aktuális értékeit? Az egyik mód az `env` parancs használata:

```
$ env
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
XDG_RUNTIME_DIR=/run/user/1000
XAUTHORITY=/run/user/1000/gdm/Xauthority
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/game
s:/snap/bin
```



```
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
[...]
```

Rengeteg kimenetet fogunk kapni — sokkal többet, mint amennyi a fenti részletben szerepel. De most figyeljük meg a `PATH` bejegyzést, amely azokat a mappákat tartalmazza, ahol a shell (és más programok) más programokat keresnek anélkül, hogy egy teljes elérési utat kellene megadnunk. Ezzel a beállítással futtathatunk egy bináris programot a saját mappánkból, amely mondjuk az `/usr/local/bin` mappában található, és úgy fog futni, mintha a fájl helyi lenne.

Váltsunk témát egy pillanatra. Az `echo` parancs azt írja ki a képernyőre, amit mondunk neki. Akár hisszük, akár nem, sokszor nagyon hasznos lesz, ha a `echo` szó szerint megismétel valamit.

```
$ echo "Szia! Hogy vagy?"
Szia! Hogy vagy?
```

De van még valami, amit a `echo`-val tehetünk. Ha megadjuk egy környezeti változó nevét — és a változó nevének `$` előtaggal való kiegészítésével jelezzük, hogy ez egy változó — akkor ahelyett, hogy csak a változó nevét írná ki, a shell kibővíti azt, és megadja az értékét. Nem tudjuk biztosan, hogy a kedvenc mappánk jelenleg benne van-e az elérési útvonalban? Gyorsan ellenőrizhetjük a `echo` futtatásával:

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/sn
ap/bin
```

Új környezeti változók létrehozása

Saját egyéni változókat adhatunk hozzá a környezethez. A legegyszerűbb megoldás a `=` karakter használata. A bal oldali sztring lesz az új változó neve, a jobb oldali pedig az értéke. A változó nevét most már megadhatjuk a `echo`-nak, hogy kipróbáljuk, működött-e:

```
$ myvar=hello
$ echo $myvar
hello
```

NOTE Vegyük észre, hogy a változó hozzárendelése során az egyenlőségjel egyik oldalán sincs szóköz!

De vajon tényleg működött? Írjuk be a terminálba a `bash` parancsot egy új shell megnyitásához!

Ez az új shell pontosan úgy néz ki, mint az, amelyben az előbb voltunk, de valójában az eredeti shell (amelyet *szülőnek* nevezünk) *gyermeke*. Most ebben az új gyermek shellben próbáljuk meg az `echo`-val elérni ugyanazt, mint korábban. Semmi. Mi történt?

```
$ bash
$ echo $myvar

$
```

Egy változó, amelyet az imént leírt módon hozunk létre, csak helyileg lesz elérhető — a közvetlen shell munkamenetben. Ha egy új shellt indítunk — vagy a munkamenetet az `exit` segítségével zárjuk be — a változó nem tart velünk. Az `exit` beírásával visszatérünk az eredeti szülő shellünkhöz, ami most éppen az a hely, ahol lenni szeretnénk. Ha szeretnénk, még egyszer lefuttathatjuk az `echo $myvar` parancsot, csak hogy megerősítsük, hogy a változó még mindig érvényes. Most írjuk be az `export myvar` parancsot, hogy a változót átadjuk a később megnyitott gyermek shelleknek. Próbáljuk ki: írjuk be a `bash`-t egy új shellhez, majd az `echo`-t:

```
$ exit
$ export myvar
$ bash
$ echo $myvar
hello
```

Mindez kissé ostobának tűnhet, amikor mindenféle cél nélkül hozunk létre shelleket. De annak megértése, hogy a shellváltozók hogyan terjednek a rendszerben, nagyon fontos lesz, amint komoly szkripteket kezdünk írni.

Környezeti változók törlése

Hogyan törölhetjük az összes létrehozott átmeneti változót? Az egyik módja, hogy egyszerűen bezárjuk a szülő shellt — vagy újraindítjuk a számítógépet. De vannak egyszerűbb módszerek is, ilyen például az `unset`. Az `unset` beírásával (a `$` nélkül) a változót kitöröljük, erről az `echo`-val megbizonyosodhatunk.

```
$ unset myvar
$ echo $myvar

$
```

Ha van egy `unset` parancs, akkor biztosak lehetünk benne, hogy van hozzá egy `set` parancs is. A `set` önmagában történő futtatása rengeteg kimenetet fog megjeleníteni, de ez valójában nem sokban különbözik attól, amit az `env` adott. Nézzük meg a kimenet első sorát, amit akkor kapunk, ha a `PATH`-ra szűrünk:

```
$ set | grep PATH
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/game
s:/snap/bin
[...]
```

Mi a különbség a `set` és az `env` között? A mi céljaink szempontjából a lényeg az, hogy a `set` az összes változót és függvényt kiadja. Hadd illusztráljuk ezt: létrehozunk egy új változót `mynewvar` néven, majd megerősítjük, hogy ott van:

```
$ mynewvar=goodbye
$ echo $mynewvar
goodbye
```

Most az `env` futtatása, miközben a `grep`-et használjuk a `mynewvar` sztring szűrésére, nem jelenít meg semmilyen kimenetet. De a `set` futtatása ugyanígy megmutatja nekünk a helyi változót.

```
$ env | grep mynewvar

$ set | grep mynewvar
mynewvar=goodbye
```

Quoting to Escape Special Characters

Ez most a legjobb alkalom arra, hogy megismerkedjünk a különleges karakterek problémájával. Az alfanumerikus karaktereket (a-z és 0-9) a Bash általában szó szerint értelmezi. Ha megpróbálunk létrehozni egy `myfile` nevű új fájlt, akkor csak írjuk be a `touch`-t, majd a `myfile`-t, és a Bash tudni fogja, hogy mit kell vele kezdeni. Ha azonban különleges karaktert akarunk a fájlnévbe tenni, akkor egy kicsit több munkát kell végeznünk.

Ennek illusztrálására írjuk be a `touch` parancsot, és utána a következő címet: `my big file`. A probléma az, hogy a szavak között két szóköz van, amit a Bash értelmez. Bár technikailag a szóközt nem neveznénk “karakter”-nek, abban az értelemben az, mert a Bash nem szó szerint fogja olvasni. Ha listázzuk az aktuális mappa tartalmát, ahelyett, hogy egyetlen `my big file` nevű fájlt találnánk, három fájlt fogunk látni, amelyek neve `my`, `big` és `file`. Ez azért van, mert a

Bash úgy gondolta, hogy több fájlt akarunk létrehozni, amelyek nevét egy listában adjuk át:

```
$ touch my big file
$ ls
my big file
```

A szóközöket ugyanúgy értelmezi a rendszer, ha a három fájlt egy paranccsal töröljük (`rm`):

```
$ rm my big file
```

Most próbáljuk ki a helyes módszert. Írjuk be a `touch` és a fájlnév három részét, de ezúttal a nevet zárjuk idézőjelbe. Ezúttal működni fog, a mappa tartalmának listázása során egyetlen fájlt fogunk látni a megfelelő névvel.

```
$ touch "my big file"
$ ls
'my big file'
```

Ugyanezt a hatást más módon is elérhetjük. Az aposztrófok például ugyanolyan jól működnek, mint a kettős idézőjelek. (Vegyük észre, hogy az aposztrófok minden karakter szó szerinti értékét megőrzik, míg a kettős idézőjelek minden karaktert megőriznek, kivéve a `$`, `` ` ``, `\` és bizonyos esetekben a `!` karaktereket.)

```
$ rm 'my big file'
```

Ha minden speciális karaktert backslash-el kezdünk, akkor a karakter különlegessége “feloldódik” (escape), és a Bash szó szerint olvassa azt.

```
$ touch my\ big\ file
```

Gyakorló feladatok

1. Az `export` paranccsal adjunk hozzá egy új mappát az elérési útvonalhoz (újraindítás után nem marad meg)!

2. Töröljük a `PATH` változót az `unset` paranccsal! Próbáljunk meg futtatni egy parancsot (például a `sudo cat /etc/shadow`) a `sudo` használatával. Mi történt? Miért? (A parancsértelmezőből való kilépés visszaállítja az eredeti állapotot.)

Gondolkodtató feladatok

1. Keressük meg az interneten a különleges karakterek teljes listáját!
2. Próbáljuk ki a parancsok futtatását speciális karakterekből álló sztringek használatával, és különböző feloldó módszerekkel! Vannak különbségek a módszerek viselkedése között?

Összefoglalás

Ebben a leckében megtanultuk:

- Hogyan azonosítsuk be a rendszer környezeti változóit.
- Hogyan hozzunk létre saját környezeti változókat és hogyan exportáljuk azokat más shellekbe.
- Hogyan távolítsuk el a környezeti változókat, és hogyan használjuk az `env` és `set` parancsokat.
- Hogyan oldjuk fel a speciális karaktereket, hogy a Bash szó szerint értelmezze őket.

A leckében a következő parancsokról volt szó:

`echo`
: A bemeneti sztringek és változók kiírása.

`env`
: A környezeti változók megértése és módosítása.

`export`
: Környezeti változó átadása a gyermek shelleknek.

`unset`
: A shell változók és függvények értékeinek és attribútumainak visszaállítása.

Válaszok a gyakorló feladatokra

1. Az `export` paranccsal adjunk hozzá egy új mappát az elérési útvonalhoz (újraindítás után nem marad meg)!

   Ideiglenesen hozzáadhatunk egy új mappát (talán a `myfiles` nevű mappát, amely a home mappában van) az elérési útvonalhoz az `export PATH="/home/yourname/myfiles:$PATH"` paranccsal. Hozzunk létre egy egyszerű szkriptet a `myfiles/` mappában, tegyük futtathatóvá, és próbáljuk meg egy másik mappából futtatni. Ezek a parancsok azt feltételezik, hogy a home mappában vagyunk, amely tartalmaz egy `myfiles` nevű mappát.

   ```
   $ touch myfiles/myscript.sh
   $ echo '#!/bin/bash' >> myfiles/myscript.sh
   $ echo 'echo Hello' >> myfiles/myscript.sh
   $ chmod +x myfiles/myscript.sh
   $ myscript.sh
   Hello
   ```

2. Töröljük a `PATH` változót az `unset` paranccsal! Próbáljunk meg futtatni egy parancsot (például a `sudo cat /etc/shadow`) a `sudo` használatával. Mi történt? Miért? (A parancsértelmezőből való kilépés visszaállítja az eredeti állapotot.)

   Az `unset PATH` beírása törli az aktuális útvonal beállításokat. Egy bináris program abszolút cím nélküli meghívása sikertelen lesz. Emiatt a sudo parancs futtatása (amely maga is egy bináris program, amely a `/usr/bin/sudo` mappában található) sikertelen lesz - hacsak nem adjuk meg az abszolút helyét, például így: `/usr/bin/sudo /bin/cat /etc/shadow`. A `PATH`-t az `export` használatával vagy egyszerűen a shellből való kilépéssel állíthatjuk vissza.

Válaszok a gondolkodtató feladatokra

1. Keressük meg az interneten a különleges karakterek teljes listáját!

   Itt egy lista: `& ; | * ? " ' [ ] ( ) $ < > { } # / \ ! ~`.

2. Próbáljuk ki a parancsok futtatását speciális karakterekből álló sztringek használatával, és különböző feloldó módszerekkel! Vannak különbségek a módszerek viselkedése között?

   A `"` karakterekkel történő feloldás megtartja a dollárjel, a backtick és a backslash speciális jelentését. A `'` karakterekkel történő feloldás azonban *minden* karaktert szó szerint értelmez.

```
$ echo "$mynewvar"
goodbye
$ echo '$mynewvar'
$mynewvar
```

103.2 Szövegfolyamok feldolgozása szűrőkkel

Hivatkozás az LPI célkitűzésre

LPIC-1 v5, Exam 101, Objective 103.2

Súlyozás

2

Kulcsfontosságú ismeretek

- Szövegfájlok és kimeneti adatfolyamok átküldése szöveges segédprogram-szűrőkön a kimenet módosítására a GNU textutils csomagban található szabványos UNIX-parancsok használatával.

A használt fájlok, kifejezések és segédprogramok listája

- `bzcat`
- `cat`
- `cut`
- `head`
- `less`
- `md5sum`
- `nl`
- `od`
- `paste`
- `sed`
- `sha256sum`
- `sha512sum`

- `sort`
- `split`
- `tail`
- `tr`
- `uniq`
- `wc`
- `xzcat`
- `zcat`

103.2 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	103 GNU és Unix parancsok
Fejezet:	103.2 Szövegfolyamok feldolgozása szűrőkkel
Lecke:	1/1

Bevezetés

A szöveggel való foglalkozás minden rendszergazda munkájának fontos részét képezi. Doug McIlroy, az eredeti Unix fejlesztőcsapatának egyik tagja foglalta össze a Unix filozófiáját, és azt mondta (más fontos dolgok mellett): “Írj programokat a szövegfolyam kezelésére, mert az egy univerzális interfész.” A Linuxot a Unix operációs rendszer ihlette, és határozottan átveszi annak filozófiáját, így egy rendszergazdának sok szövegkezelő eszközre kell számítania egy Linux-disztribúción belül.

Az átirányítások és a csővezetékek gyors áttekintése

Szintén a Unix filozófiájából:

- Írjunk olyan programokat, amelyek egy dolgot csinálnak, és azt jól csinálják.
- Írjunk olyan programokat, amelyek együtt dolgoznak.

A programok együttműködésének egyik fő módja a *csővezetékezés* (piping) és az *átirányítások* (redirections). Szinte minden szövegmanipuláló programunk szöveget kap egy szabványos bemenetről (*stdin*), majd kiadja azt egy szabványos kimenetre (*stdout*), és az esetleges hibákat egy

szabványos hibakimenetre (*stderr*) küldi. Hacsak nem adunk meg mást, a szabványos bemenet az lesz, amit a billentyűzeten beírunk (a program az Enter billentyű lenyomása után olvassa be). Hasonlóképpen, a standard kimenet és a hibák a terminál képernyőjén fognak megjelenni. Lássuk, hogyan működik ez!

A terminálba írjuk be a `cat` parancsot, majd nyomjuk meg az Enter billentyűt! Ezután írjunk be valami véletlenszerű szöveget!

```
$ cat
Ez egy teszt
Ez egy teszt
Hey!
Hey!
Mindent megismétel, amit beírok!
Mindent megismétel, amit beírok!
(Meg fogom nyomni a ctrl+c-t és megállítom ezt a nonszensz viselkedést)
(Meg fogom nyomni a ctrl+c-t és megállítom ezt a nonszensz viselkedést)
^C
```

A `cat` parancsról (a kifejezés a `concatenate` szóból származik) további információkat a man oldalakon találhatunk.

NOTE

Ha egy nagyon egyszerű Linux szerver telepítésén dolgozunk, akkor néhány parancs, mint például az `info` és a `less` nem biztos, hogy elérhető lesz. Ha ez a helyzet, telepítsük ezeket az eszközöket a megfelelő eljárással az ehhez a témához tartozó leckékben leírtak szerint.

Mint fentebb bemutattuk, ha nem adjuk meg, hogy a `cat` honnan olvasson, akkor a standard bemenetről fog olvasni (bármit is írunk be), és amit olvas, azt a terminál ablakba (a standard kimenetre) adja ki.

Próbáljuk ki a következőt:

```
$ cat > mytextfile
Ez egy teszt
Remélem, hogy a cat eltárolja ezt a mytextfile-ba, mivel átirányítottam a kimenetet
Megnyomom a ctrl+c-t és ellenőrzöm
^C

$ cat mytextfile
Ez egy teszt
Remélem, hogy a cat eltárolja ezt a mytextfile-ba, mivel átirányítottam a kimenetet
```

```
Megnyomom a ctrl+c-t és ellenőrzöm
```

A `>` (nagyobb, mint) azt mondja a `cat`-nek, hogy a kimenetét a `mytextfile` fájlba irányítsa, ne pedig a standard kimenetre. Most próbáljuk ki ezt:

```
$ cat mytextfile > mynewtextfile
$ cat mynewtextfile
Ez egy teszt
Remélem, hogy a cat eltárolja ezt a mytextfile-ba, mivel átirányítottam a kimenetet
Megnyomom a ctrl+c-t és ellenőrzöm
```

Ennek hatására a `mytextfile` másolódik az `mynewtextfile`-ba. A `diff` végrehajtásával ellenőrizhetjük, hogy a két fájl tartalma megegyezik-e:

```
$ diff mynewtextfile mytextfile
```

Mivel nincs kimenet, a fájlok megegyeznek. Most próbáljuk ki az append átirányítási operátort (`>>`):

```
$ echo 'Ez az új sor' >> mynewtextfile
$ diff mynewtextfile mytextfile
4d3
< Ez az új sor
```

Eddig átirányításokat használtunk fájlok létrehozására és manipulálására. A csöveket (pipes) (a `|` szimbólummal ábrázolva) arra is használhatjuk, hogy egy program kimenetét átirányítsuk egy másik programba. Keressük meg azokat a sorokat, ahol az “ez” szó található:

```
$ cat mytextfile | grep this
Remélem, hogy a cat eltárolja ezt a mytextfile-ba, mivel átirányítottam a kimenetet
Megnyomom a ctrl+c-t és ellenőrzöm

$ cat mytextfile | grep -i this
Ez egy teszt
Remélem, hogy a cat eltárolja ezt a mytextfile-ba, mivel átirányítottam a kimenetet
Megnyomom a ctrl+c-t és ellenőrzöm
```

Most a `cat` kimenetét átirányítottuk egy másik parancsba: A `grep` parancsot egy másik parancsra küldtük: `grep`. Figyeljük meg, hogy ha figyelmen kívül hagyjuk az esetet (a `-i` opciót használva),

akkor egy extra sort kapunk eredményként.

Szövegfolyamok feldolgozása

Tömörített fájl olvasása

Létrehozunk egy `ftu.txt` nevű fájlt, amely a következő parancsok listáját tartalmazza:

```
bzcat
cat
cut
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
sort
split
tail
tr
uniq
wc
xzcat
zcat
```

Most a `grep` paranccsal kiírjuk az összes olyan sort, amely tartalmazza a `cat` karakterláncot:

```
$ cat ftu.txt | grep cat
bzcat
cat
xzcat
zcat
```

Egy másik módja ennek az információnak az, hogy a `grep` parancsot használjuk a szöveg közvetlen szűrésére, anélkül, hogy egy másik alkalmazást kellene használni a szövegfolyam elküldésére az `stdout`-ba.

```
$ grep cat ftu.txt
bzcat
cat
xzcat
zcat
```

NOTE Ne feledjük, hogy ugyanazt a feladatot többféleképpen is el lehet végezni Linux alatt!

Vannak más parancsok is, amelyek tömörített fájlokat kezelnek (`bzcat` a `bzip` tömörített fájlokra, `xzcat` az `xz` tömörített fájlokra és `zcat` a `gzip` tömörített fájlokra), és mindegyik az adott tömörített fájl tartalmának megtekintésére szolgál az alkalmazott tömörítési algoritmus alapján.

Ellenőrizzük, hogy az újonnan létrehozott `ftu.txt` fájl az egyetlen a mappában, majd hozzuk létre a fájl `gzip` tömörített változatát:

```
$ ls ftu*
ftu.txt

$ gzip ftu.txt
$ ls ftu*
ftu.txt.gz
```

Ezután használjuk a `zcat` parancsot a gzippelt tömörített fájl tartalmának megtekintéséhez:

```
$ zcat ftu.txt.gz
bzcat
cat
cut
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
sort
split
tail
```

```
tr
uniq
wc
xzcat
zcat
```

Vegyük figyelembe, hogy a `gzip` az `ftu.txt` fájlt `ftu.txt.gz`-be tömöríti, és eltávolítja az eredeti fájlt. Alapértelmezés szerint a `gzip` parancs kimenete nem jelenik meg. Ha azonban szeretnénk, hogy a `gzip` kiírja, hogy mit csinál, használjuk a `-v` opciót a "verbose" kimenethez.

Fájl megtekintése lapozóban

Tudjuk, hogy a `cat` egy fájlt kapcsol össze a szabványos kimenettel (ha a parancs után megadjuk a fájlt). A `/var/log/syslog` fájlban tárol a Linux minden fontos eseményt, ami a rendszerben történik. A `sudo` parancs használatával bővíthetjük a jogosultságokat, hogy be tudjuk olvasni a `/var/log/syslog` fájlt:

```
$ sudo cat /var/log/syslog
```

...látni fogjuk, hogy az üzenetek nagyon gyorsan gördülnek a terminálablakban. A kimenetet a `less` programba továbbíthatjuk, így az eredmények oldalszámozva lesznek. A `less` használatával a nyílbillentyűkkel navigálhatunk a kimeneten, és a `vi` parancsaihoz hasonló parancsokat is használhatunk a szövegben való navigálásra és keresésre.

Ahelyett azonban, hogy a `cat` parancsot átvezetnénk (csővezetékkel) egy lapozó programba, hasznosabb, ha közvetlenül a lapozási programot használjuk:

```
$ sudo less /var/log/syslog
... (kimenet az átláthatóság kedvéért kihagyva)
```

Szövegfájl egy részének megszerzése

Ha csak a fájl elejét vagy végét kell áttekinteni, más módszerek is rendelkezésre állnak. A `head` paranccsal alapértelmezés szerint a fájl első tíz sorát, a `tail` parancs pedig alapértelmezés szerint a fájl utolsó tíz sorát olvassa be. Most próbáljuk ki:

```
$ sudo head /var/log/syslog
Nov 12 08:04:30 hypatia rsyslogd: [origin software="rsyslogd" swVersion="8.1910.0" x-
pid="811" x-info="https://www.rsyslog.com"] rsyslogd was HUPed
Nov 12 08:04:30 hypatia systemd[1]: logrotate.service: Succeeded.
```

```
Nov 12 08:04:30 hypatia systemd[1]: Started Rotate log files.
Nov 12 08:04:30 hypatia vdr: [928] video directory scanner thread started (pid=882, tid=928,
prio=low)
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'A - ATSC'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'C - DVB-C'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'S - DVB-S'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'T - DVB-T'
Nov 12 08:04:30 hypatia vdr[882]: vdr: no primary device found - using first device!
Nov 12 08:04:30 hypatia vdr: [929] epg data reader thread started (pid=882, tid=929,
prio=high)
$ sudo tail /var/log/syslog
Nov 13 10:24:45 hypatia kernel: [ 8001.679238] mce: CPU7: Core temperature/speed normal
Nov 13 10:24:46 hypatia dbus-daemon[2023]: [session uid=1000 pid=2023] Activating via
systemd: service name='org.freedesktop.Tracker1.Miner.Extract' unit='tracker-
extract.service' requested by ':1.73' (uid=1000 pid=2425 comm="/usr/lib/tracker/tracker-
miner-fs ")
Nov 13 10:24:46 hypatia systemd[2004]: Starting Tracker metadata extractor...
Nov 13 10:24:47 hypatia dbus-daemon[2023]: [session uid=1000 pid=2023] Successfully
activated service 'org.freedesktop.Tracker1.Miner.Extract'
Nov 13 10:24:47 hypatia systemd[2004]: Started Tracker metadata extractor.
Nov 13 10:24:54 hypatia kernel: [ 8010.462227] mce: CPU0: Core temperature above threshold,
cpu clock throttled (total events = 502907)
Nov 13 10:24:54 hypatia kernel: [ 8010.462228] mce: CPU4: Core temperature above threshold,
cpu clock throttled (total events = 502911)
Nov 13 10:24:54 hypatia kernel: [ 8010.469221] mce: CPU0: Core temperature/speed normal
Nov 13 10:24:54 hypatia kernel: [ 8010.469222] mce: CPU4: Core temperature/speed normal
Nov 13 10:25:03 hypatia systemd[2004]: tracker-extract.service: Succeeded.
```

A megjelenített sorok számának szemléltetésére a `head` parancs kimenetét átvezethetjük az `nl` parancsba, amely megjeleníti a parancsba áramló szöveg sorainak számát:

```
$ sudo head /var/log/syslog | nl
1   Nov 12 08:04:30 hypatia rsyslogd: [origin software="rsyslogd" swVersion="8.1910.0" x-
pid="811" x-info="https://www.rsyslog.com"] rsyslogd was HUPed
2   Nov 12 08:04:30 hypatia systemd[1]: logrotate.service: Succeeded.
3   Nov 12 08:04:30 hypatia systemd[1]: Started Rotate log files.
4   Nov 12 08:04:30 hypatia vdr: [928] video directory scanner thread started (pid=882,
tid=928, prio=low)
5   Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'A - ATSC'
6   Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'C - DVB-C'
7   Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'S - DVB-S'
8   Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'T - DVB-T'
9   Nov 12 08:04:30 hypatia vdr[882]: vdr: no primary device found - using first device!
```

```
10  Nov 12 08:04:30 hypatia vdr: [929] epg data reader thread started (pid=882, tid=929,
prio=high)
```

Ugyanezt megtehetjük úgy is, hogy a `tail` parancs kimenetét a `wc` parancsnak adjuk át, amely alapértelmezés szerint a dokumentumban lévő szavak számát számolja, és az `-l` kapcsolóval kiírjuk a parancs által olvasott szövegsorok számát:

```
$ sudo tail /var/log/syslog | wc -l
10
```

Ha a rendszergazdának több (vagy kevesebb) fájl elejét vagy végét kell átnéznie, az `-n` kapcsolóval korlátozhatja a parancsok kimenetét:

```
$ sudo tail -n 5 /var/log/syslog
Nov 13 10:37:24 hypatia systemd[2004]: tracker-extract.service: Succeeded.
Nov 13 10:37:42 hypatia dbus-daemon[2023]: [session uid=1000 pid=2023] Activating via
systemd: service name='org.freedesktop.Tracker1.Miner.Extract' unit='tracker-
extract.service' requested by ':1.73' (uid=1000 pid=2425 comm="/usr/lib/tracker/tracker-
miner-fs ")
Nov 13 10:37:42 hypatia systemd[2004]: Starting Tracker metadata extractor...
Nov 13 10:37:43 hypatia dbus-daemon[2023]: [session uid=1000 pid=2023] Successfully
activated service 'org.freedesktop.Tracker1.Miner.Extract'
Nov 13 10:37:43 hypatia systemd[2004]: Started Tracker metadata extractor.
$ sudo head -n 12 /var/log/syslog
Nov 12 08:04:30 hypatia rsyslogd: [origin software="rsyslogd" swVersion="8.1910.0" x-
pid="811" x-info="https://www.rsyslog.com"] rsyslogd was HUPed
Nov 12 08:04:30 hypatia systemd[1]: logrotate.service: Succeeded.
Nov 12 08:04:30 hypatia systemd[1]: Started Rotate log files.
Nov 12 08:04:30 hypatia vdr: [928] video directory scanner thread started (pid=882, tid=928,
prio=low)
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'A - ATSC'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'C - DVB-C'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'S - DVB-S'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'T - DVB-T'
Nov 12 08:04:30 hypatia vdr[882]: vdr: no primary device found - using first device!
Nov 12 08:04:30 hypatia vdr: [929] epg data reader thread started (pid=882, tid=929,
prio=high)
Nov 12 08:04:30 hypatia vdr: [882] no DVB device found
Nov 12 08:04:30 hypatia vdr: [882] initializing plugin: vnsiserver (1.8.0): VDR-Network-
Streaming-Interface (VNSI) Server
```

A sed, a Stream Editor alapjai

Nézzük meg a többi fájlt, kifejezést és segédprogramot, amelyek nevében nem szerepel a `cat`. Ezt úgy tehetjük meg, hogy a `grep`-nek átadjuk a `-v` kapcsolót, ami arra utasítja a parancsot, hogy csak a `cat`-t nem tartalmazó sorokat adja ki:

```
$ zcat ftu.txt.gz | grep -v cat
cut
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
sort
split
tail
tr
uniq
wc
```

A legtöbbet, amit a `grep`-pel megtehetünk, a `sed`-del is meg lehet — a szöveg szűrésére és átalakítására szolgáló szerkesztővel (ahogyan azt a `sed` kézikönyv oldal is írja). Először helyreállítjuk az `ftu.txt` fájlunkat a fájl `gzip` archívumának kicsomagolásával:

```
$ gunzip ftu.txt.gz
$ ls ftu*
ftu.txt
```

Most már használhatjuk a `sed`-t, hogy csak a `cat` karakterláncot tartalmazó sorokat listázzuk ki:

```
$ sed -n /cat/p < ftu.txt
bzcat
cat
xzcat
zcat
```

A `<<` jelet arra használtuk, hogy az `ftu.txt` fájl tartalmát a `sed` parancsunkba irányítsuk. A

kötőjelek közé zárt szó (pl. `/cat/`) az a kifejezés, amelyet keresünk. A `-n` opció arra utasítja a `sed` parancsot, hogy ne adjon ki semmilyen kimenetet (kivéve a későbbiekben a `p` parancs által utasítottakat). Próbáljuk meg ugyanezt a parancsot az `-n` opció nélkül futtatni, hogy lássuk, mi történik. Ezután pedig próbáljuk ki ezt:

```
$ sed /cat/d < ftu.txt
cut
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
sort
split
tail
tr
uniq
wc
```

Ha nem használjuk az `-n` opciót, akkor a `sed` mindent kiír a fájlból, kivéve azt, amit a `d` utasít a `sed`-nek, hogy törölje a kimenetből.

A `sed`-et gyakran használják a fájlon belüli szöveg keresésére és cseréjére. Tegyük fel, hogy a `cat` minden előfordulását `dog`-ra akarjuk cserélni. Erre a `sed`-et használhatjuk az `s` opció megadásával, hogy az első kifejezés, a `cat` minden egyes példányát kicseréljük a második kifejezésre, a `dog`-ra:

```
$ sed s/cat/dog/ < ftu.txt
bzdog
dog
cut
head
less
md5sum
nl
od
paste
sed
```

```
sha256sum
sha512sum
sort
split
tail
tr
uniq
wc
xzdog
zdog
```

Ahelyett, hogy átirányító operátorral (<) adnánk át az `ftu.txt` fájlt a `sed` parancsnak, a `sed` közvetlenül a fájlon is dolgozhat. Ezt próbáljuk ki a következőkben, miközben egyidejűleg biztonsági másolatot készítünk az eredeti fájlról:

```
$ sed -i.backup s/cat/dog/ ftu.txt
$ ls ftu*
ftu.txt  ftu.txt.backup
```

Az `-i` opció egy helybeni `sed` műveletet hajt végre az *eredeti* fájlon. Ha az `-i` paraméter után nem használjuk a `.backup`-ot, akkor csak átírnánk az *eredeti* fájlunkat. Bármit is használunk szövegként az `-i` paraméter után, az lesz az a név, amire az eredeti fájl el lesz mentve a `sed` által kért módosítások előtt.

Az adatok integritásának biztosítása

Bemutattuk, hogy milyen egyszerű a fájlok kezelése Linuxon. Előfordulhat, hogy egy fájlt másnak szeretnénk továbbítani, és biztosak akarunk lenni abban, hogy a címzett az eredeti fájl valódi másolatát kapja meg. Ennek a technikának egy nagyon gyakori felhasználási módja, amikor a Linux disztribúciós szerverek letölthető CD- vagy DVD-image-ket tárolnak a szoftverükről, olyan fájlokkal együtt, amelyek tartalmazzák a lemez image-k kiszámított checksum értékeit. Íme egy példa egy Debian letöltési tárhelyről:

```
                                                                    
[PARENTDIR] Parent Directory                                    -
[SUM]       MD5SUMS                          2019-09-08 17:46 274
[CRT]       MD5SUMS.sign                     2019-09-08 17:52 833
[SUM]       SHA1SUMS                         2019-09-08 17:46 306
[CRT]       SHA1SUMS.sign                    2019-09-08 17:52 833
[SUM]       SHA256SUMS                       2019-09-08 17:46 402
[CRT]       SHA256SUMS.sign                  2019-09-08 17:52 833
```

```
[SUM]     SHA512SUMS                              2019-09-08 17:46 658
[CRT]     SHA512SUMS.sign                         2019-09-08 17:52 833
[ISO]     debian-10.1.0-amd64-netinst.iso         2019-09-08 04:37 335M
[ISO]     debian-10.1.0-amd64-xfce-CD-1.iso       2019-09-08 04:38 641M
[ISO]     debian-edu-10.1.0-amd64-netinst.iso     2019-09-08 04:38 405M
[ISO]     debian-mac-10.1.0-amd64-netinst.iso     2019-09-08 04:38 334M
```

A fenti felsorolásban a Debian telepítő képfájlokat szöveges fájlok kísérik, amelyek a fájlok különböző algoritmusok (MD5, SHA1, SHA256 és SHA512) szerinti checksumait tartalmazzák.

NOTE

A checksum egy fájl ellenében egy kriptográfiai hash-függvényen alapuló matematikai számításból származó érték. A kriptográfiai hash-funkcióknak különböző típusai vannak, amelyek erőssége eltérő. A vizsgán elvárják, hogy ismerjük az `md5sum`, az `sha256sum` és az `sha512sum` használatát.

Miután letöltöttünk egy fájlt (például a `debian-10.1.0-amd64-netinst.iso` imaget), összehasonlíthatjuk a letöltött fájl checksumát a megadott checksummal.

Íme egy példa a lényeg illusztrálására. Az `ftu.txt` fájl SHA256 értékét a `sha256sum` paranccsal fogjuk kiszámítani:

```
$ sha256sum ftu.txt
345452304fc26999a715652543c352e5fc7ee0c1b9deac6f57542ec91daf261c  ftu.txt
```

A fájl neve előtt álló hosszú karaktersorozat a szöveges fájl SHA256 checksum értéke. Hozzunk létre egy olyan fájlt, amely tartalmazza ezt az értéket, hogy az eredeti szöveges fájlunk integritásának ellenőrzésére használhassuk. Ezt megtehetjük ugyanezzel az `sha256sum` paranccsal, és a kimenetet átirányíthatjuk egy fájlba:

```
$ sha256sum ftu.txt > sha256.txt
```

Most az `ftu.txt` fájl ellenőrzéséhez ugyanezt a parancsot használjuk, és megadjuk a checksum értékét tartalmazó fájlnevet a `-c` kapcsolóval együtt:

```
$ sha256sum -c sha256.txt
ftu.txt: OK
```

A fájlban található érték megegyezik az `ftu.txt` fájlunk kiszámított SHA256 ellenőrző összegével, ahogyan azt elvárnánk. Ha azonban az eredeti fájl módosulna (például néhány bájt elveszne egy

fájl letöltése során, vagy valaki szándékosan manipulálta volna), az értékellenőrzés sikertelen lenne. Ilyen esetekben tudjuk, hogy a fájlunk rossz vagy sérült, és nem bízhatunk a tartalmának integritásában. Hogy bizonyítsuk a dolgot, a fájl végére egy kis szöveget illesztünk:

```
$ echo "new entry" >> ftu.txt
```

Most megpróbáljuk ellenőrizni a fájl sértetlenségét:

```
$ sha256sum -c sha256.txt
ftu.txt: FAILED
sha256sum: WARNING: 1 computed checksum did NOT match
```

És látjuk, hogy a checksum nem egyezik a fájl elvárt értékével. Ezért nem bízhatunk a fájl integritásában. Megkísérelhetjük letölteni a fájl új példányát, jelenthetjük a checksum hibáját a fájl feladójának, vagy jelenthetjük az adatközpont biztonsági csapatának, a fájl fontosságától függően.

A fájlok mélyebb vizsgálata

Az octal dump (od) parancsot gyakran használják alkalmazások és különböző fájlok hibakeresésére. Önmagában az od parancs csak felsorolja egy fájl tartalmát oktális formátumban. A korábbi `ftu.txt` fájlunkat használhatjuk, hogy gyakoroljunk ezzel a paranccsal:

```
$ od ftu.txt
0000000 075142 060543 005164 060543 005164 072543 005164 062550
0000020 062141 066012 071545 005163 062155 071465 066565 067012
0000040 005154 062157 070012 071541 062564 071412 062145 071412
0000060 060550 032462 071466 066565 071412 060550 030465 071462
0000100 066565 071412 071157 005164 070163 064554 005164 060564
0000120 066151 072012 005162 067165 070551 073412 005143 075170
0000140 060543 005164 061572 072141 000012
0000151
```

A kimenet első oszlopa a *byte offset* a kimenet minden egyes sorához. Mivel az od alapértelmezés szerint oktális formátumban írja ki az információt, minden sor a nyolc bites bájteltolódással kezdődik, amelyet nyolc oszlop követ, amelyek mindegyike az adott oszlopon belüli adatok oktális értékét tartalmazza.

TIP Emlékezzünk arra, hogy egy *bájt* 8 bit hosszúságú!

Ha egy fájl tartalmát hexadecimális formátumban szeretnénk megtekinteni, használjuk az `-x` kapcsolót:

```
$ od -x ftu.txt
0000000 7a62 6163 0a74 6163 0a74 7563 0a74 6568
0000020 6461 6c0a 7365 0a73 646d 7335 6d75 6e0a
0000040 0a6c 646f 700a 7361 6574 730a 6465 730a
0000060 6168 3532 7336 6d75 730a 6168 3135 7332
0000100 6d75 730a 726f 0a74 7073 696c 0a74 6174
0000120 6c69 740a 0a72 6e75 7169 770a 0a63 7a78
0000140 6163 0a74 637a 7461 000a
0000151
```

Most a bájteltolást követő nyolc oszlop mindegyikét a hexadecimális megfelelőjükkel ábrázoljuk.

Az `od` parancs egyik praktikus felhasználási módja a szkriptek hibakeresése. Az `od` parancs például megmutathatja a fájlban létező, általában nem látható karaktereket, például a *newline* bejegyzéseket. Ezt a `-c` opcióval tehetjük meg, így az egyes bájtok numerikus jelölése helyett ezek az oszlopbejegyzések karakteres megfelelőiként jelennek meg:

```
$ od -c ftu.txt
0000000   b   z   c   a   t  \n   c   a   t  \n   c   u   t  \n   h   e
0000020   a   d  \n   l   e   s   s  \n   m   d   5   s   u   m  \n   n
0000040   l  \n   o   d  \n   p   a   s   t   e  \n   s   e   d  \n   s
0000060   h   a   2   5   6   s   u   m  \n   s   h   a   5   1   2   s
0000100   u   m  \n   s   o   r   t  \n   s   p   l   i   t  \n   t   a
0000120   i   l  \n   t   r  \n   u   n   i   q  \n   w   c  \n   x   z
0000140   c   a   t  \n   z   c   a   t  \n
0000151
```

A fájlon belüli összes újsoros bejegyzést a rejtett `\n` karakterek jelölik. Ha csak az összes karaktert szeretnénk megtekinteni a fájlban, és nem kell látnunk a bájteltolódási információt, a bájtolódás oszlopot a következőképpen távolíthatjuk el a kimenetből:

```
$ od -An -c ftu.txt
   b   z   c   a   t  \n   c   a   t  \n   c   u   t  \n   h   e
   a   d  \n   l   e   s   s  \n   m   d   5   s   u   m  \n   n
   l  \n   o   d  \n   p   a   s   t   e  \n   s   e   d  \n   s
   h   a   2   5   6   s   u   m  \n   s   h   a   5   1   2   s
   u   m  \n   s   o   r   t  \n   s   p   l   i   t  \n   t   a
   i   l  \n   t   r  \n   u   n   i   q  \n   w   c  \n   x   z
```

```
c a t \n z c a t \n
```

 | |

Gyakorló feladatok

1. Valaki most adományozott egy laptopot az iskolánknak, és most szeretnénk Linuxot telepíteni rá. Nincs kézikönyv és kénytelenek vagyunk egy USB pendrive-ról bootolni, ahol nincs semmilyen grafika. Kapunk egy shell terminált, és tudjuk, hogy minden processzornak lesz egy sora a `/proc/cpuinfo` fájlban:

```
processor   : 0
vendor_id   : GenuineIntel
cpu family  : 6
model       : 158

(kihagyott sorok)

processor   : 1
vendor_id   : GenuineIntel
cpu family  : 6
model       : 158

(még több kihagyott sor)
```

   - A `grep` és `wc` parancsok segítségével jelenítsük meg, hogy hány processzorral rendelkezünk!

   - Tegyük meg ugyanezt a `sed` használatával a `grep` helyett!

2. Vizsgáljuk meg a helyi `/etc/passwd` fájlt a `grep`, `sed`, `head` és `tail` parancsokkal az alábbi feladatok szerint:

   - Mely felhasználóknak van hozzáférése a Bash shellhez?

   - A rendszerünkben különböző felhasználók vannak, akik bizonyos programok kezelésére vagy adminisztrációs céllal léteznek. Nekik nincs hozzáférésük a shellhez. Hány ilyen felhasználó létezik a rendszerben?

   - Hány felhasználó és csoport létezik a rendszerben (ne feledjük: csak az `/etc/passwd` fájlt használjuk)?

- Csak az `/etc/passwd` fájl első, utolsó és tizedik sorát listázzuk ki!

3. Vegyük például az alábbi `/etc/passwd` fájlt. Másoljuk az alábbi sorokat egy `mypasswd` nevű helyi fájlba a gyakorláshoz!

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
nvidia-persistenced:x:121:128:NVIDIA Persistence Daemon,,,:/nonexistent:/sbin/nologin
libvirt-qemu:x:64055:130:Libvirt Qemu,,,:/var/lib/libvirt:/usr/sbin/nologin
libvirt-dnsmasq:x:122:133:Libvirt Dnsmasq,,,:/var/lib/libvirt/dnsmasq:/usr/sbin/nologin
carol:x:1000:2000:Carol Smith,Finance,,,Main Office:/home/carol:/bin/bash
dave:x:1001:1000:Dave Edwards,Finance,,,Main Office:/home/dave:/bin/ksh
emma:x:1002:1000:Emma Jones,Finance,,,Main Office:/home/emma:/bin/bash
frank:x:1003:1000:Frank Cassidy,Finance,,,Main Office:/home/frank:/bin/bash
grace:x:1004:1000:Grace Kearns,Engineering,,,Main Office:/home/grace:/bin/ksh
henry:x:1005:1000:Henry Adams,Sales,,,Main Office:/home/henry:/bin/bash
john:x:1006:1000:John Chapel,Sales,,,Main Office:/home/john:/bin/bash
```

- Listázzuk ki az összes felhasználót az `1000` csoportból (használjuk a `sed`-et a megfelelő mezőt kiválasztásához) a `mypasswd` fájlból!

- Listázzuk ki csak a felhasználók teljes nevét ebből a csoportból (a `sed` és `cut` használatával)!

Gondolkodtató feladatok

1. Ismét az előző feladatokban használt `mypasswd` fájl segítségével találjunk ki egy Bash-parancsot, amely kiválaszt egy személyt a Main Office-ból (Fő Irodából), aki megnyeri a tombolaversenyt. A `sed` paranccsal csak a Main Office sorait listázzuk ki, majd egy `cut` parancssorozattal nyerjük ki az egyes felhasználók keresztnevét ezekből a sorokból. Ezután ezeket a neveket rendezzük véletlenszerűen, és a listából csak a legfelső nevet írassuk ki!

2. Hányan dolgoznak a Finance (pénzügyi), a Engineering (mérnöki) és az Sales (értékesítési) területeken? (Fontoljuk meg a `uniq` parancs használatát!)

3. Most egy CSV (comma separeted values - vesszővel elválasztott értékek) fájlt kell készítenünk, hogy az előző példában szereplő `mypasswd` fájlból könnyen importálhassuk a `names.csv` fájlt a LibreOffice-ba. A fájl tartalma a következő formátumú lesz:

   ```
   First Name,Last Name,Position
   Carol,Smith,Finance
   ...
   John,Chapel,Sales
   ```

   Tipp: Használjuk a `sed`, `cut`, és `paste` parancsokat az elvárt eredmények eléréséhez! Vegyük figyelembe, hogy a vessző (`,`) lesz a delimiter (elválasztó) ennél a fájlnál!

4. Tegyük fel, hogy az előző feladatban létrehozott `names.csv` táblázat egy fontos fájl, és szeretnénk biztosítani, hogy senki ne piszkálhassa attól a pillanattól kezdve, hogy elküldjük valakinek, és attól a pillanattól kezdve, hogy a címzett megkapja. Hogyan biztosíthatjuk ennek a fájlnak az integritását az `md5sum` segítségével?

5. Eldöntöttük, hogy naponta 100 sort olvasunk egy klasszikus könyvből, és úgy döntöttünk, hogy Herman Melville *Mariner and Mystic* című művével kezdjük. Találjunk ki egy olyan parancsot a `split` segítségével, amely ezt a könyvet 100 soros szakaszokra osztja. Ahhoz, hogy a könyvet egyszerű szöveges formátumban megkapjuk, keressük azt meg a https://www.gutenberg.org címen.

6. Az `ls -l` használatával a `/etc` mappában milyen listát kapunk? A `cut` parancsot használva az adott `ls` parancs kimenetén hogyan jeleníthetnénk meg csak a fájlneveket? Mi a helyzet a

fájlnevekkel és a fájlok tulajdonosával? Az `ls -l` és a `cut` parancsok mellett használjuk a `tr` parancsot a szóköz többszörös előfordulásának egyetlen szóközzé való *összenyomására* (squeeze), hogy segítsük a kimenet formázását a `cut` paranccsal!

7. Ez a gyakorlat azt feltételezi, hogy egy valódi gépet (nem virtuális gépet) használunk. Egy USB-pendrive is legyen nálunk. Nézzük át a `tail` parancs kézikönyvének oldalait, és találjuk meg, hogyan követhetünk egy fájlt, amihez szöveg hozzá van csatolva. Miközben a `tail` parancs kimenetét figyeljük a `/var/log/syslog` fájlban, helyezzünk be egy USB-pendrive-ot. Írjuk ki a teljes parancsot, amellyel az USB-stick terméknevét, gyártóját és teljes memóriamennyiségét szeretnénk megtudni!

Összefoglalás

A szövegfolyamok kezelése nagy jelentőséggel bír bármely Linux rendszer adminisztrálásakor. A szövegfolyamokat szkriptek segítségével lehet feldolgozni a napi feladatok automatizálása vagy a naplófájlokban található releváns hibakeresési információk megtalálása érdekében. Az alábbiakban röviden összefoglaljuk a leckében tárgyalt parancsokat:

`cat`

Sima szöveges fájlok kombinálására vagy olvasására szolgál.

`bzcat`

Lehetővé teszi a `bzip2` módszerrel tömörített fájlok feldolgozását vagy olvasását.

`xzcat`

Lehetővé teszi az `xz` módszerrel tömörített fájlok feldolgozását vagy olvasását.

`zcat`

Lehetővé teszi a `gzip` módszerrel tömörített fájlok feldolgozását vagy olvasását.

`less`

Ez a parancs lapozhatóvá teszi a fájl tartalmát, és lehetővé teszi a navigációt és a keresést.

`head`

Ez a parancs alapértelmezés szerint a fájl első 10 sorát jeleníti meg. Az `-n` kapcsoló használatával kevesebb vagy több sor is megjeleníthető.

`tail`

Ez a parancs alapértelmezés szerint a fájl utolsó 10 sorát jeleníti meg. Az `-n` kapcsoló használatával kevesebb vagy több sor is megjeleníthető. Az `-f` kapcsoló arra szolgál, hogy kövessük a szöveges fájl kimenetét, amikor új adatok íródnak bele.

`wc`

A “word count” rövidítése, de a használt paraméterektől függően karaktereket, szavakat és sorokat is meg tud számolni.

`sort`

A kimenetet ábécé, fordított ábécé vagy véletlenszerű sorrendbe rendezi.

`uniq`

Az egyező sztringek listázásra (és számolására) szolgál.

`od`

Az “octal dump” parancs egy bináris fájl megjelenítésére szolgál oktális, decimális vagy hexadecimális jelöléssel.

`nl`

A “number line” parancs megjeleníti a fájlban lévő sorok számát, valamint újra létrehozza a fájlt úgy, hogy minden egyes sor elé a odaírja a sorszámát.

`sed`

A stream editor (folyamszerkesztő) használható sztringek előfordulásának keresésére reguláris kifejezések segítségével, valamint fájlok szerkesztésére előre definiált minták alapján.

`tr`

A translate parancs képes karaktereket cserélni, valamint eltávolítja és tömöríti az ismétlődő karaktereket.

`cut`

Ez a parancs ki tudja írni a szövegfájlok oszlopait mezőként a fájl karakteres elválasztójelén (delimiter) alapuló mezőként.

`paste`

Fájlok összekapcsolása oszlopokban a mezőelválasztók használata alapján.

`split`

Ez a parancs a nagyobb fájlokat kisebbekre darabolhatja a parancs beállításai által meghatározott feltételektől függően.

`md5sum`

Egy fájl MD5 hash-értékének kiszámítására szolgál. A fájl integritásának biztosítása érdekében egy fájl meglévő hash-értékkel való ellenőrzésére is használható.

`sha256sum`

Egy fájl SHA256 hash-értékének kiszámítására szolgál. A fájl integritásának biztosítása érdekében egy fájl meglévő hash-értékkel való ellenőrzésére is használható.

`sha512sum`

Egy fájl SHA512 hash-értékének kiszámítására szolgál. A fájl integritásának biztosítása érdekében egy fájl meglévő hash-értékkel való ellenőrzésére is használható.

Válaszok a gyakorló feladatokra

1. Valaki most adományozott egy laptopot az iskolánknak, és most szeretnénk Linuxot telepíteni rá. Nincs kézikönyv és kénytelenek vagyunk egy USB pendrive-ról bootolni, ahol nincs semmilyen grafika. Kapunk egy shell terminált, és tudjuk, hogy minden processzornak lesz egy sora a `/proc/cpuinfo` fájlban:

```
processor   : 0
vendor_id   : GenuineIntel
cpu family  : 6
model       : 158

(kihagyott sorok)

processor   : 1
vendor_id   : GenuineIntel
cpu family  : 6
model       : 158

(még több kihagyott sor)
```

   - A `grep` és `wc` parancsok segítségével jelenítsük meg, hogy hány processzorral rendelkezünk!

     Íme a két lehetőség:

```
$ cat /proc/cpuinfo | grep processor | wc -l
$ grep processor /proc/cpuinfo | wc -l
```

     Most, hogy tudjuk, hogy többféleképpen is megtehetjük ugyanazt a dolgot, mikor érdemes az egyiket vagy a másikat használni? Ez valójában több tényezőtől függ, a két legfontosabb a teljesítmény és az olvashatóság. Legtöbbször shell parancsokat fogunk használni shell szkripteken belül a feladatok automatizálására, és minél nagyobbak és összetettebbek lesznek a szkriptek, annál jobban kell aggódni a gyorsaságuk miatt.

   - Tegyük meg ugyanezt a `sed` használatával a `grep` helyett!

     Most a `grep` helyett nézzük a `sed`-et:

```
$ sed -n /processor/p /proc/cpuinfo | wc -l
```

Itt a `sed`-et a `-n` paraméterrel használtuk, így a `sed` nem fog kiírni semmit, kivéve azt, ami a `processor` kifejezéssel egyezik, ahogyan azt az `p` parancs utasítja. Ahogy a `grep` megoldásoknál is tettük, a `wc -l` megszámolja a sorok számát, tehát a processzorok számát.

Nézzük a következő példát:

```
$ sed -n /processor/p /proc/cpuinfo | sed -n '$='
```

Ez a parancssorozat azonos eredményt ad, mint az előző példa, ahol a `sed` kimenetét a `wc` parancsba továbbítottuk. A különbség itt az, hogy ahelyett, hogy a `wc -l` parancsot használnánk a sorok számolásához, ismét a `sed` parancsot hívjuk meg, hogy ezzel egyenértékű funkciót biztosítsunk. Ismét elnyomjuk a `sed` kimenetét a `-n` opcióval, kivéve azt a kifejezést, amelyet explicit meghívunk, ami a `'$='`. Ez a parancs arra utasítja a sed-et, hogy keresse meg az utolsó sort ($), majd jelenítse meg a hozzá tartozó sorszámot (=).

2. Vizsgáljuk meg a helyi `/etc/passwd` fájlt a `grep`, `sed`, `head` és `tail` parancsokkal az alábbi feladatok szerint:
   - Mely felhasználóknak van hozzáférése a Bash shellhez?

     ```
     $ grep ":/bin/bash$" /etc/passwd
     ```

     Ezt a választ úgy fogjuk továbbfejleszteni, hogy csak annak a felhasználónak a nevét jelenítjük meg, aki a Bash shell-t használja.

     ```
     $ grep ":/bin/bash$" /etc/passwd | cut -d: -f1
     ```

     A felhasználónév az első mező (a `cut` parancs `-f1` paramétere) és az `/etc/passwd` fájl `:`-t használ elválasztóként (a `cut` parancs `-d:` paramétere) a `grep` parancs kimenetét egyszerűen átvezetjük a megfelelő `cut` parancsba.

   - A rendszerünkben különböző felhasználók vannak, akik bizonyos programok kezelésére vagy adminisztrációs céllal léteznek. Nekik nincs hozzáférésük a shellhez. Hány ilyen felhasználó létezik a rendszerben?

     Ezt a legegyszerűbben úgy tudhatjuk meg, hogy ha kiíratjuk azokat a fiókokat, amelyek nem használják a Bash shellt:

     ```
     $ grep -v ":/bin/bash$" /etc/passwd | wc -l
     ```

Hány felhasználó és csoport létezik a rendszerben (ne feledjük: csak az `/etc/passwd` fájlt használjuk)?

Az `/etc/passwd` fájl bármelyik sorának első mezője a felhasználónév, a második mező általában egy `x`, ami azt jelzi, hogy a felhasználói jelszó nem itt van tárolva (az `/etc/shadow` fájlban van titkosítva). A harmadik a felhasználói azonosító (UID), a negyedik pedig a csoport azonosítója (GID). Ez tehát meg kell, hogy adja a felhasználók számát:

```
$ cut -d: -f3 /etc/passwd | wc -l
```

Nos, az esetek többségében igen. Vannak azonban olyan helyzetek, amikor különböző szuperfelhasználókat vagy más speciális típusú felhasználókat állítunk be, akik ugyanazt az UID-t (felhasználói azonosítót) használják. Tehát a biztonság kedvéért a `cut` parancsunk eredményét átvezetjük a `sort` parancsba, majd megszámoljuk a sorok számát.

```
$ cut -d: -f3 /etc/passwd | sort -u | wc -l
```

Most a csoportok száma:

```
$ cut -d: -f4 /etc/passwd | sort -u | wc -l
```

- Csak az `/etc/passwd` fájl első, utolsó és tizedik sorát listázzuk ki!

  Ez megoldja:

  ```
  $ sed -n -e '1'p -e '10'p -e '$'p /etc/passwd
  ```

  Ne feledjük, hogy a `-n` paraméter azt mondja a `sed`-nek, hogy ne írjon ki semmi mást, mint amit az `p` parancs megad! Az itt használt dollárjel (`$`) egy reguláris kifejezés, amely a fájl utolsó sorát jelenti.

3. Vegyük például az alábbi `/etc/passwd` fájlt. Másoljuk az alábbi sorokat egy `mypasswd` nevű helyi fájlba a gyakorláshoz!

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
```

```
nvidia-persistenced:x:121:128:NVIDIA Persistence Daemon,,,:/nonexistent:/sbin/nologin
libvirt-qemu:x:64055:130:Libvirt Qemu,,,:/var/lib/libvirt:/usr/sbin/nologin
libvirt-dnsmasq:x:122:133:Libvirt Dnsmasq,,,:/var/lib/libvirt/dnsmasq:/usr/sbin/nologin
carol:x:1000:2000:Carol Smith,Finance,,,Main Office:/home/carol:/bin/bash
dave:x:1001:1000:Dave Edwards,Finance,,,Main Office:/home/dave:/bin/ksh
emma:x:1002:1000:Emma Jones,Finance,,,Main Office:/home/emma:/bin/bash
frank:x:1003:1000:Frank Cassidy,Finance,,,Main Office:/home/frank:/bin/bash
grace:x:1004:1000:Grace Kearns,Engineering,,,Main Office:/home/grace:/bin/ksh
henry:x:1005:1000:Henry Adams,Sales,,,Main Office:/home/henry:/bin/bash
john:x:1006:1000:John Chapel,Sales,,,Main Office:/home/john:/bin/bash
```

- Listázzuk ki az összes felhasználót az `1000` csoportból (használjuk a `sed`-et a megfelelő mezőt kiválasztásához) a `mypasswd` fájlból!

  A GID a negyedik mező az `/etc/passwd` fájlban. Lehet, hogy megpróbáljuk ezt:

  ```
  $ sed -n /1000/p mypasswd
  ```

  Ebben az esetben ezt a sort fogjuk kapni:

  ```
  carol:x:1000:2000:Carol Smith,Finance,,,Main Office:/home/carol:/bin/bash
  ```

  Tudjuk, hogy ez nem helyes, mivel Carol Smith a GID 2000 tagja, és az egyezés az UID miatt történt. Azonban észrevehettük, hogy a GID után a következő mező nagybetűvel kezdődik. Ezt a problémát egy reguláris kifejezéssel tudjuk megoldani.

  ```
  $ sed -n /:1000:[A-Z]/p mypasswd
  ```

  A `[A-Z]` kifejezés bármelyik nagybetűs karakterrel megegyezik. Erről többet fogunk megtudni a megfelelő leckében.

- Listázzuk ki csak a felhasználók teljes nevét ebből a csoportból (a `sed` és `cut` használatával)!

  Használjuk ugyanazt a technikát, amit a feladat első részének megoldásához használtunk és vezessük át a `cut` parancsba!

  ```
  $ sed -n /:1000:[A-Z]/p mypasswd | cut -d: -f5
  Dave Edwards,Finance,,,Main Office
  Emma Jones,Finance,,,Main Office
  Frank Cassidy,Finance,,,Main Office
  ```

```
Grace Kearns,Engineering,,,Main Office
Henry Adams,Sales,,,Main Office
John Chapel,Sales,,,Main Office
```

Nem egészen ez még a megoldás! Figyeljük meg, hogy a mezők az eredményeken belül `,`-vel elválaszthatók. Ezért a kimenetet egy másik `cut` parancsba fogjuk átvezetni, a `,`-t használva delimiterként.

```
$ sed -n /:1000:[A-Z]/p mypasswd | cut -d: -f5 | cut -d, -f1
Dave Edwards
Emma Jones
Frank Cassidy
Grace Kearns
Henry Adams
John Chapel
```

Válaszok a gondolkodtató feladatokra

1. Ismét az előző feladatokban használt `mypasswd` fájl segítségével találjunk ki egy Bash-parancsot, amely kiválaszt egy személyt a Main Office-ból (Fő Irodából), aki megnyeri a tombolaversenyt. A `sed` paranccsal csak a Main Office sorait listázzuk ki, majd egy `cut` parancssorozattal nyerjük ki az egyes felhasználók keresztnevét ezekből a sorokból. Ezután ezeket a neveket rendezzük véletlenszerűen, és a listából csak a legfelső nevet írassuk ki!

   Először vizsgáljuk meg, hogy a `-R` paraméter hogyan módosítja a `sort` parancs kimenetét. Ismételjük meg ezt a parancsot néhányszor (vegyük figyelembe, hogy a 'Main Office'-t aposztrófok közé kell zárnunk, így a `sed` egyetlen sztringként kezelje):

   ```
   $ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d, -f1 | sort -R
   ```

   Íme a megoldás a problémára:

   ```
   $ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d, -f1 | sort -R | head -1
   ```

2. Hányan dolgoznak a Finance (pénzügyi), a Engineering (mérnöki) és az Sales (értékesítési) területeken? (Fontoljuk meg a `uniq` parancs használatát!)

   Folytassa a korábbi gyakorlatok során tanultakra építve. Próbálja ki a következőket:

   ```
   $ sed -n /'Main Office'/p mypasswd
   $ sed -n /'Main Office'/p mypasswd | cut -d, -f2
   ```

   Vegyük észre, hogy most már nem törődünk a `:`-al, mint delimiterrel. Nekünk csak a második mező kell, amikor a `,` karakterekkel választjuk el a sorokat.

   ```
   $ sed -n /'Main Office'/p mypasswd | cut -d, -f2 | uniq -c
         4 Finance
         1 Engineering
         2 Sales
   ```

   A `uniq` parancs csak az egyedi sorokat adja ki (az ismétlődő sorokat nem), a `-c` paraméter pedig arra utasítja az `uniq` parancsot, hogy számolja az egyenlő sorok előfordulásait. Van itt egy figyelmeztetés: A `uniq` csak a szomszédos sorokat veszi figyelembe. Ha ez nem megfelelő, akkor a `sort` parancsot kell használni.

3. Most egy CSV (comma separeted values - vesszővel elválasztott értékek) fájlt kell készítenünk, hogy az előző példában szereplő `mypasswd` fájlból könnyen importálhassuk a `names.csv` fájlt a LibreOffice-ba. A fájl tartalma a következő formátumú lesz:

```
First Name,Last Name,Position
Carol,Smith,Finance
...
John,Chapel,Sales
```

Tipp: Használjuk a `sed`, `cut`, és `paste` parancsokat az elvárt eredmények eléréséhez! Vegyük figyelembe, hogy a vessző (`,`) lesz a delimiter ennél a fájlnál!

Kezdjük a `sed` és `cut` parancsokkal, az előző gyakorlatokban tanultakra építve:

```
$ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d" " -f1 > firstname
```

Most már megvan a `firstname` fájl az alkalmazottaink keresztneveivel.

```
$ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d" " -f2 | cut -d, -f1 > lastname
```

Most megvan a `lastname` fájl az alkalmazottak vezetékneveivel.

Ezután meghatározzuk, hogy az egyes alkalmazottak melyik osztályon dolgoznak:

```
$ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d, -f2 > department
```

Mielőtt a végleges megoldáson dolgoznánk, próbáljuk ki a következő parancsokat, hogy lássuk, milyen kimenetet generálnak:

```
$ cat firstname lastname department
$ paste firstname lastname department
```

És most a végleges megoldás:

```
$ paste firstname lastname department | tr '\t' ,
$ paste firstname lastname department | tr '\t' , > names.csv
```

Itt a `tr` parancsot használjuk a `\t`, a tabulátor elválasztójel `,`-vel történő *cseréjére* (translate). A

`tr` nagyon hasznos, amikor egy karaktert egy másikra kell cserélnünk. Mindenképpen olvassuk el a `tr` és a `paste` man oldalakat! Például használhatjuk a `-d` opciót az elválasztójelhez, hogy az előző parancsot kevésbé bonyolulttá tegyük:

```
$ paste -d, firstname lastname department
```

A `paste` parancsot itt egyszer használtuk, hogy megismerkedjünk vele. Azonban könnyen elvégezhettük volna az összes feladatot egyetlen parancslánccal is:

```
$ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d, -f1,2 | tr ' ' , > names.csv
```

4. Tegyük fel, hogy az előző feladatban létrehozott `names.csv` táblázat egy fontos fájl, és szeretnénk biztosítani, hogy senki ne piszkálhassa attól a pillanattól kezdve, hogy elküldjük valakinek, és attól a pillanattól kezdve, hogy a címzett megkapja. Hogyan biztosíthatjuk ennek a fájlnak az integritását az `md5sum` segítségével?

   Ha megnézzük az `md5sum`, `sha256sum` és `sha512sum` man oldalait, azt fogjuk látni, hogy mindegyik az alábbi szöveggel kezdődik:

   "`compute and check XXX message digest` (XXX üzenetkivonatoló kiszámítása és ellenőrzése) "

   Ahol “XXX” az az algoritmus, amelyet az *üzenetkivonatoló* (message digest) létrehozására használnak.

   Az `md5sum`-ot fogjuk példaként használni, később pedig kipróbálhatjuk a többi parancsot is!

```
$ md5sum names.csv
61f0251fcab61d9575b1d0cbf0195e25  names.csv
```

   Most például elérhetővé tehetjük a fájlt egy biztonságos ftp szolgáltatáson keresztül, és a generált *message digest* üzenetet egy másik biztonságos kommunikációs eszközzel küldhetjük el. Ha a fájlt kissé módosították, a *message digest* teljesen más lesz. Csak a bizonyítás kedvéért, szerkesszük meg a `names.csv` állományt, és változtassuk meg Jones-t James-re, ahogy itt látható:

```
$ sed -i.backup s/Jones/James/ names.csv
$ md5sum names.csv
f44a0d68cb480466099021bf6d6d2e65  names.csv
```

Amikor fájlokat teszünk elérhetővé letöltésre, mindig jó gyakorlat, ha egy *message digest*-et is terjesztünk, hogy azok, akik letöltik a fájlt, új *message digestet* készíthessenek, és összevethessék az eredetivel. Ha átböngésszük a https://kernel.org oldalt, megtaláljuk a https://mirrors.edge.kernel.org/pub/linux/kernel/v5.x/sha256sums.asc oldalt, ahol a letölthető fájlok sha256sum-ot kaphatjuk meg.

5. Eldöntöttük, hogy naponta 100 sort olvasunk egy klasszikus könyvből, és úgy döntöttünk, hogy Herman Melville *Mariner and Mystic* című művével kezdjük. Találjunk ki egy olyan parancsot a `split` segítségével, amely ezt a könyvet 100 soros szakaszokra osztja. Ahhoz, hogy a könyvet egyszerű szöveges formátumban megkapjuk, keressük azt meg a https://www.gutenberg.org címen.

   Először a teljes könyvet a Project Gutenberg oldalról fogjuk megszerezni, ahol ez és más, közkinccsé tett könyvek is elérhetőek.

   ```
   $ wget https://www.gutenberg.org/files/50461/50461-0.txt
   ```

   Lehet, hogy telepítenünk kell a `wget`-et, ha még nincs feltelepítve. Alternatívaként használhatjuk a `curl`-t. A könyv ellenőrzéséhez használhatjuk a `less`-t:

   ```
   $ less 50461-0.txt
   ```

   Most a könyvet egyenként 100 soros részekre osztjuk:

   ```
   $ split -l 100 -d 50461-0.txt melville
   ```

   `50461-0.txt` a fájl, amit feldarabolunk. A `melville` a töredékfájlok prefixe. A `-l 100` határozza meg a sorok számát és a `-d` kapcsoló mondja meg a `split`-nek, hogy számozza a fájlokat (a megadott suffix használatával). Használhatjuk az `nl`-t bármelyik töredékfájlon (lehetőleg ne a legutolsón), hogy megerősítsük, hogy mindegyik 100 soros.

6. Az `ls -l` használatával a `/etc` mappában milyen listát kapunk? A `cut` parancsot használva az adott `ls` parancs kimenetén hogyan jeleníthetnénk meg csak a fájlneveket? Mi a helyzet a fájlnevekkel és a fájlok tulajdonosával? Az `ls -l` és a `cut` parancsok mellett használjuk a `tr` parancsot a szóköz többszörös előfordulásának egyetlen szóközzé való *összenyomására* (squeeze), hogy segítsük a kimenet formázását a `cut` paranccsal!

   Az `ls` parancs önmagában csak a fájlok nevét adja meg. Az `ls -l` (a hosszú listázás, long list) kimenetét azonban előkészíthetjük a pontosabb információk kinyeréséhez.

```
$ ls -l /etc | tr -s ' ' ,
drwxr-xr-x,3,root,root,4096,out,24,16:58,acpi
-rw-r--r--,1,root,root,3028,dez,17,2018,adduser.conf
-rw-r--r--,1,root,root,10,out,2,17:38,adjtime
drwxr-xr-x,2,root,root,12288,out,31,09:40,alternatives
-rw-r--r--,1,root,root,401,mai,29,2017,anacrontab
-rw-r--r--,1,root,root,433,out,1,2017,apg.conf
drwxr-xr-x,6,root,root,4096,dez,17,2018,apm
drwxr-xr-x,3,root,root,4096,out,24,16:58,apparmor
drwxr-xr-x,9,root,root,4096,nov,6,20:20,apparmor.d
```

Az `-s` paraméter utasítja az `tr`-t, hogy az ismétlődő szóközöket egyetlen szóközpéldánnyá zsugorítsa. A `tr` parancs bármilyen megadott ismétlődő karakterrel működik. Ezután a szóközöket vesszővel `,` helyettesítjük. Példánkban valójában nincs szükségünk a szóközök helyettesítésére, ezért a `,`-t egyszerűen elhagyjuk.

```
$ ls -l /etc | tr -s ' '
drwxr-xr-x 3 root root 4096 out 24 16:58 acpi
-rw-r--r-- 1 root root 3028 dez 17 2018 adduser.conf
-rw-r--r-- 1 root root 10 out 2 17:38 adjtime
drwxr-xr-x 2 root root 12288 out 31 09:40 alternatives
-rw-r--r-- 1 root root 401 mai 29 2017 anacrontab
-rw-r--r-- 1 root root 433 out 1 2017 apg.conf
drwxr-xr-x 6 root root 4096 dez 17 2018 apm
drwxr-xr-x 3 root root 4096 out 24 16:58 apparmor
```

Ha csak a fájlneveket akarjuk, akkor csak a kilencedik mezőt kell megjeleníteni:

```
$ ls -l /etc | tr -s ' ' | cut -d" " -f9
```

A fájl nevéhez és tulajdonosához a kilencedik és a harmadik mezőre lesz szükségünk:

```
$ ls -l /etc | tr -s ' ' | cut -d" " -f9,3
```

Mi van, ha csak a mappák nevére és tulajdonosára van szükségünk?

```
$ ls -l /etc | grep ^d | tr -s ' ' | cut -d" " -f9,3
```

7. Ez a gyakorlat azt feltételezi, hogy egy valódi gépet (nem virtuális gépet) használunk. Egy USB-

pendrive is legyen nálunk. Nézzük át a `tail` parancs kézikönyvének oldalait, és találjuk meg, hogyan követhetünk egy fájlt, amihez szöveg hozzá van csatolva. Miközben a `tail` parancs kimenetét figyeljük a `/var/log/syslog` fájlban, helyezzünk be egy USB-pendrive-ot. Írjuk ki a teljes parancsot, amellyel az USB-stick terméknevét, gyártóját és teljes memóriamennyiségét szeretnénk megtudni!

```
$ tail -f /var/log/syslog | grep -i 'product\:\|blocks\|manufacturer'
Nov  8 06:01:35 brod-avell kernel: [124954.369361] usb 1-4.3: Product: Cruzer Blade
Nov  8 06:01:35 brod-avell kernel: [124954.369364] usb 1-4.3: Manufacturer: SanDisk
Nov  8 06:01:37 brod-avell kernel: [124955.419267] sd 2:0:0:0: [sdc] 61056064 512-byte
logical blocks: (31.3 GB/29.1 GiB)
```

Természetesen ez csak egy példa, és az eredmények az USB-pendrive gyártójától függően változhatnak. Vegyük észre, hogy most az `-i` paramétert használjuk a `grep` paranccsal, mivel nem vagyunk biztosak abban, hogy a keresett sztringek kis- vagy nagybetűsek voltak-e. A `|` logikai VAGY-ot is használtuk, így a `product` VAGY `blocks` VAGY `manufacturer` tartalmú sorokat keressük.

103.3 Alapvető fájl-menedzsment

Hivatkozás az LPI célkitűzésre

LPIC-1 v5, Exam 101, Objective 103.3

Súlyozás

4

Kulcsfontosságú ismeretek

- Fájlok és mappák egyenként történő másolása, áthelyezése és eltávolítása.
- Több fájl és mappa rekurzív másolása.
- Fájlok és mappák rekurzív eltávolítása.
- Egyszerű és bővített wildcard-meghatározások használata a parancsokban.
- A keresés használata fájlok típus, méret vagy idő alapján történő megkeresésére és kezelésére.
- A tar, cpio és dd használata.

A használt fájlok, kifejezések és segédprogramok listája

- `cp`
- `find`
- `mkdir`
- `mv`
- `ls`
- `rm`
- `rmdir`
- `touch`

- `tar`
- `cpio`
- `dd`
- `file`
- `gzip`
- `gunzip`
- `bzip2`
- `bunzip2`
- file globbing

103.3 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	103 GNU és Unix parancsok
Fejezet:	103.3 Alapvető fájl-menedzsment
Lecke:	1 of 2

Bevezetés

A Linuxban minden fájl, ezért nagyon fontos, hogy tudjuk, hogyan manipuláljuk őket. Ebben a leckében a fájlokon végezhető alapvető műveletekkel foglalkozunk.

Linux-felhasználóként általában a fájlrendszerben való navigálás, fájlok másolása egyik helyről a másikra és fájlok törlése a feladat. A fájlkezeléssel kapcsolatos parancsokkal is foglalkozni fogunk.

A fájl egy olyan egység, amely adatokat és programokat tárol. Tartalomból és metaadatokból áll (fájlméret, tulajdonos, létrehozás dátuma, jogosultságok). A fájlok mappákba vannak rendezve. A mappa egy olyan fájl, amely más fájlokat tárol.

A különböző fájltípusokba beletartoznak az alábbiak:

Normál fájlok
: amelyek adatokat és programokat tárolnak.

Mappák
: amelyek más fájlokat tartalmaznak.

Speciális fájlok

amelyeket bemenetre és kimenetre használnak.

Természetesen léteznek más típusú fájlok is, de ezek nem tartoznak e lecke tárgykörébe. Később tárgyalni fogjuk, hogyan lehet ezeket a különböző fájltípusokat azonosítani.

Fájlok manipulálása

Az `ls` használata a fájlok listázáshoz

Az `ls` parancs az egyik legfontosabb parancssori eszköz, amelyet meg kell tanulnunk a fájlrendszerben való navigáláshoz.

Alapvető formájában az `ls` *csak* fájl- és mappaneveket listáz:

```
$ ls
Desktop Downloads   emp_salary  file1   Music   Public  Videos
Documents   emp_name    examples.desktop    file2   Pictures    Templates
```

Az `-l`, azaz a `long listing` (hosszú listázás) formátummal együtt használva megmutatja a fájl vagy mappa jogosultságait, tulajdonosát, méretét, a módosítás dátumát, idejét és nevét:

```
$ ls -l
total 60
drwxr-xr-x  2   frank   frank   4096    Apr 1   2018    Desktop
drwxr-xr-x  2   frank   frank   4096    Apr 1   2018    Documents
drwxr-xr-x  2   frank   frank   4096    Apr 1   2018    Downloads
-rw-r--r--  1   frank   frank     21    Sep 7   12:59   emp_name
-rw-r--r--  1   frank   frank     20    Sep 7   13:03   emp_salary
-rw-r--r--  1   frank   frank   8980    Apr 1   2018    examples.desktop
-rw-r--r--  1   frank   frank     10    Sep 1   2018    file1
-rw-r--r--  1   frank   frank     10    Sep 1   2018    file2
drwxr-xr-x  2   frank   frank   4096    Apr 1   2018    Music
drwxr-xr-x  2   frank   frank   4096    Apr 1   2018    Pictures
drwxr-xr-x  2   frank   frank   4096    Apr 1   2018    Public
drwxr-xr-x  2   frank   frank   4096    Apr 1   2018    Templates
drwxr-xr-x  2   frank   frank   4096    Apr 1   2018    Videos
```

A kimenet első karaktere a fájl típusát jelzi:

-

egy normál fájl esetén.

d

egy könyvtár esetén.

c

egy speciális fájl esetén.

A fájlméretek ember által olvasható formátumban történő megjelenítéséhez adja hozzá a `-h` opciót:

```
$ ls -lh
total 60K
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Desktop
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Documents
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Downloads
-rw-r--r--  1   frank   frank     21    Sep 7   12:59   emp_name
-rw-r--r--  1   frank   frank     20    Sep 7   13:03   emp_salary
-rw-r--r--  1   frank   frank   8.8K    Apr 1   2018    examples.desktop
-rw-r--r--  1   frank   frank     10    Sep 1   2018    file1
-rw-r--r--  1   frank   frank     10    Sep 1   2018    file2
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Music
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Pictures
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Public
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Templates
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Videos
```

Az összes fájl listázásához, beleértve a rejtett fájlokat is (a `.`-al kezdődő fájlokat), használjuk a `-a` kapcsolót:

```
$ ls -a
.               .dbus   file1   .profile
..              Desktop file2   Public
.bash_history   .dmrc   .gconf  .sudo_as_admin_successful
```

Az alapértelmezés szerint rejtett konfigurációs fájlok, mint például a `.bash_history`, mostantól láthatóak.

Általában az `ls` parancs szintaxisa a következő:

```
ls OPTIONS FILE
```

Ahol az `OPTIONS` a korábban bemutatott opciók bármelyike (az összes lehetséges opció megtekintéséhez futtassa a `man ls` parancsot), a `FILE` pedig a listázni kívánt fájl vagy mappa neve.

NOTE Ha a `FILE` nincs megadva, akkor az aktuális mappát használja.

Fájlok létrehozása, másolása, áthelyezése és törlése

Fájlok létrehozása a `touch`-al

A `touch` parancs a legegyszerűbb módja az új, üres fájlok létrehozásának. Használhatjuk a meglévő fájlok és mappák időbélyegzőinek (azaz módosítási idejének) módosítására is. A `touch` parancs használatának szintaxisa a következő:

```
touch OPTIONS FILE_NAME(S)
```

Opciók nélkül a `touch` új fájlokat hoz létre az argumentumként megadott fájlnevekhez, feltéve, hogy ilyen nevű fájlok még nem léteznek. A `touch` tetszőleges számú fájlt hozhat létre egyszerre:

```
$ touch file1 file2 file3
```

Ez három üres fájlt hoz létre `file1`, `file2` és `file3` néven.

Számos `touch` opció kifejezetten arra szolgál, hogy a felhasználó megváltoztathassa a fájlok időbélyegzőit. Például az `-a` opció csak a hozzáférési időt változtatja meg, míg az `-m` opció csak a módosítási időt. A két opció együttes használata a hozzáférési és a módosítási időt is az aktuális időpontra változtatja:

```
$ touch -am file3
```

Fájlok másolása a `cp`-vel

Linux-felhasználóként gyakran másolunk fájlokat egyik helyről a másikra. Akár egy zenei fájl egyik mappából a másikba, akár egy rendszerfájl átmásolásáról van szó, minden másolási feladathoz használjuk a `cp` parancsot:

```
$ cp file1 dir2
```

Ez a parancs szó szerint úgy értelmezhető, hogy a `file1` másolása a `dir2` mappába. Az eredmény a `file1` fájl jelenléte a `dir2` mappában. A parancs sikeres végrehajtásához a `file1`-nek léteznie kell a felhasználó aktuális mappájában. Ellenkező esetben a rendszer hibát jelez a `No such file or directory` (Nem létezik ilyen fájl vagy könyvtár) üzenettel.

```
$ cp dir1/file1 dir2
```

Ebben az esetben figyeljük meg, hogy a `file1` elérési útvonala egyértelműbb. A forrás elérési útja kifejezhető *relatív* vagy *abszolút* útként. A relatív elérési utakat egy adott mappára való hivatkozással adjuk meg, míg az abszolút elérési utakat nem adjuk meg hivatkozással. Az alábbiakban ezt a fogalmat tovább tisztázzuk.

Egyelőre csak azt figyeljük meg, hogy ez a parancs a `file1`-et a `dir2` mappába másolja. A `file1` elérési útvonalát részletesebben adjuk meg, mivel a felhasználó jelenleg nem a `dir1` mappában van.

```
$ cp /home/frank/Documents/file2 /home/frank/Documents/Backup
```

Ebben a harmadik esetben a `/home/frank/Documents` mappában található `file2` fájlt a `/home/frank/Documents/Backup` mappába másolja. Az itt megadott forrás elérési útvonala *abszolút*. A fenti két példában a forrás elérési útvonalak *relatívak*. Ha egy elérési útvonal a `/` karakterrel kezdődik, akkor abszolút elérési útvonal, egyébként relatív elérési útvonal.

A `cp` általános szintaxisa a következő:

```
cp OPTIONS SOURCE DESTINATION
```

A `SOURCE` a másolandó fájl, a `DESTINATION` pedig az a mappa, ahová a fájl másolásra kerül. A `SOURCE` és a `DESTINATION` megadható abszolút vagy relatív elérési útvonalként.

Fájlok áthelyezése az `mv`-vel

A másolás `cp` parancsához hasonlóan a Linux biztosít parancsot a fájlok áthelyezésére és átnevezésére. A neve `mv`.

Az áthelyezési művelet megfeleltethető a kivágás és beillesztés műveletének, amelyet általában grafikus felhasználói felületen (GUI) szoktunk végezni.

Ha egy fájlt új helyre szeretnénk áthelyezni, használjuk az `mv` parancsot a következő módon:

```
mv FILENAME DESTINATION_DIRECTORY
```

Íme egy példa:

```
$ mv myfile.txt /home/frank/Documents
```

Az eredmény az, hogy a `myfile.txt` áthelyeződött a `/home/frank/Documents` mappába.

Fájl átnevezéséhez az `mv`-t az alábbi módon használhatjuk:

```
$ mv old_file_name new_file_name
```

Ez megváltoztatja a fájl nevét a `old_file_name`-ről a `new_file_name`-re.

Alapértelmezés szerint az `mv` nem kér megerősítést (technikailag “nem kérdez”), ha egy meglévő fájlt felül akarunk írni (átnevezni). A `-i` opció használatával azonban engedélyezhetjük, hogy a rendszer kérdezzen:

```
$ mv -i old_file_name new_file_name
mv: overwrite 'new_file_name'?
```

Ez a parancs engedélyt kér a felhasználótól, mielőtt felülírná az `old_file_name`-t a `new_file_name`-re.

Ezzel szemben az `-f`:

```
$ mv -f old_file_name new_file_name
```

automatikusan felülírja a fájlt, engedély kérése nélkül.

Fájlok törlése az `rm`-mel

Az `rm` fájlok törlésére szolgál. Gondoljunk rá úgy, mint a “remove” (eltávolítás/törlés) szó rövidített formájára. Vegyük figyelembe, hogy a fájl törlése általában visszafordíthatatlan, ezért ezt a parancsot óvatosan kell használni.

```
$ rm file1
```

Ez a parancs kitörli a `file1`-et.

```
$ rm -i file1
rm: remove regular file 'file1'?
```

Ez a parancs megerősítést kér a felhasználótól a `file1` törlése előtt. Emlékezzünk, hogy fentebb láttuk az `-i` kapcsolót az `mv` használata során.

```
$ rm -f file1
```

Ez a parancs automatikusan törli a `file1`-et, anélkül, hogy megerősítést kérne.

Egyszerre több fájlt is törölhetünk:

```
$ rm file1 file2 file3
```

Ebben a példában a `file1`, `file2` és a `file3` egyszerre kerültek törlésre.

Az `rm` szintaxisa általában a következő:

```
rm OPTIONS FILE
```

Mappák létrehozása és törlése

Mappák létrehozása az `mkdir` segítségével

A mappák létrehozása kritikus fontosságú a fájlok és mappák rendszerezéséhez. A fájlok logikusan csoportosíthatók, ha egy mappában tartjuk őket. Egy mappa létrehozásához használjuk az `mkdir` parancsot:

```
mkdir OPTIONS DIRECTORY_NAME
```

ahol a `DIRECTORY_NAME` a létrehozandó mappa neve. Egyidejűleg tetszőleges számú mappa hozható létre:

```
$ mkdir dir1
```

létre fogja hozni a `dir1` mappát a felhasználó aktuális mappájában.

```
$ mkdir dir1 dir2 dir3
```

Az előző parancs egyszerre három mappát hoz létre: `dir1`, `dir2` és `dir3`.

Ha egy mappát az almappákkal együtt akarunk létrehozni, használjuk a `-p` ("parents" (szülők)) opciót:

```
$ mkdir -p parents/children
```

Ez a parancs létrehozza a `parents/children` mappastruktúrát, azaz létrehozza a `parents` és a `children` mappákat. A `children` a `parents` mappán belül helyezkedik el.

Mappák törlése az `rmdir`-rel

Az `rmdir` töröl egy mappát, *ha az üres*. A szintaxisa az alábbi:

```
rmdir OPTIONS DIRECTORY
```

Ahol a `DIRECTORY` lehet egy argumentum vagy argumentumok listája.

```
$ rmdir dir1
```

Ez a parancs törli a `dir1`-et.

```
$ rmdir dir1 dir2
```

Ez a parancs egyszerre törli a `dir1`-et és a `dir2`-őt.

Eltávolíthatunk egy mappát az almappájával együtt:

```
$ rmdir -p parents/children
```

Ez el fogja távolítani a `parents/children` mappastruktúrát. Jegyezük meg, hogy ha a mappák

üresek, nem kerülnek törlésre!

Fájlok és mappák rekurzív manipulációja

Egy mappának és annak tartalmának manipulálásához *rekurziót* kell alkalmaznunk. A rekurzió azt jelenti, hogy elvégzünk egy műveletet és ezt a műveletet megismételjük a mappastruktúrán végig. A Linuxban az `-r` vagy `-R` vagy `--recursive` opciók általában a rekurzióhoz kapcsolódnak.

A következő forgatókönyv segíthet jobban megérteni a rekurziót:

A `students` mappa tartalmát szeretnénk listázni, amely két almappát tartalmaz: `level 1` és `level 2`, valamint a `frank` nevű fájlt. A rekurzió alkalmazásával az `ls` parancs a `students` mappa tartalmát listázná, azaz: `level 1`, `level 2` és `frank`, de itt nem érne véget. Ugyanígy belépne az `level 1` és `level 2` almappákba, és listázná azok tartalmát, és így tovább a struktúrán lefelé.

Rekurzív listázás az `ls -R`-el

Az `ls -R` egy mappa tartalmának a listázására szolgál, beleértve az almappáit és a fájlokat is.

```
$ ls -R mydirectory
mydirectory/:
file1   newdirectory

mydirectory/newdirectory:
```

A fenti felsorolásban a `mydirectory` és annak összes tartalma szerepel. Megfigyelhetjük, hogy a `mydirectory` tartalmazza a `newdirectory` almappát és a `file1` fájlt. A `newdirectory` üres, ezért nem jelenik meg a tartalma.

Általában egy mappa tartalmának listázásához, beleértve az almappákat is, használjuk a következő parancsot:

```
ls -R DIRECTORY_NAME
```

Egy lezáró slash hozzáadása a `DIRECTORY_NAME`-hez nem jár hatással:

```
$ ls -R animal
```

megegyezik az alábbival

```
$ ls -R animal/
```

Rekurzív másolás a `cp -r`-el

A `cp -r` (vay `-R` vagy `--recursive`) lehetővé teszi egy mappa másolását az összes almappájával és fájljával együtt.

```
$ tree mydir
mydir
|_file1
|_newdir
    |_file2
    |_insidenew
        |_lastdir


3 directories, 2 files
$ mkdir newcopy
$ cp mydir newcopy
cp: omitting directory 'mydir'
$ cp -r mydir newcopy
* tree newcopy
newcopy
|_mydir
    |_file1
    |_newdir
        |_file2
        |_insidenew
            |_lastdir

4 directories, 2 files
```

A fenti listában megfigyelhetjük, hogy amikor megpróbáljuk a `mydir` mappát a `newcopy` mappába másolni, a `cp` használatával, `-r` nélkül, a rendszer a `cp: omitting directory 'mydir'` üzenetet jeleníti meg. Az `-r` opció hozzáadásával azonban a `mydir` mappa teljes tartalma, beleértve önmagát is, átmásolódik a `newcopy` mappába.

A mappák és almappák másolásához használjuk:

```
cp -r SOURCE DESTINATION
```

Rekurzív törlés az `rm -r`-el

Az `rm -r` eltávolít egy mappát és minden tartalmát (almappákat és fájlokat).

WARNING

Legyünk nagyon óvatosak a `-r` vagy az `-rf` opció kombinációjával, amikor az `rm` paranccsal együtt használjuk őket. Egy fontos rendszermappára vonatkozó rekurzív remove parancs használhatatlanná teheti a rendszert. Csak akkor használjuk a rekurzív eltávolítás parancsot, ha teljesen biztosak vagyunk benne, hogy egy mappa tartalmát biztonságosan el lehet távolítani a számítógépről.

Ha megpróbálunk törölni egy mappát a `-r` használata nélkül, a rendszer hibát jelez:

```
$ rm newcopy/
rm: cannot remove 'newcopy/': Is a directory
$ rm -r newcopy/
```

A második parancsban szereplő `-r`-t kell hozzáadni, hogy a törlés működjön.

NOTE

Talán csodálkozunk, hogy miért nem használjuk az `rmdir`-t ebben az esetben. Van egy árnyalatnyi különbség a két parancs között. Az `rmdir` csak akkor törölne sikeresen, ha az adott mappa üres, míg az `rm -r` attól függetlenül használható, hogy ez a mappa üres-e vagy sem.

Adjuk hozzá a `-i` opciót, hogy megerősítést kérjünk a fájl törlése előtt:

```
$ rm -ri mydir/
rm: remove directory 'mydir/'?
```

A rendszer kérdez, mielőtt megpróbálja törölni a `mydir`-t.

Fájl globbing és helyettesítő karakterek

A *globbing* egy olyan funkció, amelyet a Unix/Linux shell biztosít a többszörös fájlnevek ábrázolására a speciális *wildcards* (helyettesítő vagy joker karakterek) használatával. A jokerek lényegében szimbólumok, amelyek egy vagy több karakter helyettesítésére használhatók. Lehetővé teszik például az összes olyan fájl megjelenítését, amely az `A` betűvel kezdődik, vagy az összes olyan fájlét, amely a `.conf` betűkkel végződik.

A jokerek nagyon hasznosak, mivel olyan parancsokkal használhatók, mint az `cp`, `ls` vagy `rm`.

Néhány példa a fájl globbingra:

`rm *`

Az aktuális mappában lévő összes fájl törlése.

`ls l?st`

Az összes olyan fájl listázása, amelynek neve `l`-vel kezdődik, amelyet egy karakter követ, és `st` -vel végződik.

`rmdir [a-z]*`

Eltávolít minden olyan mappát, amelynek a neve betűvel kezdődik.

Joker karakterek típusai

A Linuxban három karakter használható jokerként:

`*` (csillag)

bármely karakter nulla, egy vagy több előfordulását jelenti.

`?` (kérdőjel)

bármely karakter egyetlen előfordulását jelenti.

`[ ]` (zárójeles karakterek)

a szögletes zárójelbe foglalt karakter(ek) bármely előfordulását jelenti. Lehetőség van különböző típusú karakterek használatára, legyenek azok számok, betűk vagy egyéb speciális karakterek. Például a `[0-9]` kifejezés az összes számjegyre illik.

A csillag

A csillag (`*`) bármely karakter nulla, egy vagy több előfordulásával megegyezik.

Például:

```
$ find /home -name *.png
```

Ez az összes olyan fájlt megtalálja, amelynek a vége `.png`, például `photo.png`, `cat.png`, `frank.png`. A `find` parancsot a következő leckében fogjuk részletesebben megvizsgálni.

Hasonlóképpen a:

```
$ ls lpic-*.txt
```

kilistázná az összes olyan szöveges fájlt, amely az `lpic-` karakterekkel kezdődik, amelyeket tetszőleges számú karakter követ és `.txt`-re végződik, mint például `lpic-1.txt` és `lpic-2.txt`.

A csillag helyettesítő karakter használható egy mappa teljes tartalmának manipulálására (másolás, törlés vagy mozgatás):

```
$ cp -r animal/* forest
```

Ebben a példában az `animal` teljes tartalma átmásolódik a `forest`-be.

Általában egy mappa teljes tartalmának másolásához a következőt használjuk:

```
cp -r SOURCE_PATH/* DEST_PATH
```

ahol a `SOURCE_PATH` kihagyható, ha már a szükséges mappában vagyunk.

A csillag, csakúgy, mint bármely más joker, többször is használható ugyanabban a parancsban és bármely pozícióban:

```
$ rm *ate*
```

Az olyan fájlnevek, amelyek előtt egyszer sem, egyszer vagy többször előforduló bármely karakter áll, amelyet az `ate` betűk követnek, és amelyek egyszer sem, egyszer vagy többször előforduló bármely karakterrel végződnek, eltávolításra kerülnek.

A kérdőjel

A kérdőjel (`?`) egy karakter *egyszeri* megjelenésének felel meg.

Nézzük ezt a listát:

```
$ ls
last.txt    lest.txt    list.txt    third.txt   past.txt
```

Ha csak azokat a fájlokat szeretnénk látni, amelyek `l`-el kezdődnek, amelyet bármelyik karakter és az `st.txt` karakterek követnek, akkor a kérdőjeles (`?`) jokert használjuk:

```
$ ls l?st.txt
last.txt    lest.txt    list.txt
```

Csak a `last.txt`, `lest.txt` és `list.txt` fájlok jelennek meg.

Hasonlóképp a

```
$ ls ??st.txt
last.txt    lest.txt    list.txt    past.txt
```

azokat a fájlokat jeleníti meg, amelyek előtt két tetszőleges karakter áll és az`st.txt` szöveg követ.

Zárójelezett karakterek

A szögletes zárójelbe tett helyettesítő karakterek a szögletes zárójelbe zárt karakter(ek) bármely előfordulására illeszkednek:

```
$ ls l[aef]st.txt
last.txt    lest.txt
```

Ez a parancs az összes olyan fájlt listázza, amely az `l`-el kezdődik, amelyet az `aef` halmazban szereplő karakterek közül *minden* követ, és az `st.txt`-vel végződik.

A szögletes zárójelek tartományokat is felvehetnek:

```
$ ls l[a-z]st.txt
last.txt    lest.txt    list.txt
```

Ez az összes olyan fájl nevét kiadja, amelyek neve `l`-el kezdődik, amelyet az `a`-tól `z`-ig terjedő tartományban bármilyen kisbetű követ, és `st.txt`-vel végződik.

Több tartományt is meg lehet adni a szögletes zárójelben:

```
$ ls
student-1A.txt  student-2A.txt  student-3.txt
$ ls student-[0-9][A-Z].txt
student-1A.text student-2A.txt
```

A listában megjelenik egy iskolai címjegyzék a beiratkozott diákokról. Csak azokat a tanulókat

listázza, akiknek a regisztrációs száma megfelel a következő kritériumoknak:

- `student`-el kezdődik,
- amit egy szám és egy nagybetűs karakter követ,
- és `.txt`-re végződik.

Joker karakterek kombinálása

A joker karakterek kombinálhatók, mint például:

```
$ ls
last.txt    lest.txt    list.txt    third.txt   past.txt
$ ls [plf]?st*
last.txt    lest.txt    list.txt    past.txt
```

Az első jokerelem (`[plf]`) a `p`, `l` vagy `f` karakterek bármelyikével megegyezik. A második jokerelem (`?`) bármelyik karakterrel megegyezik. A harmadik jokerelem (`*`) bármely karakter nulla, egy vagy több előfordulásával megegyezik.

```
$ ls
file1.txt file.txt file23.txt fom23.txt
$ ls f*[0-9].txt
file1.txt file23.txt fom23.txt
```

Az előző parancs megjeleníti az összes olyan fájlt, amely `f` betűvel kezdődik, amit egy tetszőleges betűsorozat követ, legalább egy számjegy előfordulása és a `.txt`-vel végződik. Vegyük figyelembe, hogy a `file.txt` nem jelenik meg, mivel nem felel meg ezeknek a feltételeknek.

Gyakorló feladatok

1. Nézzük a alábbi listát:

```
$ ls -lh
total 60K
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Desktop
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Documents
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Downloads
-rw-r--r--  1   frank   frank     21    Sep 7   12:59   emp_name
-rw-r--r--  1   frank   frank     20    Sep 7   13:03   emp_salary
-rw-r--r--  1   frank   frank   8.8K    Apr 1   2018    examples.desktop
-rw-r--r--  1   frank   frank     10    Sep 1   2018    file1
-rw-r--r--  1   frank   frank     10    Sep 1   2018    file2
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Music
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Pictures
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Public
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Templates
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Videos
```

   - Mit jelent a `d` karakter?
   - Miért vannak a méretek ember által olvasható formátumban megadva?
   - Mi lenne a különbség a kimenetben, ha az `ls`-t argumentum nélkül használnánk?

2. Nézzük az alábbi parancsot:

```
$ cp /home/frank/emp_name /home/frank/backup
```

   - Mi történik az `emp_name` fájllal, ha ez a parancs sikeresen végrehajtódik?
   - Ha az `emp_name` mappa lenne, milyen kapcsolót kellene hozzáadni a `cp`-hez, hogy lefusson a parancs?

- Ha a `cp`-t lecserélnénk `mv`-re, mi lenne az eredmény?

3. Nézzük az alábbi listát:

```
$ ls
file1.txt file2.txt file3.txt file4.txt
```

   Melyik joker karakter segítene a mappa teljes tartalmának törlésében?

4. Az előző listára alapozva, milyen fájlokat jelenítene meg a következő parancs?

```
$ ls file*.txt
```

5. Egészítsük ki a parancsot a megfelelő számjegyek és karakterek hozzáadásával a szögletes zárójelek között, amelyek a fenti tartalmakat felsorolják:

```
$ ls file[].txt
```

Gondolkodtató feladatok

1. Hozzunk létre fájlokat `dog` és `cat` néven a home mappában!
2. Hozzunk létre egy `animal` mappát, szintén a home mappában. Helyezzük át a `dog` és a `cat` fájlokat az `animal` mappába!
3. Menjünk a `Documents` mappába a home-ban és hozzuk létre a `backup` mappát!
4. Másoljuk az `animal`-t és a tartalmát a `backup`-ba!
5. Nevezzük át az a `animal`-t a `backup`-ban `animal.bkup`-ra!
6. A `/home/lpi/databases` mappa számos fájlt tartalmaz, többek között: tar.gz`, `db-2.tar.gz` és `db-3.tar.gz`. Melyik egyetlen paranccsal tudjuk csak a fent említett fájlokat listázni?

7. Nézzük az alábbi listát:

```
$ ls
cne1222223.pdf cne12349.txt cne1234.pdf
```

   Melyik paranccsal törölnénk csak a pdf fájlokat, egyetlen globbing karakter használatával?

Összefoglalás

Ebben a leckében azt vizsgáltuk meg, hogyan nézhetjük meg az `ls` paranccsal, hogy mi van egy mappában, hogyan másolhatunk (`cp`) fájlokat és mappákat, és hogyan mozgathatjuk (`mv`) őket. Azt is megnéztük, hogyan hozhatunk létre új mappákat az `mkdir` paranccsal. A fájlok (`rm`) és mappák (`rmdir`) eltávolítására szolgáló parancsokat is tárgyaltuk.

Ebben a leckében megismerkedhettünk a fájl globbinggal és a joker karakterekkel is. A fájl globbingot több fájlnév ábrázolására használjuk speciális karakterek, úgynevezett jokerek használatával. Az alapvető helyettesítő karakterek és jelentésük:

`?` (kérdőjel)

bármely karakter egyetlen előfordulását jelenti.

`[ ]` (szögletes zárójelek)

a szögletes zárójelbe foglalt karakter(ek) bármely előfordulását jelenti.

`*` (csillag)

bármely karakter nulla, egy vagy több előfordulását jelenti.

Ezek a jokerek kombinálhatók ugyanabban az utasításban.

Válaszok a gyakorló feladatokra

1. Nézzük az alábbi listát:

```
$ ls -lh
total 60K
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Desktop
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Documents
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Downloads
-rw-r--r--  1   frank   frank     21    Sep 7   12:59   emp_name
-rw-r--r--  1   frank   frank     20    Sep 7   13:03   emp_salary
-rw-r--r--  1   frank   frank   8.8K    Apr 1   2018    examples.desktop
-rw-r--r--  1   frank   frank     10    Sep 1   2018    file1
-rw-r--r--  1   frank   frank     10    Sep 1   2018    file2
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Music
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Pictures
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Public
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Templates
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Videos
```

   - Mit jelent a `d` karakter?

     A `d` karakter a mappát jelöli.

   - Miért vannak a méretek ember által olvasható formátumban megadva?

     A `-h` kapcsoló miatt.

   - Mi lenne a különbség a kimenetben, ha az `ls`-t argumentum nélkül használnánk?

     Csak a mappa és fájlnevek jelennének meg.

2. Nézzük az alábbi parancsot:

```
$ cp /home/frank/emp_name /home/frank/backup
```

   - Mi történik az `emp_name` fájllal, ha ez a parancs sikeresen végrehajtódik?

     Az `emp_name` bemásolódik a `backup`-ba.

   - Ha az `emp_name` mappa lenne, milyen kapcsolót kellene hozzáadni a `cp`-hez, hogy lefusson a parancs?

`-r`

- Ha a `cp`-t lecserélnénk `mv`-re, mi lenne az eredmény?

  Az `emp_name` áthelyeződne a `backup`-ba. `frank` home mappájában már nem lenne elérhető.

3. Nézzük az alábbi listát:

```
$ ls
file1.txt file2.txt file3.txt file4.txt
```

   Melyik helyettesítő karakter segítene a mappa teljes tartalmának törlésében?

   A csillag `*`.

4. Az előző listára alapozva, milyen fájlokat jelenítene meg a következő parancs?

```
$ ls file*.txt
```

   Mindegyik, mivel a csillag karakter tetszőleges számú karaktert jelöl.

5. Egészítsük ki a parancsot a megfelelő számjegyek és karakterek hozzáadásával a szögletes zárójelek között, amelyek a fenti tartalmakat felsorolják:

```
$ ls file[].txt
```

   `file[0-9].txt`

Válaszok a gondolkodtató feladatokra

1. Hozzunk létre fájlokat `dog` és `cat` néven a home mappában!

```
$ touch dog cat
```

2. Hozzunk létre egy `animal` mappát, szintén a home mappában. Helyezzük át a `dog` és a `cat` fájlokat az `animal` mappába!

```
$ mkdir animal
$ mv dog cat -t animal/
```

3. Menjünk a `Documents` mappába a home-ban és hozzuk létre a `backup` mappát!

```
$ cd ~/Documents
$ mkdir backup
```

4. Másoljuk az `animal`-t és a tartalmát a `backup`-ba!

```
$ cp -r animal ~/Documents/backup
```

5. Nevezzük át az a `animal`-t a `backup`-ban `animal.bkup`-ra!

```
$ mv animal/ animal.bkup
```

6. A `/home/lpi/databases` mappa számos fájlt tartalmaz, többek között: tar.gz`, `db-2.tar.gz` és `db-3.tar.gz`. Melyik egyetlen paranccsal tudjuk csak a fent említett fájlokat listázni?

```
$ ls db-[1-3].tar.gz
```

7. Nézzük az alábbi listát:

```
$ ls
cne1222223.pdf cne12349.txt cne1234.pdf
```

   Melyik paranccsal törölnénk csak a pdf fájlokat, egyetlen globbing karakter használatával?

```
$ rm *.pdf
```


103.3 Lecke 2

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	103 GNU és Unix parancsok
Fejezet:	103.3 Alapvető fájl-menedzsment
Lecke:	2/2

Bevezetés

Fájlok megtalálása

A gép használatakor a fájlok száma és mérete fokozatosan növekszik. Néha nehézségeket okozhat egy adott fájl megtalálása. Szerencsére a Linux biztosítja a `find` funkciót a fájlok gyors keresésére és megtalálására. A `find` a következő szintaxist használja:

```
find STARTING_PATH OPTIONS EXPRESSION
```

`STARTING_PATH`

meghatározza azt a mappát, ahol a keresés kezdődik.

`OPTIONS`

szabályozza a viselkedést és specifikus kritériumokat ad hozzá a keresési folyamat optimalizálása érdekében.

`EXPRESSION`
: meghatározza a keresési lekérdezést.

```
$ find . -name "myfile.txt"
./myfile.txt
```

A kezdő elérési út ebben az esetben az aktuális mappa. A `-name` opció azt adja meg, hogy a keresés a fájl neve alapján történjen. A `myfile.txt` a keresendő fájl neve. Ha fájl-globbingot használunk, ügyeljünk arra, hogy a kifejezés idézőjelek közé kerüljön:

```
$ find /home/frank -name "*.png"
/home/frank/Pictures/logo.png
/home/frank/screenshot.png
```

Ez a parancs megkeresi az összes `.png` végződésű fájlt a `/home/frank/` mappában és alatta. Ha nem értjük a csillag (`*`) használatát, akkor az előző leckéből megismerhetjük.

Kritériumok használata a keresés felgyorsításához

A `find` segítségével *típus*, *méret* vagy *idő* alapján kereshetünk fájlokat. Egy vagy több opció megadásával a kívánt eredményt kevesebb idő alatt kapjuk meg.

Fájl típusa alapján kereső kapcsolók, például:

`-type f`
: fájl keresése.

`-type d`
: mappa keresése.

`-type l`
: szimbolikus link keresése.

```
$ find . -type d -name "example"
```

Ez a parancs megkeresi az összes olyan mappát az aktuális mappában és alatta, amelynek a neve `example`.

A `find`-al használható egyéb kritériumok a következők:

`-name`

a megadott név alapján végez keresést.

`-iname`

a név alapján keres, azonban az kis- és nagybetű nem fontos (pl. a `myFile` teszteset hasonló a `MYFILE`-hoz).

`-not`

visszaadja azokat az eredményeket, amelyek *nem* felelnek meg a tesztesetnek.

`-maxdepth N`

az aktuális mappában, valamint az `N` szint mélységű mappákban keres.

Fájlok keresése módosítási idő szerint

A `find` lehetővé teszi a mappahierarchia szűrését is a fájl módosításának időpontja alapján:

```
$ sudo find / -name "*.conf" -mtime 7
/etc/logrotate.conf
```

Ez a parancs a teljes fájlrendszerben (a kiindulási útvonal a gyökérkönyvtár, azaz a `/`) minden olyan fájlt megkeres, amely a `.conf` karakterekkel végződik és amelyet az elmúlt hét napban módosítottak. Ez a parancs a rendszer mappaszerkezetének alján kezdődő mappák eléréséhez emelt szintű jogosultságokat igényel, ezért használjuk itt a `sudo` szót. Az `mtime`-nak átadott argumentum a fájl utolsó módosítása óta eltelt napok számát jelöli.

Fájlok keresése méret alapján

A `find` *méret* alapján is tud keresni fájlokat. Így kereshetünk `2G`-nél nagyobb fájlokat a `/var`-ban:

```
$ sudo find /var -size +2G
/var/lib/libvirt/images/debian10.qcow2
/var/lib/libvirt/images/rhel8.qcow2
```

A `-size` opció az átadott argumentumnak megfelelő méretű fájlokat jeleníti meg. Néhány példa a következő argumentumokra:

`-size 100b`

pontosan 100 bájt méretű fájlok.

`-size +100k`

100 kilobájtnál nagyobb fájlok.

`-size -20M`

20 megabájtnál kisebb fájlok.

`-size +2G`

2 gigabájtnál nagyobb fájlok.

NOTE Az üres fájlok kereséséhez használhatjuk a következőket: `find . -size 0b` or `find . -empty`.

Az eredményhalmazon végrehajtható akciók

Ha a keresés megtörtént, az `-exec` használatával a kapott halmazon műveleteket lehet végrehajtani:

```
$ find . -name "*.conf" -exec chmod 644 '{}' \;
```

Ez az aktuális mappában (`.`) és alatta minden objektumot szűr a `.conf` végződésű fájlnevekre, majd a `chmod 644` parancsot hajtja végre, hogy módosítsa a jogosultságokat az eredményeken.

Egyelőre ne foglalkozzunk a `'{}' \;` jelentésével, mivel arról később lesz szó.

Fájlok tartalom alapján történő szűrése a `grep` segítségével

A `grep` egy kulcsszó előfordulásának keresésére szolgál.

Vegyünk egy olyan helyzetet, amikor a fájlok keresése a tartalom alapján történik:

```
$ find . -type f -exec grep "lpi" '{}' \; -print
./.bash_history
Alpine/M
helping/M
```

Ez az aktuális mappahierarchiában (`.`) minden olyan objektumot megkeres, amely egy fájl (`-type f`), majd végrehajtja a `grep "lpi"` parancsot minden olyan fájlra, amely megfelel a feltételeknek. A feltételeknek megfelelő fájlok a képernyőre kerülnek (`-print`). A kapcsos zárójelek (`{}`) a `find` találati eredmények placeholderei. A `{}` aposztrófok (`'`) közé van zárva, hogy elkerüljük a speciális karaktereket tartalmazó fájlok `grep`-nek történő átadását. Az `-exec` parancsot

pontosvesszővel (`;`) fejezzük be, amelyet a shell értelmezésének elkerülése érdekében fel kell oldani (`\;`).

A `-delete` opció hozzáadása a kifejezés végéhez az összes megfelelő fájlt törli. Ezt az opciót akkor érdemes használni, ha biztosak vagyunk benne, hogy a találatok csak a törölni kívánt fájlokkal egyeznek meg.

Az alábbi példában a `find` megkeresi, majd törli az összes olyan fájlt a hierarchiában, kezdve az aktuális mappával, amely a `.bak` karakterekkel végződik:

```
$ find . -name "*.bak" -delete
```

Fájlok archiválása

A `tar` parancs (archiválás és tömörítés)

A `tar` parancs, a “tape archive(r)” rövidítése, tar archívumok létrehozására szolgál, egy fájlcsoport archívummá alakításával. Az archívumokat úgy hozzuk létre, hogy egy fájlcsoportot könnyen lehessen mozgatni vagy biztonsági másolatot készíteni róla. Gondoljunk a `tar`-ra úgy, mint egy olyan eszközre, amely olyan ragasztót hoz létre, amelyre a fájlokat fel lehet ragasztani, csoportosítani és könnyen mozgatni.

A `tar` képes a tar archívumok kicsomagolására, az archívumban található fájlok listájának megjelenítésére, valamint további fájlok hozzáadására egy meglévő archívumhoz.

A `tar` parancs szintaxisa az alábbi:

```
tar [OPERATION_AND_OPTIONS] [ARCHIVE_NAME] [FILE_NAME(S)]
```

`OPERATION`

Csak egy műveleti argumentum megengedett és kötelező. A leggyakrabban használt műveletek a következők:

`--create` (`-c`)

Új tar archívum létrehozása.

`--extract` (`-x`)

A teljes archívum vagy egy vagy több fájl csomagolása az archívumból.

`--list` (`-t`)

Az archívumban található fájlok listájának megjelenítése.

`OPTIONS`

A leggyakrabban használt opciók a következők:

`--verbose` (`-v`)

A `tar` parancs által feldolgozott fájlok megjelenítése.

`--file=archive-name` (`-f archive-name`)

Az archívum nevének megadása.

`ARCHIVE_NAME`

Az archívum neve.

`FILE_NAME(S)`

A kicsomagolandó fájlok nevei vesszővel elválasztva. Ha nincs megadva, az egész archívum kitömörítésre kerül.

Archívum létrehozása

Tegyük fel, hogy van egy `stuff` nevű mappánk az aktuális mappában, és azt egy `archive.tar` nevű fájlba szeretnénk menteni. A következő parancsot futtatnánk:

```
$ tar -cvf archive.tar stuff
stuff/
stuff/service.conf
```

Nézzük, mit jelentenek a kapcsolók:

`-c`

Létrehoz egy archívumot.

`-v`

Az archívum létrehozása közben a terminálban megjeleníti a folyamatot, más néven “verbose” mód. A `-v` mindig opcionális ezekben a parancsokban, de hasznos lehet.

`-f`

Lehetővé teszi az archívum nevének megadását.

Általában egyetlen mappa vagy egyetlen fájl archiválásához Linuxon a következőt használjuk:

```
tar -cvf NAME-OF-ARCHIVE.tar /PATH/TO/DIRECTORY-OR-FILE
```

NOTE A `tar` rekurzívan működik. A kívánt műveletet a megadott mappán belül minden további mappában végrehajtja.

Ha egyszerre több mappát szeretnénk archiválni, akkor a `/PATH/TO/DIRECTORY-OR-FILE` szakaszban felsoroljuk az összes mappát egy szóközzel elválasztva:

```
$ tar -cvf archive.tar stuff1 stuff2
```

Ez az `archive.tar` állományban a `stuff1` és `stuff2` archívumát eredményezné.

Archívum kicsomagolása

A `tar` segítségel kicsomagolni is lehet:

```
$ tar -xvf archive.tar
stuff/
stuff/service.conf
```

Ez ki fogja csomagolni az `archive.tar` tartalmát az aktuális mappába.

Ez a parancs ugyanaz, mint a fentebb használt parancs az archívum létrehozására, kivéve a `-x` kapcsolót, amely a `-c` kapcsolót helyettesíti.

Az archívum tartalmának egy adott mappába történő kicsomagolásához a `-C` parancsot használjuk:

```
$ tar -xvf archive.tar -C /tmp
```

Ez kicsomagolja az `archive.tar` tartalmát a `/tmp` mappába.

```
$ ls /tmp
stuff
```

Tömörítés a `tar`-ral

A Linux disztribúciókhoz mellékelt GNU `tar` parancs egyetlen képes létrehozni egy `.tar` archívumot, majd azt `gzip` vagy `bzip2` tömörítéssel tömöríteni egyetlen paranccsal:

```
$ tar -czvf name-of-archive.tar.gz stuff
```

Ez a parancs egy tömörített fájlt hoz létre az `gzip` algoritmus (`-z`) használatával.

Míg a `gzip` tömörítést leggyakrabban a `.tar.gz` vagy `.tgz` fájlok létrehozására használják, a `tar` támogatja a `bzip2` tömörítést is. Ez lehetővé teszi a `bzip2` tömörített fájlok létrehozását, amelyek neve gyakran `.tar.bz2`, `.tar.bz` vagy `.tbz`.

Ehhez a `-z`-t a `gzip` helyett a `-j`-t a `bzip2`-re cseréljük:

```
$ tar -cjvf name-of-archive.tar.bz stuff
```

A fájl kitömörítéséhez a `-c`-t `-x`-el helyettesítjük, ahol az `x` a “extract”-ot jelenti:

```
$ tar -xzvf archive.tar.gz
```

A `gzip` gyorsabb, de általában valamivel kevesebbet tömörít, így valamivel nagyobb fájlt kapunk. A `bzip2` lassabb, de kicsit jobban tömörít, így valamivel kisebb fájlt kapunk. Általában azonban a `gzip` és a `bzip2` gyakorlatilag ugyanaz, és mindkettő hasonlóan működik.

Alternatívaként alkalmazhatjuk a `gzip` vagy a `bzip2` tömörítést a `gzip` parancs segítségével a `gzip` tömörítéshez és a `bzip` parancs segítségével a `bzip` tömörítéshez. Például a `gzip` tömörítés alkalmazásához használjuk a következőt:

```
gzip FILE-TO-COMPRESS
```

`gzip`
: ugyanezzel a névvel, de `.gz` végződéssel hozza létre a tömörített fájlt.

`gzip`
: a tömörített fájl létrehozása után eltávolítja az eredeti fájlokat.

A `bzip2` parancs hasonló módon működik.

A fájlok kicsomagolásához a `gunzip` vagy a `bunzip2` algoritmust használjuk a fájl tömörítéséhez használt algoritmustól függően.

A `cpio` parancs

A `cpio` a `copy in, copy out` rövidítése. Olyan archív fájlok feldolgozására szolgál, mint a `*.cpio` vagy a `*.tar` fájlok.

A `cpio` az alábbi műveleteket hajtja végre:

- Fájlok archívumba másolása.
- Fájlok kicsomagolása archívumból.

A fájlok listáját a szabványos bemenetről veszi (többnyire az `ls` kimenetéről).

Egy `cpio` archívumot így hozhatunk létre:

```
$ ls | cpio -o > archive.cpio
```

Az `-o` opció utasítja a `cpio-t`, hogy hozzon létre egy kimenetet. Ebben az esetben a létrehozott kimeneti fájl az `archive.cpio`. Az `ls` parancs felsorolja az aktuális mappa archiválandó tartalmát.

Az archívum kicsomagolása az alábbi módon történik:

```
$ cpio -id < archive.cpio
```

Az `-i` opciót a kicsomagolás elvégzésére használjuk. A `-d` opció létrehozza a célmappát. A `<` karakter a szabványos bemenetet jelöli. A kicsomagolandó bemeneti fájl az `archive.cpio`.

A `dd` parancs

A `dd` adatokat másol át egyik helyről a másikra. A `dd` parancssori szintaxisa sok más Unix programtól eltér, a GNU szabványos `-option value` vagy `--option=value` formátumai helyett az `option=value` szintaxist használja a parancssori opciókhoz:

```
$ dd if=oldfile of=newfile
```

Ez a parancs a `oldfile` tartalmát másolja az `newfile`-be, ahol a `if=` a bemeneti fájl, az `of=` pedig a kimeneti fájlra utal.

NOTE

A `dd` parancs általában nem ír ki semmit a képernyőre, amíg a parancs be nem fejeződik. A `status=progress` opció megadásával a konzol megjeleníti a parancs által végzett munka mennyiségét. Például: `dd status=progress if=oldfile of=newfile`.

A `dd` az adatok nagy/kisbetűsre változtatására vagy közvetlenül a blokkeszközökre, például a `/dev/sdb`-re történő írásra is használható:

```
$ dd if=oldfile of=newfile conv=ucase
```

Ez az `oldfile` teljes tartalmát átmásolja a `newfile`-ba, és az egész szöveget nagybetűvel írja.

A következő parancs a `/dev/sda` lemezen található teljes merevlemezről biztonsági mentést készít a `backup.dd` nevű fájlba:

```
$ dd if=/dev/sda of=backup.dd bs=4096
```

Gyakorló feladatok

1. Nézzük az alábbi listát:

```
$ find /home/frank/Documents/ -type d
/home/frank/Documents/
/home/frank/Documents/animal
/home/frank/Documents/animal/domestic
/home/frank/Documents/animal/wild
```

   - Milyen típusó fájlokat fog kiírni ez a parancs?

   - Melyik mappában kezdődik meg a keresés?

2. Egy felhasználó tömöríteni szeretné a backup mappáját. Az alábbi parancsot használja:

```
$ tar cvf /home/frank/backup.tar.gz /home/frank/dir1
```

   Melyik kapcsoló hiányzik a mentés tömörítéséhez a `gzip` algoritmus használatával?

Gondolkodtató feladatok

1. Rendszergazdaként rendszeres ellenőrzéseket kell végezni nagy mennyiségű fájlok eltávolítása érdekében. Ezek a terjedelmes fájlok a `/var` könyvtárban találhatók, és `.backup` kiterjesztéssel végződnek.
   - Írjuk le a parancsot, ami a `find` használatával megkeresi ezeket a fájlokat:
   - A fájlok méretének elemzése azt mutatja, hogy a fájlok mérete `100M` és `1000M` között mozog. Egészítsük ki az előző parancsot ezzel az új információval, hogy megtalálhassuk a `100M` és `1000M` közötti méretű mentési fájlokat:
   - Végül fejezzük be ezt a parancsot a törlés művelettel, hogy ezek a fájlok eltávolításra kerüljenek:
2. A `/var` mappában négy backup fájl található:

```
db-jan-2018.backup
db-feb-2018.backup
db-march-2018.backup
db-apr-2018.backup
```

   - A `tar` segítségével adja meg azt a parancsot, amely létrehozza az `db-first-quarter-2018.backup.tar` nevű archívumfájlt:
   - Adjuk meg azt a `tar` parancsot, amely létrehozza az archívumot és a `gzip` segítségével tömöríti. Figyeljünk arra, hogy az eredményül kapott fájlnévnek `.gz` végződéssel kell végződnie:

Összefoglalás

Ebben a leckében megtanultuk:

- Hogyan keressünk fájlokat a `find` segítségével.
- Hogyan adhatunk hozzá keresési feltételeket idő, fájltípus vagy méret alapján a `find`-hoz való argumentumokkal.
- Hogyan hajtsunk végre műveletet a visszaadott eredményhalmazon.
- Hogyan archiváljunk, tömörítsünk és csomagoljunk ki fájlokat a `tar` segítségével.
- Archívumok feldolgozása a `cpio`-val.
- Fájlok másolása a `dd` programmal.

Válaszok a gyakorló feladatokra

1. Nézzük az alábbi listát:

```
$ find /home/frank/Documents/ -type d
/home/frank/Documents/
/home/frank/Documents/animal
/home/frank/Documents/animal/domestic
/home/frank/Documents/animal/wild
```

   - Milyen típusó fájlokat fog kiírni ez a parancs?

     Mappákat.

   - Melyik mappában kezdődik meg a keresés?

     `/home/frank/Documents`

2. Egy felhasználó tömöríteni szeretné a backup mappáját. Az alábbi parancsot használja:

```
$ tar cvf /home/frank/backup.tar.gz /home/frank/dir1
```

   Melyik kapcsoló hiányzik a mentés tömörítéséhez a `gzip` algoritmus használatával?

   A `-z`.

Válaszok a gondolkodtató feladatokra

1. Rendszergazdaként rendszeres ellenőrzéseket kell végezni nagy mennyiségű fájlok eltávolítása érdekében. Ezek a terjedelmes fájlok a `/var` könyvtárban találhatók, és `.backup` kiterjesztéssel végződnek.
   - Írjuk le a parancsot, ami a `find` használatával megkeresi ezeket a fájlokat:

     ```
     $ find /var -name *.backup
     ```

   - A fájlok méretének elemzése azt mutatja, hogy a fájlok mérete `100M` és `1000M` között mozog. Egészítsük ki az előző parancsot ezzel az új információval, hogy megtalálhassuk a `100M` és `1000M` közötti méretű backup fájlokat:

     ```
     $ find /var -name *.backup -size +100M -size -1000M
     ```

   - Végül fejezzük be ezt a parancsot a törlés művelettel, hogy ezek a fájlok eltávolításra kerüljenek:

     ```
     $ find /var -name *.backup -size +100M -size -1000M -delete
     ```

2. A `/var` mappában négy backup fájl található:

   ```
   db-jan-2018.backup
   db-feb-2018.backup
   db-march-2018.backup
   db-apr-2018.backup
   ```

   - A `tar` segítségével adja meg azt a parancsot, amely létrehozza az `db-first-quarter-2018.backup.tar` nevű archívumfájlt:

     ```
     $ tar -cvf db-first-quarter-2018.backup.tar db-jan-2018.backup db-feb-2018.backup db-march-2018.backup db-apr-2018.backup
     ```

   - Adjuk meg azt a `tar` parancsot, amely létrehozza az archívumot és a `gzip` segítségével tömöríti. Figyeljünk arra, hogy az eredményül kapott fájlnévnek `.gz` végződéssel kell végződnie:

```
$ tar -zcvf db-first-quarter-2018.backup.tar.gz db-jan-2018.backup db-feb-2018.backup
db-march-2018.backup db-apr-2018.backup
```


103.4 Folyamok, csövek és átirányítások használata

Hivatkozás az LPI célkitűzésre

LPIC-1 v5, Exam 101, Objective 103.4

Súlyozás

4

Kulcsfontosságú ismeretek

- A standard bemenet, a standard kimenet és a standard hiba átirányítása.
- Egy parancs kimenetének átvezetése egy másik parancs bemenetére.
- Egy parancs kimenetének használata egy másik parancs argumentumaként.
- Kimenet küldése az stdout-ra és egy fájlba is.

A használt fájlok, kifejezések és segédprogramok listája

- `tee`
- `xargs`

Linux Professional Institute

103.4 Lecke 1

Tanúsítvány:	LPIC-1 (101)
Verzió:	5.0
Témakör:	103 GNU és Unix parancsok
Fejezet:	103.4 Folyamok, csövek és átirányítások használata
Lecke:	1/2

Bevezetés

Minden számítógépes program ugyanazt az általános elvet követi: a valamilyen forrásból kapott adatokat átalakítják, hogy érthető eredményt hozzanak létre. Linux shell környezetben az adatforrás lehet egy helyi fájl, egy távoli fájl, egy eszköz (például egy billentyűzet), stb. A program kimenete általában a képernyőn jelenik meg, de gyakori az is, hogy a kimeneti adatokat egy helyi fájlrendszerben tárolja, egy távoli eszközre küldi, hangszórókon keresztül játssza le, stb.

A Unix által inspirált operációs rendszerek, mint például a Linux, sokféle be- és kimeneti módszert kínálnak. Különösen a *fájlleírók* (file descriptors) módszer teszi lehetővé, hogy egész számokat dinamikusan hozzárendeljünk az adatcsatornákhoz, így egy processz hivatkozhat rájuk, mint a bemeneti/kimeneti adatfolyamaira.

A szabványos Linux-folyamatok alapértelmezés szerint három kommunikációs csatornát nyitnak meg: a *standard input* csatornát (legtöbbször egyszerűen *stdin*), a *standard output* csatornát (*stdout*) és a *standard error* csatornát (*stderr*). Az ezekhez a csatornákhoz rendelt numerikus fájlleírók: `0` az stdin, `1` az stdout és `2` az stderr. A kommunikációs csatornák a `/dev/stdin`, `/dev/stdout` és `/dev/stderr` speciális eszközökön keresztül is elérhetők.

Ez a három szabványos kommunikációs csatorna lehetővé teszi a programozók számára, hogy olyan kódot írjanak, amely adatokat olvas és ír anélkül, hogy aggódnának amiatt, hogy milyen adathordozóról érkezik vagy hova megy. Ha például egy programnak adathalmazra van szüksége bemenetként, akkor egyszerűen kérhet adatokat a szabványos bemenetről, és bármi is legyen az, amit szabványos bemenetként használnak, biztosítja az adatokat. Hasonlóképpen, a legegyszerűbb módszer, amellyel egy program megjelenítheti a kimenetét, ha azt a standard kimenetre írja. Egy standard shell munkamenetben a billentyűzetet stdin-ként, a monitor képernyőjét pedig stdout és stderr-ként definiáljuk.

A Bash shell képes a kommunikációs csatornák újbóli kiosztására egy program betöltésekor. Lehetővé teszi például, hogy a képernyő mint szabványos kimenet felülbírálható legyen, és a helyi fájlrendszerben lévő fájlt használjuk stdoutként.

Átirányítások

Egy csatorna fájlleírójának a shell környezetében történő újbóli hozzárendelését *redirect*-nek (átirányítás) nevezzük. Az átirányítást egy speciális karakter határozza meg a parancssoron belül. Például egy folyamat standard kimenetének egy fájlba történő átirányításához a parancs végére a *nagyobb mint* szimbólum `>` kerül, amelyet az átirányított kimenetet fogadó fájl elérési útvonala követ:

```
$ cat /proc/cpuinfo >/tmp/cpu.txt
```

Alapértelmezés szerint csak az stdout-ra érkező tartalom kerül átirányításra. Ez azért történik, mert a fájlleíró numerikus értékét közvetlenül a nagyobb mint szimbólum előtt kell megadni, és ha ez nincs megadva, a Bash a standard kimenetet irányítja át. Ezért a `>` használata egyenértékű az `1>` használatával (az stdout fájlleíró értéke `1`).

Az stderr tartalmának rögzítéséhez a `2>` átirányítást kell használni. A legtöbb parancssoros program hibakeresési információkat és hibaüzeneteket küld a standard hibacsatornára. Lehetséges például egy nem létező fájl beolvasási kísérlete által kiváltott hibaüzenet rögzítése:

```
$ cat /proc/cpu_info 2>/tmp/error.txt
$ cat /tmp/error.txt
cat: /proc/cpu_info: No such file or directory
```

Mind az stdout, mind az stderr átirányítása ugyanabból a célból történik a `&>` vagy a `>&` segítségével. Fontos, hogy az amperjel mellé ne tegyünk szóközöket, különben a Bash azt a folyamat háttérben történő futtatására vonatkozó utasításnak fogja venni, és nem az átirányítás

végrehajtására.

A cél egy írható fájl elérési útja, például `/tmp/cpu.txt`, vagy egy írható fájlleíró. A fájlleíró célpontot a fájlleíró numerikus értéke után egy amperjel jelöli. Például az `1>&2` átirányítja az stdout-ot az stderr-re. Ha ennek ellenkezőjét, stderr-t stdout-ra akarjuk átirányítani, akkor a `2>&1` -t kell használni helyette.

Bár nem túl hasznos, mivel ugyanezt a feladatot rövidebb úton is el lehet végezni, lehetséges az stderr-t átirányítani az stdout-ra, majd átirányítani egy fájlba. Például egy átirányítás, amely az stderr-t és az stdout-ot is egy `log.txt` nevű fájlba írja, a következőképpen írható: `>log.txt 2>&1`. Az stderr átirányításának fő oka azonban az, hogy lehetővé tegye a hibakeresési és hibaüzenetek elemzését. Lehetséges egy program standard kimenetét egy másik program standard bemenetére átirányítani, de nem lehetséges a standard hiba közvetlen átirányítása egy másik program standard bemenetére. Így a program stderr-re küldött üzeneteit először át kell irányítani az stdout-ra ahhoz, hogy egy másik program stdin-je beolvashassa.

Ha a parancs kimenetét csak el akarjuk dobni, akkor annak tartalmát átirányíthatjuk a `/dev/null` speciális fájlba. Például a `>log.txt 2>/dev/null` az stdout tartalmát a `log.txt` fájlba menti, az stderr-t pedig eldobja. A `/dev/null` állományt bármely felhasználó írhatja, de semmilyen adat nem állítható vissza belőle, mivel nem tárolja sehol.

Hibaüzenet jelenik meg, ha a megadott cél nem írható (ha az útvonal mappára vagy csak olvasható fájlra mutat), és a cél nem módosítható. A kimeneti átirányítás azonban megerősítés nélkül felülír egy meglévő írható célt. A kimeneti átirányítások felülírják a fájlokat, kivéve, ha a Bash `noclobber` opció engedélyezve van, amit az aktuális munkamenetre a `set -o noclobber` vagy a `set -C` paranccsal lehet megtenni:

```
$ set -o noclobber
$ cat /proc/cpu_info 2>/tmp/error.txt
-bash: /tmp/error.txt: cannot overwrite existing file
```

A `noclobber` opció aktuális munkamenetből történő visszavonásához futtassuk a `set +o noclobber` vagy a `set +C` parancsot. Ahhoz, hogy a `noclobber` opció tartós legyen, a felhasználó Bash-profiljában vagy a rendszerszintű profilban kell szerepelnie.

Még a `noclobber` opció engedélyezésével is lehetséges az átirányított adatok hozzáadása a meglévő tartalomhoz. Ez egy átirányítással érhető el, amelyet két nagyobb mint szimbólummal `>>` írunk:

```
$ cat /proc/cpu_info 2>>/tmp/error.txt
$ cat /tmp/error.txt
```

```
cat: /proc/cpu_info: No such file or directory
cat: /proc/cpu_info: No such file or directory
```

Az előző példában az új hibaüzenet a `/tmp/error.txt` fájlban lévőhöz lett hozzácsatolva. Ha a fájl még nem létezik, akkor létrejön az új adatokkal.

A folyamat standard bemenetének adatforrása is átrendezhető. A kevesebb, mint szimbólum `<` arra szolgál, hogy egy fájl tartalmát átirányítsuk egy folyamat stdin-jébe. Ebben az esetben az adatok jobbról balra áramlanak: az átirányított leíró a kisebb mint szimbólum bal oldalán 0-nak, az adatforrásnak (egy fájl elérési útvonalának) pedig a kisebb mint szimbólum jobb oldalán kell lennie. A `uniq` parancs, mint a legtöbb szövegfeldolgozó parancssori segédprogram, alapértelmezés szerint elfogadja az stdin-re küldött adatokat:

```
$ uniq -c </tmp/error.txt
      2 cat: /proc/cpu_info: No such file or directory
```

A `-c` opcióval a `uniq` azt jeleníti meg, hogy hányszor szerepel egy ismételt sor a szövegben. Mivel az átirányított fájlleíró numerikus értéke el lett nyomva, a példaparancs a `uniq -c 0</tmp/error.txt` paranccsal egyenértékű. A `0`-tól eltérő fájlleíró használatának egy bemeneti átirányításban csak bizonyos kontextusokban van értelme, mert lehetséges, hogy egy program a `3`, `4` stb. fájlleíróknál kérjen adatokat. Valójában a programok bármely 2 feletti egész számot használhatnak új fájlleíróként az adatok be- és kimenetéhez. Például a következő C kód adatokat olvas be a `3` fájlleíróból, és egyszerűen átmásolja a `4` fájlleíróba:

NOTE A programnak helyesen kell kezelnie az ilyen fájlleírókat, különben érvénytelen olvasási vagy írási műveletet kísérelhet meg, és összeomolhat.

```
#include <stdio.h>

int main(int argc, char **argv){
  FILE *fd_3, *fd_4;
  // 3-as fájlleíró megnyitása
  fd_3 = fdopen(3, "r");
  // 4-es fájlleíró megnyitása
  fd_4 = fdopen(4, "w");
  // Olvasás a 3-as fájlleíróból
  char buf[32];
  while ( fgets(buf, 32, fd_3) != NULL ){
    // Írás a 4-es fájlleíróba
    fprintf(fd_4, "%s", buf);
  }
```

```
  // Mindkét fájlleíró bezárása
  fclose(fd_3);
  fclose(fd_4);
}
```

A teszteléshez mentsük el a mintakódot `fd.c` néven, és fordítsuk le a `gcc -o fd fd.c` paranccsal. Ennek a programnak szüksége van arra, hogy a 3. és 4. fájlleíró elérhető legyen, hogy tudjon olvasni és írni rájuk. Példaként a korábban létrehozott `/tmp/error.txt` fájlt használhatjuk a `3` fájlleíró forrásaként, a `4` fájlleírót pedig átirányíthatjuk az stdout-ra:

```
$ ./fd 3</tmp/error.txt 4>&1
cat: /proc/cpu_info: No such file or directory
cat: /proc/cpu_info: No such file or directory
```

A programozó szempontjából a fájlleírók használatával elkerülhető, hogy az opcióelemzéssel és a fájlrendszeri elérési utakkal kelljen foglalkozni. Ugyanaz a fájlleíró akár bemenetként és kimenetként is használható. Ebben az esetben a fájlleírót a parancssorban kisebb mint és nagyobb mint szimbólumokkal definiáljuk, mint például a `3<>/tmp/error.txt`-ben.

Here Document és Here String

A bemenet átirányításának másik módja a *Here document* és *Here string* metódusok használata. A Here document átirányítás lehetővé teszi, hogy többsoros szöveget írjunk be, amely az átirányított tartalomként fog szerepelni. Két kisebb mint szimbólum `<<<` jelzi a Here document átirányítást:

```
$ wc -c <<EOF
> How many characters (Hány karakter)
> in this Here document? (van ebben a Here document-ben?)
> EOF
43
```

A két kisebb mint szimbólum `<<` jobb oldalán található az `EOF` záró kifejezés. A beszúrási mód befejeződik, amint egy olyan sor kerül beírásra, amely csak a befejező kifejezést tartalmazza. Bármilyen más kifejezés is használható befejező kifejezésként, de fontos, hogy a kevesebb mint szimbólum és a befejező kifejezés közé ne kerüljenek üres karakterek. A fenti példában a két sornyi szöveget a `wc -c` parancs küldte el az stdin-be, amely a karakterek számát jeleníti meg. A fájlok bemeneti átirányításához hasonlóan az stdin (`0` fájlleíró) feltételezhető, ha az átirányított fájlleíró el van nyomva.

A Here string módszer hasonló a Here document módszerhez, de csak egy sorra vonatkozik:

```
$ wc -c <<<"How many characters in this Here string?"
41
```

Ebben a példában a három kisebb, mint jel jobb oldalán lévő sztringet a `wc -c` elküldi az stdin-be, amely megszámolja a karakterek számát. A szóközöket tartalmazó sztringeknek idézőjelek között kell lenniük, különben csak az első szó lesz Here stringként használva, a többi pedig argumentumként kerül átadásra a parancsnak.

Gyakorló feladatok

1. A szöveges fájlok mellett a `cat` parancs bináris adatokkal is dolgozhat, például egy blokkeszköz tartalmát küldheti el egy fájlba. Az átirányítás segítségével hogyan tudja a `cat` elküldeni a `/dev/sdc` eszköz tartalmát az aktuális könyvtárban lévő `sdc.img` fájlba?

2. Mi a neve a `date 1> now.txt` parancs által átirányított szabványos csatornának?

3. Miután megpróbáltunk felülírni egy fájlt átirányítással, a felhasználó hibaüzenetet kap, amely szerint a `noclobber` opció engedélyezve van. Hogyan lehet a `noclobber` opciót kikapcsolni az aktuális munkamenetre?

4. Mi lesz a `cat <<.>/dev/stdout` parancs eredménye?

Gondolkodtató feladatok

1. A `cat /proc/cpu_info` parancs hibaüzenetet jelenít meg, mert a `/proc/cpu_info` nem létezik. Hova irányítja a hibaüzenetet a `cat /proc/cpu_info 2>1` parancs?

2. Akkor is lehetséges lesz a `/dev/null`-ra küldött tartalmak eldobása, ha a `noclobber` opció engedélyezve van az aktuális shell munkamenethez?

3. Az `echo` használata nélkül hogyan lehetne a `$USER` változó tartalmát átirányítani a `sha1sum` parancs stdin-jébe?

4. A Linux kernel a `/proc/PID/fd/` szimbolikus hivatkozásokat tart fenn minden olyan fájlhoz, amelyet egy folyamat megnyitott, ahol *PID* a megfelelő folyamat azonosító száma. Hogyan használhatná a rendszergazda ezt a mappát az `nginx` által megnyitott naplófájlok helyének ellenőrzésére, ha feltételezzük, hogy a PID-je `1234`?

5. Aritmetikai számításokat csak a shell beépített parancsaival lehet végezni, de a lebegőpontos számításokhoz speciális programokra van szükség, mint például a `bc` (*basic calculator*). A `bc` programmal még a tizedesjegyek számát is meg lehet adni a `scale` paraméterrel. A `bc` azonban csak a szabványos bemeneten keresztül fogad el műveleteket, általában interaktív módban beírva. Egy Here string segítségével hogyan küldhetjük el a `scale=6; 1/3` lebegőpontos műveletet a `bc` standard bemenetére?

Összefoglalás

Ez a lecke olyan módszereket mutatott be, amelyekkel egy program a szabványos kommunikációs csatornák átirányításával futtatható. A Linux-folyamatok ezeket a szabványos csatornákat általános *fájlleíróként* használják az adatok olvasására és írására, így lehetővé téve, hogy tetszőlegesen fájlokra vagy eszközökre változtassuk őket. A lecke a következő lépéseken ment keresztül:

- Mik azok a fájlleírók és milyen szerepet játszanak a Linuxban.
- Minden folyamat szabványos kommunikációs csatornái: *stdin*, *stdout* és *stderr*.
- Hogyan kell helyesen végrehajtani egy parancsot az adatok átirányításával, mind a bemenet, mind a kimenet esetében.
- Hogyan használjuk a *Here Documents* és a *Here Strings* kifejezéseket a bemeneti átirányításokban.

A tárgyalt parancsok és eljárások a következők voltak:

- Átirányító operátorok: `>`, `<`, `>>`, `<<`, `<<<`.
- A `cat`, `set`, `uniq` és `wc` parancsok.

Válaszok a gyakorló feladatokra

1. A szöveges fájlok mellett a `cat` parancs bináris adatokkal is dolgozhat, például egy blokkeszköz tartalmát küldheti el egy fájlba. Az átirányítás segítségével hogyan tudja a `cat` elküldeni a `/dev/sdc` eszköz tartalmát az aktuális mappában lévő `sdc.img` fájlba?

   ```
   $ cat /dev/sdc > sdc.img
   ```

2. Mi a neve a `date 1> now.txt` parancs által átirányított szabványos csatornának?

   Standard output (standard kimenet) vagy stdout

3. Miután megpróbáltunk felülírni egy fájlt átirányítással, a felhasználó hibaüzenetet kap, amely szerint a `noclobber` opció engedélyezve van. Hogyan lehet a `noclobber` opciót kikapcsolni az aktuális munkamenetre?

   `set +C` vagy `set +o noclobber`

4. Mi lesz a `cat <<.>/dev/stdout` parancs eredménye?

   A Bash belép a Heredoc beviteli módba, majd kilép, ha csak egy pont jelenik meg egy sorban. A beírt szöveget átirányítja az stdout-ra (a képernyőre).

Válaszok a gondolkodtató feladatokra

1. A `cat /proc/cpu_info` parancs hibaüzenetet jelenít meg, mert a `/proc/cpu_info` nem létezik. Hova irányítja a hibaüzenetet a `cat /proc/cpu_info 2>1` parancs?

   Az `1` nevű fájlba az aktuális mappában.

2. Akkor is lehetséges lesz a `/dev/null`-ra küldött tartalmak eldobása, ha a `noclobber` opció engedélyezve van az aktuális shell munkamenethez?

   Igen, a `/dev/null` egy speciális fájl és nincs rá hatással a `noclobber`.

3. Az `echo` használata nélkül hogyan lehetne a `$USER` változó tartalmát átirányítani a `sha1sum` parancs stdin-jébe?

   ```
   $ sha1sum <<<$USER
   ```

4. A Linux kernel a `/proc/PID/fd/` szimbolikus hivatkozásokat tart fenn minden olyan fájlhoz, amelyet egy folyamat megnyitott, ahol *PID* a megfelelő folyamat azonosító száma. Hogyan használhatná a rendszergazda ezt a mappát az `nginx` által megnyitott naplófájlok helyének ellenőrzésére, ha feltételezzük, hogy a PID-je `1234`?

   Az `ls -l /proc/1234/fd` parancs kiadásával, amely megjeleníti a mappában lévő összes szimbolikus link célpontját.

5. Aritmetikai számításokat csak a shell beépített parancsaival lehet végezni, de a lebegőpontos számításokhoz speciális programokra van szükség, mint például a `bc` (*basic calculator*). A `bc` programmal még a tizedesjegyek számát is meg lehet adni a `scale` paraméterrel. A `bc` azonban csak a szabványos bemeneten keresztül fogad el műveleteket, általában interaktív módban beírva. Egy Here string segítségével hogyan küldhetjük el a `scale=6; 1/3` lebegőpontos műveletet a `bc` standard bemenetére?

   ```
   $ bc <<<"scale=6; 1/3"
   ```

103.4 Lecke 2

Tanúsítvány:	LPIC-1 (101)
Verzió:	5.0
Témakör:	103 GNU és Unix parancsok
Fejezet:	103.4 Folyamok, csövek és átirányítások használata
Lecke:	2/2

Bevezetés

A Unix filozófia egyik aspektusa szerint minden programnak meghatározott céllal kell rendelkeznie, és nem szabad megpróbálnia a hatáskörén kívül eső funkciókat beépíteni. Az egyszerűség azonban nem jelenti azt, hogy a dolgok kevésbé kidolgozott eredményekkel járnak, mivel különböző programok összekapcsolhatók, hogy egy kombinált kimenetet állítsanak elő. A függőleges vonal karakter `|`, más néven a *pipe* szimbólummal létrehozható egy olyan csővezeték, amely egy program kimenetét közvetlenül egy másik program bemenetéhez kapcsolja, míg a *parancshelyettesítés* (command substitution) lehetővé teszi, hogy egy program kimenetét egy változóban tároljuk, vagy közvetlenül egy másik parancs argumentumaként használjuk.

Csővezetékek

Az átirányítással ellentétben a csövek (pipe-ok) esetében az adatok balról jobbra haladnak a parancssorban, és a cél egy másik processz, nem pedig egy fájlrendszeri elérési útvonal, fájlleíró vagy Here document. A pipe karakter `|` azt mondja a shellnek, hogy minden különböző parancsot egyszerre indítson el, és az előző parancs kimenetét balról jobbra haladva kösse össze a következő parancs bemenetével. Például az átirányítások használata helyett a `cat` által a standard

kimenetre küldött `/proc/cpuinfo` fájl tartalmát a következő paranccsal a `wc` stdin-jébe vezethetjük:

```
$ cat /proc/cpuinfo | wc
   208    1184    6096
```

A fájl elérési útvonalának hiányában a `wc` a sorok, szavak és karakterek számát számolja, amit az stdin-en kap, ahogy a példában is. Egy összetett parancsban több csővezeték is lehet. A következő példában két csővezetéket használunk:

```
$ cat /proc/cpuinfo | grep 'model name' | uniq
model name      : Intel(R) Xeon(R) CPU           X5355  @ 2.66GHz
```

A `cat /proc/cpuinfo` parancs által előállított `/proc/cpuinfo` fájl tartalmát a `grep 'model name'` parancshoz továbbítottuk, amely csak a `model name` kifejezést tartalmazó sorokat választja ki. A példát futtató gépen sok CPU van, ezért többször is vannak `model name` tartalmú sorok. Az utolsó cső a `grep 'model name'-t a `uniq` parancshoz köti, amely az előzővel megegyező sorok kihagyásáért felelős.

A csővezetékek kombinálhatók átirányításokkal ugyanabban a parancsban. Az előző példa átírható egyszerűbb formára:

```
$ grep 'model name' </proc/cpuinfo | uniq
model name      : Intel(R) Xeon(R) CPU           X5355  @ 2.66GHz
```

A `grep` bemeneti átirányítása nem feltétlenül szükséges, mivel a `grep` elfogadja a fájl elérési útvonalát argumentumként, de a példa jól szemlélteti, hogyan lehet ilyen kombinált parancsokat létrehozni.

A csövek és az átirányítások kizárólagosak, azaz egy forrás csak egy célhoz rendelhető hozzá. Mégis lehetséges egy kimenetet átirányítani egy fájlba, és láthatjuk azt a `tee` segítségével a képernyőn. Ehhez az első program a kimenetét a `tee` stdin-jébe küldjük, és az utóbbinak megadunk egy fájlnevet az adatok tárolásához:

```
$ grep 'model name' </proc/cpuinfo | uniq | tee cpu_model.txt
model name      : Intel(R) Xeon(R) CPU           X5355  @ 2.66GHz
$ cat cpu_model.txt
model name      : Intel(R) Xeon(R) CPU           X5355  @ 2.66GHz
```

A lánc utolsó, `uniq` által generált programjának kimenete megjelenik, és a `cpu_model.txt` fájlban tárolódik. Ahhoz, hogy a megadott fájl tartalmát ne írjuk felül, hanem adatokat csatoljunk hozzá, a `tee`-nek meg kell adni az `-a` opciót.

Egy csővezeték csak a processz standard kimenetét rögzíti. Tegyük fel, hogy egy hosszú fordítási folyamatot kell végigvinnünk a képernyőn, és ugyanakkor a standard kimenetet és a standard hibát is el kell mentenünk egy fájlba későbbi ellenőrzés céljából. Feltételezve, hogy az aktuális mappánkban nincs *Makefile*, a következő parancs hibát fog kiadni:

```
$ make | tee log.txt
make: *** No targets specified and no makefile found.  Stop.
```

Bár a képernyőn megjelenik, a `make` által generált hibaüzenetet a `tee` nem rögzítette, és a *log.txt* fájl üresen jött létre. Egy átirányítást kell végrehajtani, mielőtt a pipe rögzíteni tudja az stderr-t:

```
$ make 2>&1 | tee log.txt
make: *** No targets specified and no makefile found.  Stop.
$ cat log.txt
make: *** No targets specified and no makefile found.  Stop.
```

Ebben a példában a `make` stderr-je át lett irányítva az stdout-ra, így a `tee` képes volt azt egy csővezeték segítségével rögzíteni, megjeleníteni a képernyőn és elmenteni a `log.txt` fájlba. Ilyen esetekben hasznos lehet a hibaüzeneteket elmenteni a későbbi ellenőrzéshez.

Parancshelyettesítés

Egy másik módszer egy parancs kimenetének rögzítésére a *command substitution* (parancshelyettesítés). Ha egy parancsot backtickek (aposztrófok) közé helyezünk, a Bash helyettesíti azt a standard kimenetével. A következő példa azt mutatja, hogyan használhatjuk egy program stdoutját egy másik program argumentumaként:

```
$ mkdir `date +%Y-%m-%d`
$ ls
2019-09-05
```

A `date` program kimenete, az aktuális dátum *év-hónap-nap* formátumban, az `mkdir` programmal történő mappa létrehozásához használt argumentum volt. Ugyanilyen eredményt kapunk, ha a `$()`-t használjuk az backtickek helyett:

```
$ rmdir 2019-09-05
$ mkdir $(date +%Y-%m-%d)
$ ls
2019-09-05
```

Ugyanez a módszer használható egy parancs kimenetének változóként való tárolására is:

```
$ OS=`uname -o`
$ echo $OS
GNU/Linux
```

Az `uname -o` parancs kiadja az aktuális operációs rendszer általános nevét, amelyet az `OS` munkamenetváltozóban tároltunk. Egy parancs kimenetének változóhoz rendelése nagyon hasznos a szkriptekben, lehetővé téve az adatok sokféleképpen történő tárolását és kiértékelését.

A helyettesített parancs által generált kimenettől függően előfordulhat, hogy a beépített parancshelyettesítés nem megfelelő. Kifinomultabb módszer egy program kimenetének egy másik program argumentumaként való felhasználására a `xargs` nevű köztes eszközt használja. A `xargs` program az stdin-en keresztül kapott tartalmat használja fel egy adott parancs futtatására, amelynek a tartalma az argumentuma. A következő példában az `xargs` a `identify` programot futtatja a `find` program által megadott argumentumokkal:

```
$ find /usr/share/icons -name 'debian*' | xargs identify -format "%f: %wx%h\n"
debian-swirl.svg: 48x48
debian-swirl.png: 22x22
debian-swirl.png: 32x32
debian-swirl.png: 256x256
debian-swirl.png: 48x48
debian-swirl.png: 16x16
debian-swirl.png: 24x24
debian-swirl.svg: 48x48
```

Az `identify` program az *ImageMagick* része, amely egy parancssori eszközkészlet a legtöbb képfájltípus vizsgálatára, konvertálására és szerkesztésére. A példában az `xargs` a `find` által felsorolt összes elérési utat argumentumként adta meg az `identify`-nak, amely ezután minden egyes fájlhoz a `-format` kapcsoló által előírt módon formázott információt jelenít meg. A példában a `find` által talált fájlok a Debian fájlrendszerben lévő disztribúciós logót tartalmazó képek. A `-format` az `identify` paramétere, nem az `xargs`-é.

Az `-n 1` opció a `xargs`-t arra utasítja, hogy az adott parancsot egyszerre csak egy

argumentummal futtassa. A példa esetében ahelyett, hogy a `find` által talált összes elérési utat átadnánk az `identify`-nak argumentumlistaként, az `xargs -n 1` használatával az `identify` parancsot minden egyes elérési útra külön-külön hajtjuk végre. Az `-n 2` használata az `identify` parancsot két útvonallal, az `-n 3` parancsot három útvonallal és így tovább. Hasonlóképpen, amikor a `xargs` többsoros tartalmakat dolgoz fel — mint a `find` által megadott input esetében — a `-L` opcióval korlátozható, hogy hány sor kerüljön argumentumként felhasználásra egy parancs végrehajtásakor.

NOTE A `xargs` használata a `-n 1` vagy `-L 1` opcióval a `find` által generált kimenet feldolgozásához szükségtelen lehet. A `find` parancsnak van `-exec` opciója egy adott parancs futtatására minden egyes keresési eredményre.

Ha az elérési utak szóközöket tartalmaznak, fontos, hogy a `find`-t a `-print0` opcióval futtassuk. Ez az opció utasítja a `find`-t, hogy minden egyes bejegyzés között null karaktert használjon, hogy a listát az `xargs` helyesen tudja elemezni (a kimenet el lett nyomva):

```
$ find . -name '*avi' -print0 -o -name '*mp4' -print0 -o -name '*mkv' -print0 | xargs -0 du
| sort -n
```

A `-0` opció azt mondja a `xargs`-nak, hogy a null karaktert kell elválasztóként használni. Így a `find` által megadott fájl elérési útvonalakat helyesen elemzi, még akkor is, ha üres vagy más speciális karaktereket tartalmaznak. Az előző példa azt mutatja, hogy a `du` paranccsal hogyan lehet minden megtalált fájl lemezhasználatát megtudni, majd az eredményeket méret szerint rendezni. A tömörség kedvéért a kimenetet elnyomtuk. Vegyük figyelembe, hogy minden keresési feltételhez szükséges a `find`-hez a `-print0` kapcsolót megadni.

Alapértelmezés szerint az `xargs` a végrehajtott parancs argumentumait az utolsó helyre helyezi. Ennek a viselkedésnek a megváltoztatásához a `-I` opciót kell használni:

```
$ find . -mindepth 2 -name '*avi' -print0 -o -name '*mp4' -print0 -o -name '*mkv' -print0 |
xargs -0 -I PATH mv PATH ./
```

Az utolsó példában a `find` által talált minden fájl az aktuális mappába kerül. Mivel a forrás elérési útvonal(aka)t a cél elérési útvonal előtt közölni kell az `mv`-vel, az `xargs` opció `-I`-jének adunk egy helyettesítő kifejezést, amely aztán megfelelően az `mv` mellé kerül. Mivel a null karaktert használjuk elválasztóként, a helyettesítő kifejezést nem szükséges idézőjelekkel körülvenni.

Gyakorló feladatok

1. Praktikus az automatizált szkriptek által végrehajtott műveletek végrehajtási dátumának elmentése. A `date +%Y-%m-%d` parancs az aktuális dátumot mutatja *év-hónap-nap* formátumban. Hogyan lehet egy ilyen parancs kimenetét egy `TODAY` nevű shell változóban tárolni parancshelyettesítéssel?

2. Hogyan küldhetjük el a `TODAY` változó tartalmát az `echo` parancs segítségével a `sed s/-/./g` parancs standard bemenetére?

3. Hogyan lehet a `date +%Y-%m-%d%d` parancs kimenete használható Here stringként a `sed s/-/./g` parancshoz?

4. A `convert image.jpeg -resize 25% small/image.jpeg` parancs létrehozza a `image.jpeg` egy kisebb változatát, és az így kapott képet egy hasonló nevű fájlban helyezi el a `small` almappában. Az `xargs` használatával hogyan lehet ugyanazt a parancsot végrehajtani a `filelist.txt` fájlban felsorolt minden képre?

Gondolkodtató feladatok

1. Egy egyszerű mentési rutin rendszeresen létrehoz egy imaget a `/dev/sda1` partícióról a `dd < /dev/sda1 > sda1.img` címmel. A későbbi adatintegritás-ellenőrzések elvégzéséhez a rutin egy SHA1 hash-t is generál a fájlról a `sha1sum < sda1.img > sda1.sha1` paranccsal. A csővezetékek és a `tee` parancs hozzáadásával hogyan lehetne ezt a két parancsot eggyé kombinálni?

2. A `tar` parancs sok fájl egyetlen fájlba történő archiválására szolgál, megőrizve a mappaszerkezetet. A `-T` opció lehetővé teszi az archiválandó elérési utakat tartalmazó fájl megadását. Például a `find /etc -type f | tar -cJ -f /srv/backup/etc.tar.xz -T -T -` egy tömörített tar fájlt hoz létre `etc.tar.xz` néven a `find` parancs által megadott listából (a `-T -` opció a szabványos bemenetet jelöli az elérési útvonalak listájaként). A szóközöket tartalmazó elérési utak miatti esetleges elemzési hibák elkerülése érdekében milyen parancsbeállításoknak kell jelen lenniük a `find` és a `tar` parancsoknál?

3. Ahelyett, hogy új távoli shell munkamenetet nyitna, az `ssh` parancs egyszerűen végrehajtja az argumentumként megadott parancsot: `ssh user@storage "remote command"`. Mivel az `ssh` azt is lehetővé teszi, hogy egy helyi program standard kimenetét átirányítsuk a távoli program standard bemenetére, hogyan tudna a `cat` parancs egy helyi `etc.tar.gz` nevű fájlt a `/srv/backup/etc.tar.tar.gz`-be a `user@storage`-nél az `ssh`-n keresztül átvezetni?

Összefoglalás

Ez a lecke a Linux által alkalmazott hagyományos, folyamatok közötti kommunikációs technikákat tárgyalja. A *command pipelining* (parancs csővezetékezés) egyirányú kommunikációs csatornát hoz létre két folyamat között, a *command substitution* (parancshelyettesítés) pedig lehetővé teszi, hogy egy folyamat kimenetét egy shell-változóban tároljuk. A lecke a következő lépéseken ment keresztül:

- Hogyan lehet a *pipes* (csővezetékek) segítségével egy folyamat kimenetét egy másik folyamat bemenetére továbbítani.
- A `tee` és `xargs` parancsok célja.
- Hogyan lehet egy folyamat kimenetét a *command substitution* (parancshelyettesítés) segítségével rögzíteni, egy változóban tárolni vagy közvetlenül egy másik parancs paramétereként használni.

A tárgyalt parancsok és eljárások a következők voltak:

- Parancs csővezetékezés a `|`-vel.
- Parancshelyettesítés backtickekkel és a `$()`.
- A `tee`, `xargs` és a `find` parancsok.

Válaszok a gyakorló feladatokra

1. Praktikus az automatizált szkriptek által végrehajtott műveletek végrehajtási dátumának elmentése. A `date +%Y-%m-%d` parancs az aktuális dátumot mutatja *év-hónap-nap* formátumban. Hogyan lehet egy ilyen parancs kimenetét egy `TODAY` nevű shell változóban tárolni parancshelyettesítéssel?

   ```
   $ TODAY=`date +%Y-%m-%d`
   ```

   vagy

   ```
   $ TODAY=$(date +%Y-%m-%d)
   ```

2. Hogyan küldhetjük el a `TODAY` változó tartalmát az `echo` parancs segítségével a `sed s/-/./g` parancs standard bemenetére?

   ```
   $ echo $TODAY | sed s/-/./g
   ```

3. Hogyan lehet a `date +%Y-%m-%d%d` parancs kimenete használható Here stringként a `sed s/-/./g` parancshoz?

   ```
   $ sed s/-/./g <<< `date +%Y-%m-%d`
   ```

   vagy

   ```
   $ sed s/-/./g <<< $(date +%Y-%m-%d)
   ```

4. A `convert image.jpeg -resize 25% small/image.jpeg` parancs létrehozza a `image.jpeg` egy kisebb változatát, és az így kapott képet egy hasonló nevű fájlban helyezi el a `small` almappában. A `xargs` használatával hogyan lehet ugyanazt a parancsot végrehajtani a `filelist.txt` fájlban felsorolt minden képre?

   ```
   $ xargs -I IMG convert IMG -resize 25% small/IMG < filelist.txt
   ```

   vagy

```
$ cat filelist.txt | xargs -I IMG convert IMG -resize 25% small/IMG
```

Válaszok a gondolkodtató feladatokra

1. Egy egyszerű mentési rutin rendszeresen létrehoz egy képet a `/dev/sda1` partícióról a `dd < /dev/sda1 > sda1.img` címmel. A későbbi adatintegritás-ellenőrzések elvégzéséhez a rutin egy SHA1 hash-t is generál a fájlról a `sha1sum < sda1.img > sda1.sha1` paranccsal. A csővezetékek és a `tee` parancs hozzáadásával hogyan lehetne ezt a két parancsot eggyé kombinálni?

```
# dd < /dev/sda1 | tee sda1.img | sha1sum > sda1.sha1
```

2. A `tar` parancs sok fájl egyetlen fájlba történő archiválására szolgál, megőrizve a mappaszerkezetet. A `-T` opció lehetővé teszi az archiválandó elérési utakat tartalmazó fájl megadását. Például a `find /etc -type f | tar -cJ -f /srv/backup/etc.tar.xz -T -T -` egy tömörített tar fájlt hoz létre `etc.tar.xz` néven a `find` parancs által megadott listából (a `-T -` opció a szabványos bemenetet jelöli az elérési útvonalak listájaként). A szóközöket tartalmazó elérési utak miatti esetleges elemzési hibák elkerülése érdekében milyen parancsbeállításoknak kell jelen lenniük a `find` és a `tar` parancsoknál?

   A `-print0` és `--null` kapcsolókkal:

```
$ find /etc -type f -print0 | tar -cJ -f /srv/backup/etc.tar.xz --null -T -
```

3. Ahelyett, hogy új távoli shell munkamenetet nyitna, az `ssh` parancs egyszerűen végrehajtja az argumentumként megadott parancsot: `ssh user@storage "remote command"`. Mivel az `ssh` azt is lehetővé teszi, hogy egy helyi program standard kimenetét átirányítsuk a távoli program standard bemenetére, hogyan tudna a `cat` parancs egy helyi `etc.tar.gz` nevű fájlt a `/srv/backup/etc.tar.tar.gz`-be a `user@storage`-nél az `ssh`-n keresztül átvezetni?

```
$ cat etc.tar.gz | ssh user@storage "cat > /srv/backup/etc.tar.gz"
```

   vagy

```
$ ssh user@storage "cat > /srv/backup/etc.tar.gz" < etc.tar.gz
```

103.5 Folyamatok létrehozása, monitorozása és megszakítása

Hivatkozás az LPI célkitűzésre

LPIC-1 v5, Exam 101, Objective 103.5

Súlyozás

4

Kulcsfontosságú ismeretek

- Futtasson feladatokat előtérben és háttérben.
- Jelzés egy programnak, hogy a kijelentkezés után is folytassa a futást.
- Aktív folyamatok figyelése.
- Folyamatok kijelölése és rendezése megjelenítésre.
- Jelzések küldése a folyamatoknak.

A használt fájlok, kifejezések és segédprogramok listája

- `&`
- `bg`
- `fg`
- `jobs`
- `kill`
- `nohup`
- `ps`
- `top`
- `free`

- `uptime`
- `pgrep`
- `pkill`
- `killall`
- `watch`
- `screen`
- `tmux`

103.5 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	103 GNU és Unix parancsok
Fejezet:	103.5 Folyamatok létrehozása, monitorozása és megszakítása
Lecke:	1/2

Bevezetés

Minden alkalommal, amikor meghívunk egy parancsot, egy vagy több folyamat (processz) indul el. Egy jól képzett rendszergazdának nemcsak létrehoznia kell a folyamatokat, hanem képesnek kell lennie arra is, hogy nyomon kövesse őket, és szükség esetén különböző típusú jelzéseket küldjön nekik. Ebben a leckében a feladatvezérlést és a processzek felügyeletét fogjuk megvizsgálni.

Job Control

A *Jobok* (munkák) olyan folyamatok, amelyeket interaktívan indítottak el egy terminálon keresztül, majd a háttérbe küldték őket, és még nem fejezték be a végrehajtásukat. A Linux-rendszerben lévő aktív jobokról (és azok állapotáról) a `jobs` futtatásával tájékozódhatunk:

```
$ jobs
```

A fenti `jobs` parancs nem adott ki semmilyen kimenetet, ami azt jelenti, hogy jelenleg nincsenek

aktív jobok. Hozzuk létre az első jobunkat egy olyan parancs futtatásával, amelynek a végrehajtása némi időt vesz igénybe (a `sleep` parancs a `60` paraméterrel), és — futás közben — nyomjuk meg a `Ctrl` + `Z` billentyűt:

```
$ sleep 60
^Z
[1]+  Stopped                 sleep 60
```

A parancs végrehajtása leállt (vagy inkább felfüggesztésre került), és a parancssor ismét elérhetővé vált. Másodszor is megnézhetjük a jobokat, és most már a *felfüggesztett* jobot fogjuk látni:

```
$ jobs
[1]+  Stopped                 sleep 60
```

Nézzük meg, mit jelent ez a kimenet:

`[1]`

Ez a szám a job azonosítója, és felhasználható — egy százalékjellel (%) megelőzve — a job állapotának megváltoztatására az `fg`, `bg` és `kill` segédprogramok segítségével (amint azt később be is mutatjuk).

`+`

A plusz jel az aktuális, alapértelmezett jobot jelzi (vagyis az utolsót, amelyet felfüggesztettek vagy háttérbe küldtek). Az előző jobot mínusz jellel (`-`) jelöli. A többi korábbi jobot nem jelöli.

`Stopped`

A job állapotának leírása.

`sleep 60`

A parancs vagy maga a job.

Az `-l` kapcsolóval a jobok a processz azonosítóját (PID) is megjelenítik közvetlenül az állapot előtt:

```
$ jobs -l
[1]+  1114 Stopped                 sleep 60
```

A jobok további lehetséges kapcsolói a következők:

`-n`

Csak azokat a folyamatokat listázza, amelyek állapota megváltozott az utolsó értesítés óta. A lehetséges státuszok a következők: `Running` (futó, folyamatban lévő), `Stopped` (megállt), `Terminated` (terminált) vagy `Done` (kész).

`-p`

A processzek azonosítóinak listája.

`-r`

Csak a futó jobokat listázza.

`-s`

Csak a leállított (vagy felfüggesztett) jobokat listázza.

NOTE Ne feledjük, egy jobnak van *job ID*-ja és *process ID*-ja (PID) is.

A jobok specifikációja

A `jobs` parancsnak, valamint más segédprogramoknak, mint például az `fg`, `bg` és `kill` (amelyeket a következő szakaszban fogunk tárgyalni), szüksége van egy specifikációra (vagy `jobspec`) ahhoz, hogy egy adott jobra hatni tudjon. Mint az imént láttuk, ez lehet — és általában ez is — a job azonosítója, amelyet `%` előz meg. Azonban más specifikációk is lehetségesek. Nézzük meg ezeket:

`%n`

Az a job, amelynek az azonosítója `n`:

```
$ jobs %1
[1]+  Stopped                 sleep 60
```

`%str`

Az a job, amelynek parancsa `str`-el kezdődik:

```
$ jobs %sl
[1]+  Stopped                 sleep 60
```

`%?str`

Az a job, amelynek a parancsa tartalmazza az `str`-t:

```
$ jobs %?le
```

```
[1]+  Stopped                 sleep 60
```

%+ vagy %%

Az aktuális job (az, amely a háttérben lett elindítva vagy az előtérből felfüggesztve):

```
$ jobs %+
[1]+  Stopped                 sleep 60
```

%-

Az előző job (ami %+ volt, az aktuális előtti):

```
$ jobs %-
[1]+  Stopped                 sleep 60
```

A mi esetünkben, mivel csak egy job van, ez a jelenlegi és az előző is.

Job státusz: felfüggesztés, előtér és háttér

Ha egy job a háttérben van vagy felfüggesztésre került, három dolog közül bármit tehetünk vele:

1. Előtérbe hozni az fg-vel:

   ```
   $ fg %1
   sleep 60
   ```

   Az fg a megadott jobot előtérbe helyezi, és aktuális jobbá teszi. Most megvárhatjuk, amíg befejeződik, vagy újra leállíthatjuk a Ctrl + Z paranccsal, vagy megszüntethetjük a Ctrl + C paranccsal.

2. Háttérbe küldeni a bg paranccsal:

   ```
   $ bg %1
   [1]+ sleep 60 &
   ```

   Ha a háttérben van, a job az fg paranccsal visszahozható az előtérbe, vagy megszüntethető (lsd. lentebb). Figyeljük meg az írásjelet (&), ami azt jelenti, hogy a jobot a háttérbe küldtük. Ami azt illeti, az amperjelet használhatjuk arra is, hogy egy folyamatot közvetlenül a háttérben indítsunk el:

```
$ sleep 100 &
[2] 970
```

Az új job azonosítójával (`[2]`) együtt most már a folyamat azonosítóját is megkapjuk (`970`). Most már mindkét feladat fut a háttérben:

```
$ jobs
[1]-  Running                 sleep 60 &
[2]+  Running                 sleep 100 &
```

Egy kicsivel később az első job befejezi a futást:

```
$ jobs
[1]-  Done                    sleep 60
[2]+  Running                 sleep 100 &
```

3. Terminálhatjuk a `SIGTERM` szignálon keresztül a `kill` segítségével:

```
$ kill %2
```

Hogy biztosak legyünk benne, hogy a job terminálódott, futtassuk újra a `jobs`-t:

```
$ jobs
[2]+  Terminated              sleep 100
```

NOTE Ha nem adunk meg jobot, akkor az `fg` és `bg` az aktuális, alapértelmezett jobot értelmezve fog működni. A `kill`-nek azonban mindig szüksége van a job megadására.

Önálló jobok: `nohup`

Az előző szakaszokban látott jobok mindegyike az őket előhívó felhasználó munkamenetéhez kapcsolódott. Ez azt jelenti, hogy ha a munkamenet megszűnik, a jobok is eltűnnek. Lehetőség van azonban a jobok munkamenetről való leválasztására, és ezek így a munkamenet bezárása után is futtathatók. Ez a `nohup` (“no hangup”) paranccsal érhető el. A szintaxis a következő:

```
nohup COMMAND &
```

Ne feledjük, hogy a `&` a folyamatot a háttérbe küldi, és felszabadítja a terminált, amelyben éppen dolgozunk!

Válasszuk le a `ping localhost` háttérjobot az aktuális munkamenetről:

```
$ nohup ping localhost &
[1] 1251
$ nohup: ignoring input and appending output to 'nohup.out'
^C
```

A kimenet mutatja a feladat azonosítóját (`[1]`) és a PID-t (`1251`), majd egy üzenetet a `nohup.out` fájlról. Ez az alapértelmezett fájl, ahová az `stdout`-ot és az `stderr`-t mentjük. Most megnyomhatjuk a `Ctrl` + `C` billentyűkombinációt a parancssor felszabadításához, bezárhatjuk a munkamenetet, indíthatunk egy másikat `root`-ként, és a `tail -f` paranccsal ellenőrizhetjük, hogy a parancs fut-e, és a kimenet az alapértelmezett fájlba íródik-e:

```
$ exit
logout
$ tail -f /home/carol/nohup.out
64 bytes from localhost (::1): icmp_seq=3 ttl=64 time=0.070 ms
64 bytes from localhost (::1): icmp_seq=4 ttl=64 time=0.068 ms
64 bytes from localhost (::1): icmp_seq=5 ttl=64 time=0.070 ms
^C
```

TIP Az alapértelmezett `nohup.out` helyett a `nohup ping localhost > /path/to/your/file &` paranccsal megadhatjuk a kívánt kimeneti fájlt.

Ha meg akarjuk szüntetni a folyamatot (kill), meg kell adnunk a PID-jét:

```
# kill 1251
```

Folyamatok monitorozása

A folyamat (processz) vagy feladat (task) egy futó program példánya. Ez azt jelenti, hogy minden alkalommal új folyamatokat hozunk létre, amikor parancsokat írunk be a terminálba.

A `watch` parancs periodikusan (alapértelmezés szerint 2 másodpercenként) végrehajt egy programot, és lehetővé teszi, hogy *megfigyeljük* a program kimenetének időbeli változását. Például a `watch uptime` parancs beírásával nyomon követhetjük, hogyan változik a terhelés átlaga, ha több folyamatot futtatunk:

```
Every  2.0s: uptime          debian: Tue Aug 20 23:31:27 2019

 23:31:27 up 21 min,  1 user,  load average: 0.00, 0.00, 0.00
```

A parancs addig fut, amíg meg nem szakad, ezért a `Ctrl` + `C` billentyűkombinációval kell leállítanunk. Kimenetként két sort kapunk: az első a `watch`-nak felel meg, és megmondja, hogy milyen gyakran fog futni a parancs (`Every 2.0s: uptime`), milyen parancsot/programot kell figyelni (`uptime`), valamint a hostnevet és a dátumot (`debian: Tue Aug 20 23:31:27 2019`). A második kimeneti sor az uptime-ról szól, és tartalmazza az időt (`23:31:27`), azt, hogy mennyi ideje megy a rendszer (`up 21 min`), az aktív felhasználók számát (`1 user`) és a rendszer átlagos terhelését vagy a végrehajtásban vagy várakozó állapotban lévő folyamatok számát az elmúlt 1, 5 és 15 percben (`load average: 0.00, 0.00, 0.00, 0.00`).

Hasonlóképpen ellenőrizhetjük a memóriahasználatot az új folyamatok létrehozásakor a `watch free` paranccsal:

```
Every  2.0s: free            debian: Tue Aug 20 23:43:37 2019

 23:43:37 up 24 min,  1 user,  load average: 0.00, 0.00, 0.00
              total        used        free      shared  buff/cache   available
Mem:       16274868      493984    14729396       35064     1051488    15462040
Swap:      16777212           0    16777212
```

A `watch` frissítési intervallumának módosításához használjuk az `-n` vagy `--interval` kapcsolókat és a másodpercek számát, csakúgy, mint ebben a példában:

```
$ watch -n 5 free
```

Mostantól a `free` parancs 5 másodpercenként fog futni.

A `uptime`, `free` és `watch` opciókkal kapcsolatos további információkért olvassuk el a man oldalaikat.

NOTE Az `uptime` és `free` által szolgáltatott információkat a `top` és `ps` átfogóbb eszközök is tartalmazzák (lsd. lentebb).

Szignálok küldése a folyamatoknak: `kill`

Minden egyes folyamat egyedi processz azonosítóval vagy PID-del rendelkezik. Egy processz PID-jét a `pgrep` paranccsal, majd a folyamat nevének megadásával lehet megtudni:


```
$ pgrep sleep
1201
```

NOTE Egy folyamat azonosítója a `pidof` paranccsal is kideríthető (pl. `pidof sleep`).

A `pgrep`-hez hasonlóan a `pkill` parancs eliminál egy folyamatot a neve alapján:

```
$ pkill sleep
[1]+  Terminated              sleep 60
```

Ugyanazon folyamat több példányának megállításához a `killall` parancs használható:

```
$ sleep 60 &
[1] 1246
$ sleep 70 &
[2] 1247
$ killall sleep
[1]-  Terminated              sleep 60
[2]+  Terminated              sleep 70
```

Mind a `pkill`, mind a `killall` ugyanúgy működik, mint a `kill`, mivel egy befejező jelet küld a megadott folyamat(ok)nak. Ha nem adunk meg jelet, akkor az alapértelmezett `SIGTERM` jelet küldjük. A `kill` azonban csak egy feladat vagy egy folyamat azonosítóját veszi fel argumentumként.

A szignálokat a következő módokon lehet megadni:

- Név:

```
$ kill -SIGHUP 1247
```

- Szám:

```
$ kill -1 1247
```

- Kapcsoló:

```
$ kill -s SIGHUP 1247
```

Ahhoz, hogy a `kill` hasonlóan működjön, mint a `pkill` vagy `killall` (és megspóroljuk magunknak a PID-ek kiderítésének parancsát), használhatjuk a parancshelyettesítést:

```
$ kill -1 $(pgrep sleep)
```

Mint azt már tudjuk, az alternatív szintaxis a `kill -1 `pgrep sleep``.

TIP Az összes `kill` jelzés és kódjaik teljes listájának megszerzéséhez írjuk be a terminálba a `kill -l` parancsot. A `-KILL` (`-9` vagy `-s KILL`) segítségével a lázadó folyamatokat lehet terminálni, ha bármely más szignál sikertelen.

`top` és `ps`

A folyamatok monitorozásában két felbecsülhetetlen értékű eszköz a `top` és a `ps`. Míg az előbbi dinamikusan, az utóbbi statikusan állít elő kimenetet. Mindenesetre mindkettő kiváló segédprogram ahhoz, hogy átfogó képet kapjunk a rendszerben lévő összes folyamatról.

A `top` használata

A `top` használatához csak írjuk be, hogy `top`:

```
$ top

top - 11:10:29 up  2:21,  1 user,  load average: 0,11, 0,20, 0,14
Tasks:  73 total,   1 running,  72 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0,0 us,  0,3 sy,  0,0 ni, 99,7 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem :  1020332 total,   909492 free,    38796 used,    72044 buff/cache
KiB Swap:  1046524 total,  1046524 free,        0 used.   873264 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
   436 carol     20   0   42696   3624   3060 R  0,7  0,4   0:00.30 top
     4 root      20   0       0      0      0 S  0,3  0,0   0:00.12 kworker/0:0
   399 root      20   0   95204   6748   5780 S  0,3  0,7   0:00.22 sshd
     1 root      20   0   56872   6596   5208 S  0,0  0,6   0:01.29 systemd
     2 root      20   0       0      0      0 S  0,0  0,0   0:00.00 kthreadd
     3 root      20   0       0      0      0 S  0,0  0,0   0:00.02 ksoftirqd/0
     5 root       0 -20       0      0      0 S  0,0  0,0   0:00.00 kworker/0:0H
     6 root      20   0       0      0      0 S  0,0  0,0   0:00.00 kworker/u2:0
     7 root      20   0       0      0      0 S  0,0  0,0   0:00.08 rcu_sched
     8 root      20   0       0      0      0 S  0,0  0,0   0:00.00 rcu_bh
     9 root      rt   0       0      0      0 S  0,0  0,0   0:00.00 migration/0
    10 root       0 -20       0      0      0 S  0,0  0,0   0:00.00 lru-add-drain
```

```
    ( . . . )
```

A `top` lehetővé teszi a felhasználó számára az interakciót. Alapértelmezés szerint a kimenetet az egyes folyamatok által felhasznált CPU-idő százalékos aránya szerint rendezi csökkenő sorrendben. Ez a viselkedés a következő billentyűkkel módosítható a `top`-ban:

`M`

Rendezés *memóriahasználat* alapján.

`N`

Rendezés a folyamat *azonosítószáma* (ID) alapján.

`T`

Rendezés futási *idő* alapján.

`P`

Rendezés a CPU használat *százalékos aránya* alapján.

TIP A csökkenő/növekvő sorrend közötti váltáshoz csak nyomjuk meg az `R` billentyűt.

További érdekes kulcsok a `top`-pal való interakcióhoz:

`?` or `h`

Segítség.

`k`

Egy folyamat terminálása. A `top` várja a terminálandó folyamat `PID` azonosítóját, valamint a küldendő szignált (alapértelmezés szerint `SIGTERM` vagy `15`).

`r`

Egy folyamat prioritásának módosítása (`renice`). A `top` várja a `nice` értéket. A lehetséges értékek -20 és 19 között mozognak, de csak a szuperfelhasználó (`root`) állíthatja be olyan értékre, amely negatív vagy alacsonyabb az aktuálisnál.

`u`

Egy adott felhasználó folyamatainak listázása (alapértelmezés szerint az összes felhasználó folyamatai megjelennek).

`c`

Megjeleníti a programok abszolút elérési útvonalait, és különbséget tesz a felhasználói és a kernelspace folyamatok között (szögletes zárójelben).

`V`

A folyamatok erdő/hierarchia nézete.

`t` és `m`

A CPU- és memóriaértékek megjelenésének megváltoztatása egy négylépcsős ciklusban: az első két megnyomásnál progress barok jelennek meg, a harmadik elrejti a progress bart, a negyedik pedig visszahozza azt.

`W`

A konfigurációs beállítások mentése a `~/.toprc`-be.

TIP

A `top` egy szebb és felhasználóbarátabb változata a `htop`. Egy másik — talán aprólékosabb — alternatíva az `atop`. Ha még nincsenek telepítve, használjuk a csomagkezelőt a telepítésükhöz és próbáljuk ki őket.

A `top` kimenetének magyarázata

A `top` kimenete két területre van osztva: az *összefoglaló területre* (summary area) és a *feladat területre* (task area).

A `top` összefoglaló területe

Az *összefoglaló terület* az öt felső sorból áll, és a következő információkat tartalmazza:

- `top - 11:10:29 up 2:21, 1 user, load average: 0,11, 0,20, 0,14`
  - aktuális idő (24 órás formátumban): `11:20:29`
  - uptime (mióta fut a rendszer): `up 2:21`
  - a bejelentkezett felhasználók száma és a CPU átlagos terhelése az elmúlt 1, 5, illetve 15 percben: `load average: 0,11, 0,20, 0,14`
- `Tasks: 73 total, 1 running, 72 sleeping, 0 stopped, 0 zombie` (információk a folyamatokról)
  - az aktív módú processzek száma: `73 total`
  - running (amelyek épp futtatás alatt állnak): `1 running`
  - sleeping (amelyek várnak a futtatás folytatására): `72 sleeping`
  - stopped (megállítva egy job szignál által): `0 stopped`
  - zombie (amelyek befejezték a futást, de még mindig a szülő processzre várnak, hogy eltávolítsa őket a táblából): `0 zombie`

- `%Cpu(s): 0,0 us, 0,3 sy, 0,0 ni, 99,7 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st` (a CPU-idő százalékos aránya)
  - felhasználói processzek: `0,0 us`
  - rendszer/kernel folyamatok: `0,4 sy`
  - folyamatok *nice* értékre állítva — minél szebb az érték, annál alacsonyabb a prioritás: `0,0 ni`
  - semmi — idle CPU-idő: `99,7 id`
  - I/O műveletekre váró processzek: `0,0 wa`
  - hardveres megszakításokat kiszolgáló folyamatok - perifériák, amelyek figyelmet igénylő jeleket küldenek a processzornak.: `0,0 hi`
  - szoftveres megszakításokat kiszolgáló folyamatok: `0,0 si`
  - más virtuális gépek feladatait kiszolgáló folyamatok egy virtuális környezetben, így időt lopva: `0,0 st`
- `KiB Mem : 1020332 total, 909492 free, 38796 used, 72044 buff/cache` (memóriainformáció kilobájtokban)
  - a memória teljes mennyisége: `1020332 total`
  - használatlan memória: `909492 free`
  - használt memória: `38796 used`
  - a pufferelt és gyorsítótárazott memória a túlzott lemezelérés elkerülése érdekében: `72044 buff/cache`

    Vegyük észre, hogy a `total` a másik három érték — `free`, `used` és `buff/cache` — összege (durván 1 GB a mi esetünkben).
- `KiB Swap: 1046524 total, 1046524 free, 0 used. 873264 avail Mem` (swap információ kilobájtokban)
  - a swap terület teljes mérete: `1046524 total`
  - a használatlan swap mérete: `1046524 free`
  - használt swap terület: `0 used`
  - a swap-memória mennyisége, amelyet a folyamatok számára ki lehet osztani anélkül, hogy további lapozást okozna: `873264 avail Mem`

A `top` feladat területe: mezők és oszlopok

Az *összefoglaló terület* alatt található a *feladat terület*, amely egy sor *mezőt* és *oszlopot* tartalmaz, amelyek a futó folyamatokra vonatkozó információkat jelentik:

`PID`

Processz azonosító.

`USER`

A felhasználó, aki kiadta a parancsot, ami generálta a processzt.

`PR`

A processz prioriítása a kernel számára.

`NI`

A folyamat nice értéke. Az alacsonyabb értékek magasabb prioritást élveznek, mint a magasabbak.

`VIRT`

A folyamat által használt memória teljes mennyisége (beleértve a swapot is).

`RES`

A processz által használt RAM memória.

`SHR`

A folyamat más folyamatokkal megosztott memóriája.

`S`

A processz státusza. A lehetséges értékek közé tartoznak: `S` (megszakítható sleep (alvó) — egy eseményre vár a befejezéshez), `R` (runnable (futtatható) — vagy épp végrehajtás alatt van, vagy a végrehajtásra váró sorban van) vagy `Z` (zombie — befejezett gyermekfolyamatok, amelyek adatszerkezeteit még nem távolították el a folyamat táblából).

`%CPU`

A processz által használt CPU százalékos aránya.

`%MEM`

A processz által használta RAM százalékos aránya, azaz a `RES` érték százalékban kifejezve.

`TIME+`

A folyamat tevékenységének teljes időtartama.

`COMMAND`

A parancs/program neve, ami generálta a folyamatot.

Folyamatok statikus megtekintése: `ps`

Ahogy fentebb említettük, a `ps` a folyamatok pillanatfelvételét (snapshot) mutatja. Ha az összes terminállal (tty) rendelkező folyamatot látni szeretnénk, írjuk be a `ps a` parancsot:

```
$ ps a
  PID TTY      STAT   TIME COMMAND
  386 tty1     Ss+    0:00 /sbin/agetty --noclear tty1 linux
  424 tty7     Ssl+   0:00 /usr/lib/xorg/Xorg :0 -seat seat0 (...)
  655 pts/0    Ss     0:00 -bash
 1186 pts/0    R+     0:00 ps a
 (...)
```

A `ps` kapcsolóinak szintaxisa és kimenete

A kapcsolókat illetően a `ps` három különböző stílust fogadhat el: BSD, UNIX és GNU. Nézzük meg, hogyan működnének ezek a stílusok, ha egy adott folyamat azonosítójáról adunk információt:

BSD

A kapcsolóknál nincs kötőjel:

```
$ ps p 811
  PID TTY      STAT   TIME COMMAND
  811 pts/0    S      0:00 -su
```

UNIX

A kapcsolóknál van kötőjel:

```
$ ps -p 811
  PID TTY          TIME CMD
  811 pts/0    00:00:00 bash
```

GNU

A kapcsolóknál két kötőjel van:

```
$ ps --pid 811
```

```
  PID TTY          TIME CMD
  811 pts/0    00:00:00 bash
```

Mindhárom esetben a `ps` arról a folyamatról ad információt, amelynek `PID`-je a `811` — ebben az esetben a `bash`.

Hasonlóképpen használhatjuk a `ps` parancsot egy adott felhasználó által indított folyamatok megkeresésére:

- `ps U carol` (BSD)
- `ps -u carol` (UNIX)
- `ps --user carol` (GNU)

Ellenőrizzük a `carol` által indított processzeket:

```
$ ps U carol
  PID TTY      STAT   TIME COMMAND
  811 pts/0    S      0:00 -su
  898 pts/0    R+     0:00 ps U carol
```

Két folyamatot indított el: `bash` (`-su`) és `ps` (`ps U carol`). A `STAT` oszlop árulja el a folyamatok státuszát (lsd. lentebb).

A `ps` lehetőségeit egyesítve a legjobbat hozhatjuk ki belőle. Egy nagyon hasznos parancs (amely a `top` parancshoz hasonló kimenetet ad) a `ps aux` (BSD stílusban). Ebben az esetben az összes shell (nem csak az aktuális) folyamatai megjelennek. A kapcsolók jelentése a következő:

a

Megmutatja azokat a folyamatokat, amelyek egy `tty`-hez vagy terminálhoz kapcsolódnak.

u

Felhasználó-orientált formátum megjelenítése.

x

Megjeleníti azokat a folyamatokat, amelyek nem kapcsolódnak egy `tty`-hez vagy terminálhoz.

```
$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1 204504  6780 ?        Ss   14:04   0:00 /sbin/init
root         2  0.0  0.0      0     0 ?        S    14:04   0:00 [kthreadd]
```

```
root         3  0.0  0.0      0     0 ?        S    14:04   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   14:04   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S    14:04   0:00 [rcu_sched]
root         8  0.0  0.0      0     0 ?        S    14:04   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    14:04   0:00 [migration/0]
(...)
```

Nézzük az oszlopok értelmezését:

`USER`

A processz tulajdonosa.

`PID`

A processz azonosítója.

`%CPU`

A CPU-használat százalékos aránya.

`%MEM`

A fizikiális memória-használat százalékos aránya.

`VSZ`

A processz virtuális memóriája KiB-ban.

`RSS`

A processz által használt nem-lapozható fizikális memória KiB-ban.

`TT`

A processzt kontrolálló terminál (`tty`).

`STAT`

A folyamat állapotát jelző kód. Az `S`, `R` és `Z` (amelyeket a `top` kimenetének leírásakor láttunk) mellett a következő értékek lehetségesek: `D` (megszakítás nélküli alvás — általában I/O-ra várva), `T` (leállítva — általában egy szignál által). Néhány extra módosító a következő: `<` (magas prioritású — nem `"kedves`" (nice) más folyamatokkal), `N` (alacsony prioritású — `"kedves`" más folyamatokkal), vagy `+` (az előtérben lévő folyamatcsoportban).

`STARTED`

A processz kezdetének időpontja.

`TIME`

Összesített CPU-idő.

`COMMAND`

Parancs, ami elindította a folyamatot.

Gyakorló feladatok

1. Az `oneko` egy kedves, vicces program, amely egy macskát jelenít meg, amely az egérkurzort üldözi. Ha még nincs telepítve, telepítsük a csomagkezelő segítségével! Ezt fogjuk használni a job kontroll (job control) tanulmányozására.
   - Indítsuk el a programot! Hogyan kell ezt megtenni?
   - Mozgassuk az egérkurzort, hogy lássuk, ahogy kergeti a macska. Most függesszük fel a processzt. Hogyan kell ezt megtenni? Mi a kimenet?
   - Nézzük meg, hogy mennyi jobunk van. Mit kell beírnunk? Mi a kimenet?
   - Most pedig küldjük a háttérbe a job azonosítójának megadásával. Mi a kimenet? Honnan tudhatjuk, hogy a job a háttérben fut?
   - Végül terminálju a jobot az ID megadásával. Mit kell beírnunk?
2. Fedezzük fel az *Apache HTTPD web server* (`apache2`) által indított összes folyamat PID-jeit két különböző paranccsal:
3. Fejezzük be az összes `apache2` folyamatot a PID-jük használata nélkül, két különböző paranccsal:
4. Tegyük fel, hogy az `apache2` összes példányát meg kell szüntetnünk, és nincs időnk kideríteni, hogy mi a PID-jük. Hogyan tudnánk ezt elérni a `kill` használatával az alapértelmezett `SIGTERM` szignállal egy sorban:
5. Indítsuk el a `top`-ot és tegyük meg a következőket:

- Nézzük meg a processzek forest (erdő) nézetét:

- A folyamatok teljes útvonalának megjelenítése, megkülönböztetve a felhasználói tér és a kerneltér között:

6. A `ps` használatával jelenítsük meg az *Apache HTTPD web server* felhasználó által indított összes folyamatot (`www-data`):
   - BSD szintaxis:

   - UNIX szintaxis:

   - GNU szintaxis:

Gondolkodtató feladatok

1. A `SIGHUP` szignál bizonyos daemonok újraindítására használható. Az *Apache HTTPD web server* esetében például a `SIGHUP` elküldése a szülőfolyamatnak (amelyet az `init` indít el) megöli a gyermekfolyamatokat. A szülő azonban újraolvassa a konfigurációs fájljait, újra megnyitja a naplófájlokat, és új gyermekeket indít. Végezzük el a következő feladatokat:
   - Indítsuk el a webszervert:
   - Tudjuk meg a szülőprocessz PID-jét:
   - Az Apache HTTPD webkiszolgáló újraindítása a `SIGHUP` szignál elküldésével a szülőfolyamatnak:
   - Ellenőrizzük, hogy a szülő nem terminálódott-e, és születtek-e új gyermekek:
2. Bár kezdetben statikus, a `ps` kimenete a `ps` és a `watch` kombinálásával dinamikussá *fordítható*. Monitorozzuk az *Apache HTTPD web server* új kapcsolatait! Az alább leírt feladatok elvégzése előtt ajánlott elolvasni a `MaxConnectionsPerChild` direktíva leírását a Apache MPM Common Directives oldalon.
   - Adjuk hozzá a `MaxConnectionsPerChild` direktívát `1` értékkel az `apache2` konfigurációs fájljához — *Debian* és Debian-alapú rendszerek esetén ez az `/etc/apache2/apache2.conf` fájl; *CentOS* család esetén az `/etc/httpd/conf/httpd.conf`. Ne felejtsük el újraindítani az `apache2`-t, hogy a változtatások életbe lépjenek!
   - Írjunk be egy parancsot, ami használja a `watch`, `ps` és `grep` parancsokat `apache2` kapcsolatok esetén!
   - Nyissuk meg a böngészőt vagy használjunk egy parancssori böngészőt, mint például a `lynx`, hogy létrehozzuk a kapcsolatot a webszerverrel az IP-címén keresztül! Mit figyelhetünk meg a `watch` kimenetén?

3. Amint azt már megtanultauk, a `top` alapértelmezés szerint a CPU-használat százalékos aránya szerint rendezi a feladatokat csökkenő sorrendben (a magasabb értékek vannak felül). Ez a viselkedés módosítható az `M` (memóriahasználat), `N` (folyamat egyedi azonosítója), `T` (futási idő) és `P` (CPU-idő százalékos aránya) interaktív kulcsokkal. A feladatlistát azonban a `top -o` kapcsolóval történő elindításával is tetszés szerint rendezhetjük (további információkért nézzük meg a top `man` oldalát). Most hajtsuk végre a következő feladatokat:
   - Indítsuk el a `top`-ot, hogy a feladatok memóriahasználat szerint legyenek rendezve:
   - Ellenőrizzük a memória oszlop kiemelésével, hogy a megfelelő parancsot írtuk-e be:
4. A `ps` szintén rendelkezik egy `o` kapcsolóval, amellyel megadhatjuk a megjeleníteni kívánt oszlopokat. Vizsgáljuk meg ezt a lehetőséget, és végezzük el a következő feladatokat:
   - Indítsuk el a `ps`-t úgy, hogy csak a *user*, a *percentage of memory used*, a *percentage of CPU time used* és a *full command* információk jelenjenek meg:
   - Most indítsuk el a `ps`-t úgy, hogy csak a felhasználó adatai és az általa használt programok neve jelenjen meg:

Összefoglalás

Ebben a leckében a *jobok*-ról és a *job control*-ról tanultunk. Fontos tények és fogalmak, amelyeket szem előtt kell tartanunk:

- A jobok olyan folyamatok, amelyek a háttérbe kerülnek.
- A *process ID* mellett a jobok létrehozásukkor egy *job ID* azonosítót is kapnak.
- A jobok vezérléséhez egy specifikációra (`jobspec`) van szükség.
- A jobok előtérbe hozhatók, háttérbe küldhetők, felfüggeszthetők és megszüntethetők (vagy *terminálhatók*).
- Egy job leválasztható a terminálról és a munkamenetről, amelyben létrehozták.

Hasonlóképpen, a *folyamatok* és a *folyamatfelügyelet* fogalmát is megvitattuk. A legfontosabb gondolatok a következők:

- A folyamatok futó programok.
- A folyamatok monitorozhatók.
- Különböző segédprogramok segítségével megtudhatjuk a folyamatok *process ID* azonosítóját, valamint szignálokat küldhetünk nekik, hogy befejezzük őket.
- A szignálokat meg lehet adni névvel (pl. `-SIGTERM`), számmal (pl. `-15`) vagy kapcsolóval (pl. `-s SIGTERM`).
- A `top` és a `ps` nagyon hatékonyak, ha folyamatok megfigyeléséről van szó. Az előbbi kimenete dinamikus és folyamatosan frissül; ezzel szemben a `ps` statikusan mutatja a kimenetet.

A leckében használt parancsok:

`jobs`
: Aktív jobok és állapotuk megjelenítése.

`sleep`
: Késleltetés egy meghatározott ideig.

`fg`
: Job előtérbe hozása.

`bg`
: Job háttérbe küldése.

`kill`

Job terminálása.

`nohup`

Job leválasztása a munkamenetről/terminálról.

`exit`

Az aktuális shell bezárása.

`tail`

A fájl legfrissebb sorainak megjelenítése.

`watch`

Egy parancs ismételt futtatása (alapértelmezés szerint 2 másodperces ciklus).

`uptime`

Megjeleníti, hogy mióta fut a rendszer, az aktuális felhasználók számát és a rendszer átlagos terhelését.

`free`

A memórihasználat megjelenítése.

`pgrep`

A folyamat azonosítójának keresése a név alapján.

`pidof`

A folyamat azonosítójának keresése a név alapján.

`pkill`

Szignál küldése a folyamatnak a neve alapján.

`killall`

Processz(ek) terminálása a név alapján.

`top`

Linux folyamatok megjelenítése.

`ps`

A jelenlegi folyamatok pillanatfelvétele.

Válaszok a gyakorló feladatokra

1. Az `oneko` egy kedves, vicces program, amely egy macskát jelenít meg, amely az egérkurzort üldözi. Ha még nincs telepítve, telepítsük a csomagkezelő segítségével! Ezt fogjuk használni a munkairányítás tanulmányozására.

   - Indítsuk el a programot! Hogyan kell ezt megtenni?

     Az `oneko` beírásával a terminálba.

   - Mozgassuk az egérkurzort, hogy lássuk, ahogy kergeti a macska. Most függesszük fel a processzt. Hogyan kell ezt megtenni? Mi a kimenet?

     A `Ctrl` + `z` billentyűkombináció lenyomásával:

     ```
     [1]+  Stopped                 oneko
     ```

   - Nézzük meg, hogy jobunk van. Mit kell beírnunk? Mi a kimenet?

     ```
     $ jobs
     [1]+  Stopped                 oneko
     ```

   - Most pedig küldjük a háttérbe a job azonosítójának megadásával. Mi a kimenet? Honnan tudhatjuk, hogy a job a háttérben fut?

     ```
     $ bg %1
     [1]+ oneko &
     ```

     A macska ismét mozog.

   - Végül terminálju a jobot az ID megadásával. Mit kell beírnunk?

     ```
     $ kill %1
     ```

2. Fedezzük fel az *Apache HTTPD web server* (`apache2`) által indított összes folyamat PID-jét két különböző paranccsal:

   ```
   $ pgrep apache2
   ```

vagy

```
$ pidof apache2
```

3. Fejezzük be az összes `apache2` folyamatot a PID-jük használata nélkül, két különböző paranccsal:

```
$ pkill apache2
```

vagy

```
$ killall apache2
```

4. Tegyük fel, hogy az `apache2` összes példányát meg kell szüntetnünk, és nincs időnk kideríteni, hogy mi a PID-jük. Hogyan tudnánk ezt elérni a `kill` használatával az alapértelmezett `SIGTERM` jellel egy sorban:

```
$ kill $(pgrep apache2)
$ kill `pgrep apache2`
```

vagy

```
$ kill $(pidof apache2)
$ kill `pidof apache2`
```

> **NOTE** Mivel a `SIGTERM` (`15`) az alapértelmezett jel, nem szükséges semmilyen opciót átadni a `kill`-nek.

5. Indítsuk el a `top`-ot és tegyük meg a következőket:
   - Nézzük meg a processzek forest (erdő) nézetét:

     Nyomjuk meg a `V`-t.

   - A folyamatok teljes útvonalának megjelenítése, megkülönböztetve a felhasználói tér és a kerneltér között:

     Nyomjuk meg a `c`-t.

6. A `ps` használatával jelenítsük meg az *Apache HTTPD web server* felhasználó által indított összes folyamatot (`www-data`):
   - BSD szintaxis:

     ```
     $ ps U www-data
     ```

   - UNIX szintaxis:

     ```
     $ ps -u www-data
     ```

   - GNU szintaxis:

     ```
     $ ps --user www-data
     ```

Válaszok a gondolkodtató feladatokra

1. A `SIGHUP` jel bizonyos daemonok újraindítására használható. Az *Apache HTTPD web server* esetében például a `SIGHUP` elküldése a szülőfolyamatnak (amelyet az `init` indít el) megöli a gyermekfolyamatokat. A szülő azonban újraolvassa a konfigurációs fájljait, újra megnyitja a naplófájlokat, és új gyermekeket indít. Végezzük el a következő feladatokat:
   - Indítsuk el a webszervert:

     ```
     $ sudo systemctl start apache2
     ```

   - Tudjuk meg a szülőprocessz PID-jét:

     ```
     $ ps aux | grep apache2
     ```

     A szülői folyamat a `root` felhasználó által indított folyamat. A mi esetünkben az, amelynek PID azonosítója `1653`.

   - Az Apache HTTPD webkiszolgáló újraindítása a `SIGHUP` jel elküldésével a szülőfolyamatnak:

     ```
     $ kill -SIGHUP 1653
     ```

   - Ellenőrizzük, hogy a szülő nem terminálódott-e, és születtek-e új gyermekek:

     ```
     $ ps aux | grep apache2
     ```

     Most a szülő `apache2` folyamatot kell látnunk két új gyermekével együtt.

2. Bár kezdetben statikus, a `ps` kimenete a `ps` és a `watch` kombinálásával dinamikussá *fordítható*. Monitorozzuk az *Apache HTTPD web server* új kapcsolatait! Az alább leírt feladatok elvégzése előtt ajánlott elolvasni a `MaxConnectionsPerChild` direktíva leírását a Apache MPM Common Directives oldalon.
   - Adjuk hozzá a `MaxConnectionsPerChild` direktívát `1` értékkel az `apache2` konfigurációs fájljához — *Debian* és Debian-alapú rendszerek esetén ez az `/etc/apache2/apache2.conf` fájl; *CentOS* család esetén az `/etc/httpd/conf/httpd.conf`. Ne felejtsük el újraindítani az `apache2`-t, hogy a változtatások életbe lépjenek!

     A konfigurációs fájlhoz a `MaxConnectionsPerChild 1` sort kell hozzáadnunk. A webszerver újraindításának egy módja: `sudo systemctl restart apache2`.

- Írjunk be egy parancsot, ami használja a `watch`, `ps` és `grep` parancsokat `apache2` kapcsolatok esetén!

  ```
  $ watch 'ps aux | grep apache2'
  ```

  vagy

  ```
  $ watch "ps aux | grep apache2"
  ```

- Nyissuk meg a böngészőt vagy használjunk egy parancssori böngészőt, mint például a `lynx`, `hogy létrehozzuk a kapcsolatot a webszerverrel az IP-címén keresztül! Mit figyelhetünk meg a` `watch` kimenetén?

  A `www-data` által tulajdonolt gyermek-processzek egyike eltűnik.

3. Amint azt már megtanultuk, a `top` alapértelmezés szerint a CPU-használat százalékos aránya szerint rendezi a feladatokat csökkenő sorrendben (a magasabb értékek vannak felül). Ez a viselkedés módosítható az `M` (memóriahasználat), `N` (folyamat egyedi azonosítója), `T` (futási idő) és `P` (CPU-idő százalékos aránya) interaktív kulcsokkal. A feladatlistát azonban a `top -o` kapcsolóval történő elindításával is tetszés szerint rendezhetjük (további információkért nézzük meg a top `man` oldalát). Most hajtsuk végre a következő feladatokat:

   - Indítsuk el a `top`-ot, hogy a feladatok memóriahasználat szerint legyenek rendezve:

     ```
     $ top -o %MEM
     ```

   - Ellenőrizzük a memória oszlop kiemelésével, hogy a megfelelő parancsot írtuk-e be:

     Nyomjuk meg az `x`-et.

4. A `ps` szintén rendelkezik egy `o` kapcsolóval, amellyel megadhatjuk a megjeleníteni kívánt oszlopokat. Vizsgáljuk meg ezt a lehetőséget, és végezzük el a következő feladatokat:

   - Indítsuk el a `ps`-t úgy, hogy csak a *user*, a *percentage of memory used*, a *percentage of CPU time used* és a *full command* információk jelenjenek meg:

     ```
     $ ps o user,%mem,%cpu,cmd
     ```

   - Most indítsuk el a `ps`-t úgy, hogy csak a felhasználó adatai és az általa használt programok neve jelenjen meg:

```
$ ps o user,comm
```

Linux Professional Institute

103.5 Lecke 2

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	103 GNU és Unix parancsok
Fejezet:	103.5 Folyamatok létrehozása, monitorozása és megszakítása
Lecke:	2/2

Bevezetés

Az előző leckében látott eszközök és segédprogramok nagyon hasznosak a folyamatok általános monitorozásához. Egy rendszergazdának azonban szüksége lehet arra, hogy egy lépéssel tovább menjen. Ebben a leckében a terminálmultiplexerek fogalmát tárgyaljuk, és megismerkedünk a *GNU Screen* és a *tmux* programokkal, mivel — a mai modern és nagyszerű terminálemulátorok ellenére — a multiplexerek még mindig őriznek néhány nagyszerű, hatékony funkciót egy produktív rendszergazda számára.

A terminál multiplexerek funkciói

Az elektronikában a multiplexer (vagy *mux*) olyan eszköz, amely lehetővé teszi, hogy több bemenetet egyetlen kimenethez csatlakoztassanak. Így egy terminálmultiplexer lehetővé teszi számunkra, hogy szükség szerint váltsunk a különböző bemenetek között. Bár nem teljesen azonosak, a `screen` és a `tmux` egy sor közös tulajdonsággal rendelkeznek:

- Minden sikeres invokálás legalább egy munkamenetet eredményez, amely — viszont — legalább egy ablakot tartalmaz. Az ablakok programokat tartalmaznak.

- Az ablokok régiókra vagy ablaktáblákra/panelekre oszthatók - ami segítheti a produktivitást, ha egyszerre több programmal dolgozunk.
- Könnyű kezelhetőség: a legtöbb parancs futtatásához egy billentyűkombinációt — az úgynevezett *command prefix* vagy *command key* — használnak, amelyet egy másik karakter követ.
- A munkamenetek leválaszthatók az aktuális terminálról (azaz a programok a háttérbe kerülnek és tovább futnak). Ez garantálja a programok teljes körű végrehajtását, függetlenül attól, hogy véletlenül bezárjuk a terminált, időnként terminálfagyást vagy akár távoli kapcsolatvesztést tapasztalunk.
- Socket kapcsolat.
- Másolás mód.
- Nagymértékben testreszabhatók.

GNU Screen

A Unix korábbi korszakában (1970-80-as évek) a számítógépek alapvetően egy központi számítógéphez csatlakoztatott terminálokból álltak. Ennyi volt minden, nem volt több ablak vagy lap. És ez volt az oka a GNU Screen 1987-es létrehozásának: több független *VT100 képernyő* emulálása egyetlen fizikai terminálon.

Ablakok

A GNU Screen a `screen` terminálba történő beírásával hívható elő. Először egy üdvözlő üzenetet fogunk látni:

```
GNU Screen version 4.05.00 (GNU) 10-Dec-16

Copyright (c) 2010 Juergen Weigert, Sadrul Habib Chowdhury
Copyright (c) 2008, 2009 Juergen Weigert, Michael Schroeder, Micah Cowan, Sadrul Habib
Chowdhury
Copyright (c) 1993-2002, 2003, 2005, 2006, 2007 Juergen Weigert, Michael Schroeder
Copyright (c) 1987 Oliver Laumann
(...)
```

Nyomjuk meg a Space vagy az Enter billentyűt az üzenet bezárásához, és megjelenik a parancssor:

```
$
```

Úgy tűnhet, hogy semmi sem történt, de a valóság az, hogy a `screen` már létrehozta és kezeli az első munkamenetet és ablakot. A képernyő parancsának előtagja Ctrl + a. Ha a terminál kijelzőjének alján lévő összes ablakot szeretnénk látni, írjuk be a Ctrl + a - w kombinációt:

```
0*$ bash
```

Itt van, az eddigi egyetlen ablakunk! Figyeljük meg, hogy a számlálás 0-nál kezdődik. Egy másik ablak létrehozásához nyomjuk le a Ctrl + a - c billentyűket. Egy új promptot fogunk látni. Indítsuk el a `ps`-t ebben az új ablakban:

```
$ ps
  PID TTY          TIME CMD
  974 pts/2    00:00:00 bash
  981 pts/2    00:00:00 ps
```

és írjuk be a Ctrl + a - w újra:

```
0-$ bash  1*$ bash
```

Itt van a két ablakunk (figyeljük meg a csillagot, amely azt jelzi, hogy éppen melyik van megjelenítve). Mivel azonban mindkettő a Bash segítségével indult, mindkettőnek ugyanaz a neve. Mivel a `ps`-t hívtuk meg a jelenlegi ablakunkban, nevezzük át ugyanezzel a névvel! Ehhez be kell írnunk Ctrl + a - A és írjuk be az új ablak nevét (`ps`):

```
Írjuk át az ablak címét erre: ps
```

Most hozzunk létre egy másik ablakot, de már az elején adjunk neki nevet: `yetanotherwindow`. Ezt úgy tehetjük meg, hogy a screent a `-t` kapcsolóval hívjuk meg:

```
$ screen -t yetanotherwindow
```

Különböző módon válthatunk az ablakok között:

- A Ctrl + a - n használatával (váltás a *következő* ablakra) és Ctrl + a - p (váltás az *előző* ablakra).
- A Ctrl + a - number használatával (váltás a *number* számú ablakra).
- A Ctrl + a - " használatával az összes ablak listájának megtekintéséhez. A nyílbillentyűkkel mozoghatunk fel és le, majd az Enter billentyűvel kiválaszthatjuk a kívánt ablakot:

```
Num Name                                                Flags

   0 bash                                                  $
   1 ps                                                    $
   2 yetanotherwindow
```

Az ablakokkal kapcsolatban fontos megjegyezni a következőket:

- Az ablakok egymástól függetlenül futtatják a programjaikat.
- A programok akkor is tovább futnak, ha az ablakuk nem látható (akkor is, ha a képernyő munkamenet leválasztásra kerül, amint azt hamarosan látni fogjuk).

Egy ablak eltávolításához egyszerűen fejezzük be a benne futó programot (az utolsó ablak eltávolítása után a `screen` maga is befejeződik). Alternatív megoldásként használjuk a `Ctrl` + `a` - `k` parancsot, miközben az eltávolítani kívánt ablakban vagyunk; ekkor a rendszer megerősítést fog kérni:

```
Really kill this window [y/n]

Window 0 (bash) killed.
```

Régiók

A `screen` a terminál képernyőjét több régióra oszthatja, amelyekben ablakokat helyezhet el. Ezek a felosztások lehetnek vízszintesek (`Ctrl` + `a` - `S`) vagy függőlegesek (`Ctrl` + `a` - `|`).

Az egyetlen dolog, amit az új régió mutatni fog, az a `--` az alján, ami azt jelenti, hogy üres:

```
   1 ps                                                    --
```

Az új régióra való áttéréshez nyomjuk le a `Ctrl` + `a` - `Tab` billentyűket. Most a már látott módszerek bármelyikével hozzáadhatunk egy ablakot, például: `Ctrl` + `a` - `2`. Most a `--`-nak `2 yetanotherwindow` -ra kell változnia:

```
$ ps                                                      $
  PID TTY          TIME CMD
 1020 pts/2    00:00:00 bash
 1033 pts/2    00:00:00 ps
$ screen -t yetanotherwindow
```

```
                                                                                
  1 ps                                                    2 yetanotherwindow
```

A régiókkal való munka során a következő fontos szempontokat kell szem előtt tartani:

- Régiók között így lehet váltani: `Ctrl` + `a` - `Tab`.
- Az aktuális kivételével minden régiót így lehet terminálni: `Ctrl` + `a` - `Q`.
- Az aktuális régiót így lehet terminálni: `Ctrl` + `a` - `X`.
- Egy régió megszüntetése nem szünteti meg a hozzá tartozó ablakot..

Munkamenetek

Eddig néhány ablakkal és régióval játszottunk, de mindegyik ugyanahhoz az egyetlen munkamenethez (session) tartozott. Itt az ideje, hogy elkezdjünk a munkamenetekkel is játszani. Az összes munkamenet listájának megtekintéséhez írjuk be a `screen -list` vagy `screen -ls` parancsot:

```
$ screen -list
There is a screen on:
        1037.pts-0.debian       (08/24/19 13:53:35)     (Attached)
1 Socket in /run/screen/S-carol.
```

Eddig ez az egyetlen munkamenetünk:

PID

`1037`

Név

`pts-0.debian` (a terminál — esetünkben egy *pseudo terminal slave* — és hosztnév).

Státusz

`Csatolt` (Attached)

Hozzunk létre egy új munkamenetet, és adjunk neki egy beszédesebb nevet:

```
$ screen -S "second session"
```

A terminál tartalma törlődik, és egy új promptot kapunk. Ismét ellenőrizhetjük a munkameneteket:

```
$ screen -ls
There are screens on:
        1090.second session     (08/24/19 14:38:35)     (Attached)
        1037.pts-0.debian       (08/24/19 13:53:36)     (Attached)
2 Sockets in /run/screen/S-carol.
```

A munkamenet megállításához lépjünk ki az összes ablakából, vagy írjuk be a `screen -S SESSION-PID -X quit` parancsot (megadhatjuk a munkamenet nevét is). Szabaduljunk meg az első munkamenetünktől:

```
$ screen -S 1037 -X quit
```

Visszakerülünk a terminál prompthoz a `screen`-en kívülre. De ne feledjük, a második munkamenetünk még mindig él:

```
$ screen -ls
There is a screen on:
    1090.second session (08/24/19 14:38:35) (Detached)
1 Socket in /run/screen/S-carol.
```

Mivel azonban terminátuk a szülő munkamenetet, új címkét kap: `Detached` (leválasztott).

Munkamenet leválasztása

Több okból is előfordulhat, hogy egy screen munkamenetet le akarunk választani a termináljáról:

- Hagyni, hogy a számítógép elvégezze a dolgát és később otthonról csatlakozni rá.
- Munkamenet megosztása más felhasználókkal.

A munkamenet leválasztása a `Ctrl` + `a`-`d` billentyűkombinációval lehetséges, ezzel visszakerülünk a terminálunkba:

```
[detached from 1090.second session]
$
```

A munkamenethez való újbóli kapcsolódáshoz használjuk a `screen -r SESSION-PID` parancsot.

Alternatívaként használhatjuk a `SESSION-NAME` parancsot is, ahogy fentebb láttuk. Ha csak egy leválasztott munkamenet van, egyik sem kötelező:

```
$ screen -r
```

Ez a parancs elegendő ahhoz, hogy újra csatlakozzunk a második munkamenetünkhöz:

```
$ screen -ls
There is a screen on:
        1090.second session      (08/24/19 14:38:35)     (Attached)
1 Socket in /run/screen/S-carol.
```

Fontos kapcsolók a munkamenet újracsatolásához:

`-d -r`
: Újracsatol egy munkamenetet és — ha szükséges — először leválasztja.

`-d -R`
: Ugyanaz, mint a `-d -r`, de a `screen` akkor is létre fogja hozni a munkamenetet, ha az nem létezik.

`-d -RR`
: Ugyanaz, mint a `-d -R`, azonban az első munkamenetet használja, ha egynél több elérhető.

`-D -r`
: Munkamenet újracsatolása. Ha szükséges, először távolról válasszuk le és jelentkezzünk ki a munkamenetből.

`-D -R`
: Ha egy munkamenet fut, akkor újracsatolja (szükség esetén előbb távolról leválasztva és kijelentkezve). Ha nem futott, létrehozza, és értesíti a felhasználót.

`-D -RR`
: Ugyanaz, mint a `-D -R` — csak erősebb.

`-d -m`
: Elindítja a `screen`-t *leválasztott módban* (detached mode). Ez új munkamenetet hoz létre, de nem kapcsolódik hozzá. Ez hasznos lehet a rendszerindító szkripteknél.

`-D -m`

Ugyanaz, mint a `-d -m`, de nem forkol új processzt. A parancs kilép, ha a munkamenet terminálódik.

Olvassuk el a `screen` manualját, hogy megismerjük a többi kapcsolót.

Másolás és beillesztés: Scrollback mód

A GNU Screen rendelkezik másolási (copy) vagy *scrollback móddal*. Ha beléptünk, a kurzort az aktuális ablakban és annak előzményeiben a nyílbillentyűkkel mozgathatjuk. Szöveget jelölhetünk ki és másolhatunk az ablakok között. A követendő lépések a következők:

1. Belépés a copy/scrollback módba: `Ctrl` + `a`-`[`.
2. Menjünk a másolni kívánt szövegrész elejére a nyílbillentyűkkel.
3. Jelöljük ki a másolni kívánt szövegrész elejét: szóköz.
4. Menjünk a másolni kívánt szövegrész végére a nyílbillentyűkkel.
5. Jelöljük meg a másolni kívánt szövegrész végét: szóköz.
6. Menjünk a kívánt ablakba és illesszük be a szövegrészt: `Ctrl` + `a`-`]`.

A képernyő testreszabása

A screen rendszerszintű konfigurációs fájlja az `/etc/screenrc`. Alternatívaként a felhasználói szintű `~/.screenrc` is használható. A fájl négy fő konfigurációs részt tartalmaz:

`SCREEN SETTINGS`

Általános beállításokat úgy határozhatunk meg, hogy megadjuk a *direktívát*, amelyet egy szóköz és az *érték* követ, mint az alábbi: `defscrollback 1024`.

`SCREEN KEYBINDINGS`

Ez a szakasz elég érdekes, mivel lehetővé teszi, hogy átdefiniálja azokat a billentyűkódokat, amelyek esetleg zavarják a terminál mindennapi használatát. Használjuk a `bind` kulcsszót, amelyet egy szóköz, a parancs előtagja után használandó karakter, egy újabb szóköz és a parancs követ, mint az alábbi példában: `bind l kill` (ez a beállítás az ablakok megállításának alapértelmezett módját `Ctrl` + `a`-`l`-re változtatja).

A képernyő összes hozzárendelésének megjelenítéséhez használjuk a `Ctrl` + `a`-`?` billentyűkombinációt vagy nézzük meg a man oldalát.

TIP Természetesen magát a parancs előtagját is megváltoztathatjuk. Ha például erről

`Ctrl` + `a` erre `Ctrl` + `b` akarunk váltani, adjuk hozzá ezt a sort: `escape ^Bb`.

`TERMINAL SETTINGS`

Ez a szakasz többek között a terminálablak méretével és a pufferekkel kapcsolatos beállításokat tartalmazza. A nem blokkoló üzemmód engedélyezéséhez, hogy jobban megbirkózzunk például az instabil ssh-kapcsolatokkal, a következő konfigurációt használjuk: `defnonblock 5`.

`STARTUP SCREENS`

Beépíthetünk parancsokat, hogy különböző programok fussanak a `screen` indításakor; például: `screen -t top top` (a screen megnyit egy `top` nevű ablakot, benne a `top` programmal).

tmux

A `tmux` 2007-ben jelent meg. Bár nagyon hasonlít a `screen`-re, van néhány jelentős különbség:

- Kliens-szerver modell: a kiszolgáló számos munkamenetet biztosít, amelyek mindegyikéhez több ablak is kapcsolódhat, amelyeket viszont különböző kliensek használhatnak.
- A munkamenetek, ablakok és kliensek interaktív kiválasztása menükön keresztül.
- Ugyanaz az ablak több munkamenethez is kapcsolódhat.
- Elérhető mind *vim* és *Emacs* billentyűkiosztás.
- UTF-8 és 256-színű terminál támogatás.

Ablakok

A `tmux` egyszerűen a parancssorba beírva hívható elő. Ekkor megjelenik egy shell prompt és egy állapotsor az ablak alján:

```
[0] 0:bash*                                                    "debian" 18:53 27-Aug-19
```

A `hosztnév`, az idő és a dátum mellett az állapotsor a következő információkat tartalmazza:

Munkamenet neve

`[0]`

Ablak száma

`0:`

Ablak neve

`bash*`. Alapértelmezés szerint ez az ablakban futó program neve, és — a `screen`-nel ellentétben — a `tmux` automatikusan frissíti, hogy tükrözze az aktuálisan futó programot. Figyeljük meg a csillagot, ami azt jelzi, hogy ez az aktuális, látható ablak.

A munkamenet- és ablakneveket a `tmux` meghívásakor adhatjuk meg:

```
$ tmux new -s "LPI" -n "Window zero"
```

Az állapotsor ennek megfelelően változik:

```
[LPI] 0:Window zero*                                          "debian" 19:01 27-Aug-
19
```

A tmux parancs prefixe `Ctrl` + `b`. Egy új ablak létrehozásához csak erre van szükség: `Ctrl` + `b`-`c`; ez egy új prompthoz vezet és az állapotsor is jelzi az új ablakot:

```
[LPI] 0:Window zero- 1:bash*                                  "debian" 19:02 27-Aug-
19
```

Mivel a Bash az alapértelmezett shell, az új ablak alapértelmezés szerint ezt a nevet kapja. Indítsuk el a `top` ablakot, és nézzük meg, hogy a név `top`-ra változik:

```
[LPI] 0:Window zero- 1:top*                                   "debian" 19:03 27-Aug-19
```

Mindenesetre az ablakot átnevezhetjük a `Ctrl` + `b`-`,` billentyűkombinációval. Amikor a rendszer kéri, adjuk meg az új nevet, és nyomjuk meg az Entert:

```
(rename-window) Window one
```

A `Ctrl` + `b`-`w` billentyűkombinációval az összes ablakot megjeleníthetjük kiválasztásra (a nyílbillentyűkkel felfelé és lefelé mozoghatunk, a `enter` billentyűvel pedig választhatunk):

```
(0)  0: Window zero- "debian"
(1)  1: Window one* "debian"
```

A `screen`-hez hasonlóan az egyik ablakból a másikba ugorhatunk a következővel:

`Ctrl` + `b` - `n`
következő ablak.

`Ctrl` + `b` - `p`
előző ablak.

`Ctrl` + `b` - `number`
number számú ablak.

Az ablakok eltávolításához használjuk a `Ctrl` + `b` - `&` parancsot, ez megerősítést fog kérni:

```
kill-window Window one? (y/n)
```

További érdekes parancsok az ablakokhoz:

`Ctrl` + `b` - `f`
ablak keresése név alapján.

`Ctrl` + `b` - `.`
ablak indexének megváltoztatása.

A parancsok teljes listáját a man oldalon találhatjuk meg.

Panelek

A `screen` ablakfelosztási lehetősége a `tmux`-ban is jelen van. Az így kapott felosztásokat azonban nem *régióknak*, hanem *paneleknek* (pane) nevezzük. A legfontosabb különbség a régiók és a panelek között az, hogy az utóbbiak egy ablakhoz kapcsolódó teljes pszeudo-terminálok. Ez azt jelenti, hogy egy panel terminálása a hozzá tartozó pszeudo-terminált és a benne futó kapcsolódó programokat is terminálja.

Egy ablak vízszintes felosztása így lehetséges:[Ctrl+b]-`"`:

```
Tasks:  93 total,   1 running,  92 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  4050960 total,  3730920 free,   114880 used,   205160 buff/cache
KiB Swap:  4192252 total,  4192252 free,        0 used.  3716004 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 1340 carol     20   0   44876   3400   2800 R  0.3  0.1   0:00.24 top
    1 root      20   0  139088   6988   5264 S  0.0  0.2   0:00.50 systemd
```

```
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.04 ksoftirqd/0
    4 root      20   0       0      0      0 S  0.0  0.0   0:01.62 kworker/0:0
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S  0.0  0.0   0:00.06 rcu_sched
    8 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 migration/0
   10 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 lru-add-drain
   11 root      rt   0       0      0      0 S  0.0  0.0   0:00.01 watchdog/0
   12 root      20   0       0      0      0 S  0.0  0.0   0:00.00 cpuhp/0
$
─────────────────────────────────────────────────────────────────────────────────
───
$




[LPI] 0:Window zero- 1:Window one*                                  "debian" 19:05 27-
Aug-19
```

A függőleges felosztás pedig így: `Ctrl` + `b`-`%`:

```
                                                                              |$
    1 root      20   0  139088   6988   5264 S  0.0  0.2   0:00.50 systemd     |
                                                                              |
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd    |
                                                                              |
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.04 ksoftirqd/0 |
    4 root      20   0       0      0      0 S  0.0  0.0   0:01.62 kworker/0:0 |
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H|
    7 root      20   0       0      0      0 S  0.0  0.0   0:00.06 rcu_sched   |
    8 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_bh      |
    9 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 migration/0 |
                                                                              |
```

```
   10 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 lru-add-drai
n                                                                              
   11 root      rt   0       0      0      0 S  0.0  0.0   0:00.01 watchdog/0  
                                                                               
   12 root      20   0       0      0      0 S  0.0  0.0   0:00.00 cpuhp/0     
                                                                               
$                                                                              
_______________________________________________________________________________
___
$


[LPI] 0:Window zero- 1:Window one*                                 "debian" 19:05 27-
Aug-19
```

Az aktuális panel (a benne futó pszeudo-terminállal és a hozzá tartozó programokkal együtt) terminálásához használjuk a `Ctrl` + `b` - `x` parancsot. Az állapotsor megerősítést fog kérni:

```
kill-pane 1? (y/n)
```

Fontos panelekhez kötődő parancsok:

`Ctrl` + **`b`** - **`↑`**,**`↓`**,**`←`**,**`→`**
: mozgás a panelek között.

`Ctrl` + **`b`** - **`;`**
: az utolsó aktív panelre váltás.

`Ctrl` + `b` - `Ctrl` + `arrow key`
: a panel méretének csökkentése egy sorral.

`Ctrl` + `b` - `Alt` + `arrow key`
: a panel átméretezése öt sorral.

`Ctrl` + `b` - `{`
: panelek lapozása (aktuálisról az előzőre).

`Ctrl` + `b` - `}`
: panelek lapozása (aktuálisról a következőre).

`Ctrl` + `b` - `z`
: panelba/ból történő nagyítás.

`Ctrl` + `b` - `t`
: a `tmux` megjelenít egy dizájnos órát a panelon belül (eliminálni a `q` lenyomásával lehet).

`Ctrl` + `b` - `!`
: panel ablakká alakítása.

A parancsok teljes listája a man oldalon található.

Munkamenetek

A munkameneteket `tmux`-ban így lehet listázni: `Ctrl` + `b` - `s`:

```
(0) + LPI: 2 windows (attached)
```

Alternatívaként használhatjuk a `tmux ls` parancsot is:

```
$ tmux ls
LPI: 2 windows (created Tue Aug 27 19:01:49 2019) [158x39] (attached)
```

Csak egy munkamenet van (`LPI`), amely két ablakot tartalmaz. Hozzunk létre egy új munkamenetet a jelenlegi munkamenetünkből. Ezt a `Ctrl` + `b` billentyűkombinációval tehetjük meg, a promptba írjuk be a `:new` parancsot, majd nyomjuk le az Entert. Az új munkamenetbe kerülünk, amint az az állapotsoron is látható:

```
[2] 0:bash*                                                    "debian" 19:15 27-Aug-19
```

Alapértelmezés szerint az `tmux` a munkamenetet `2`-nek nevezte el. Átnevezéséhez használjuk a `Ctrl` + `b`-`$` parancsot. Amikor a rendszer kéri, adjuk meg az új nevet, és nyomjuk meg az Entert:

```
(rename-session) Second Session
```

A munkamenetek között a `Ctrl` + `b`-`s` segítségével válthatunk (használjuk a nyílbillentyűket és az `enter` billentyűt):

```
(0) + LPI: 2 windows
(1) + Second Session: 1 windows (attached)
```

Egy munkamenet megszüntetéséhez használhatja az `tmux kill-session -t SESSION-NAME` parancsot. Ha a parancsot az aktuális, csatlakoztatott munkamenetből írjuk be, akkor a `tmux`-ból kilépünk és visszatérünk az eredeti terminál munkamenetéhez:

```
$ tmux kill-session -t "Second Session"
[exited]
$
```

Munkamenetek leválasztása

A `Second Session` terminálásával kikerültünk a `tmux'-on kívülre. Azonban még mindig van egy aktív munkamenetünk. Kérjük le a `tmux` munkameneteinek listáját, és meg is fogjuk találni:

```
$ tmux ls
LPI: 2 windows (created Tue Aug 27 19:01:49 2019) [158x39]
```

Ez a munkamenet azonban lecsatolódott a termináljáról. A `tmux attach -t SESSION-NAME` paranccsal csatolhatjuk (az `attach` helyettesíthető `at`-val vagy — egyszerűen — `a`-val). Ha csak egyetlen munkamenet van, a név megadása opcionális:

```
$ tmux a
```

Most újra a munkamenetben vagyunk; a munkamenetből való kilépéshez nyomjuk meg a `Ctrl` + `b`-`d`

billentyűkombinációt:

```
[detached (from session LPI)]
$
```

TIP Ugyanaz a munkamenet több terminálhoz is csatlakoztatható. Ha egy munkamenetet úgy akarunk csatolni, hogy előbb minden más terminálról leválasztjuk, használjuk a `-d` kapcsolót: `tmux attach -d -t SESSION-NAME`.

Fontos paracsok a munkamenetek csatolásához/leválasztásához:

`Ctrl` + `b` - `D`
: a leválasztandó kliens kijelölése.

`Ctrl` + `b` - `r`
: a kliens terminálján frissítése.

A parancsok teljes listáját a man oldalon találjuk.

Másolás és beillesztés: Scrollback mód

A `tmux` szintén rendelkezik másolási móddal, alapvetően ugyanúgy, mint a `screen` (ne felejtsük el, hogy a tmux és nem a screen parancs prefixét kell használni!). Az egyetlen különbség a parancsok között, hogy a `Ctrl` + Space billentyűkombinációval jelöljük a kijelölés kezdetét, és a `Alt` + `w` billentyűkombinációval másoljuk a kijelölt szöveget.

A tmux testreszabása

A `tmux` konfigurációs fájljai általában a `/etc/tmux.conf` és a `~/.tmux.conf`. Indításkor a `tmux` ezeket a fájlokat forrásként kezeli, ha léteznek. Lehetőség van arra is, hogy a `tmux`-ot az `-f` kapcsolóval indítsuk el, hogy egy másik konfigurációs fájlt adjunk meg. Egy példa a `tmux` konfigurációs fájlra a `/usr/share/doc/tmux/example_tmux.conf`. A testreszabás lehetőségei meglehetősen magas szinten vannak. Íme, néhány példa:

- A prefix megváltoztatása

```
# Change the prefix key to C-a
set -g prefix C-a
unbind C-b
bind C-a send-prefix
```

- Extra billentyű-hozzárendelések beállítása 9-esnél magasabb számú ablakokhoz

```
# Some extra key bindings to select higher numbered windows
bind F1 selectw -t:10
bind F2 selectw -t:11
bind F3 selectw -t:12
```

Az összes hozzárendelés átfogó listáját a `Ctrl` + `b` - `?` (nyomjuk meg a `q`-t a kilépéshez) billentyűparanccsal érhetjük el vagy a man oldalon találhatjuk meg.

Gyakorló feladatok

1. Jelöljük meg, hogy a következő utasítások/funkciók a GNU Screenhez, a tmuxhoz vagy mindkettőhöz tartoznak-e:

Funkció/utasítás	**GNU Screen**	**tmux**
Az alapértelmezett prefix: `Ctrl` + `a`		
Kliens-szerver modell		
A panelek pszeudo-terminálok		
Egy régió terminálása nem terminálja a hozzátartozó ablako(ka)t		
A munkamenetekbe beletartoznak az ablakok		
A munkamenetek leválaszthatók		

2. Telepítsük a GNU Screen-t (csomag neve: `screen`) és végezzük el a következő feladatokat:
   - Indítsuk el a programot! Milyen parancsot használunk?
   - Indítsuk el a `top`-ot:
   - A prefix segítségével nyissunk meg egy új ablakot, majd nyissuk meg a `/etc/screenrc`-t a `vi` használatával:
   - Listázzuk ki az ablakokat a képernyő alján:
   - Változtassuk meg az aktuális ablak nevét `vi`-ra:
   - A fennmaradó ablak nevét változtassuk meg `top`-ra. Ehhez először jelenítsük meg az összes ablak listáját, hogy fel-le mozoghassunk és kiválaszthassuk a megfelelőt:

- Ellenőrizzük, hogy a nevek megváltoztak-e, úgy, hogy az ablaknevek ismét megjelennek a képernyő alján:

- Válasszuk le a munkamenetet és a `screen` hozzon létre egy újat `ssh` néven:

- Válasszuk le az `ssh`-t is és listázzuk ki a munkamenetet a `screen` segítségével:

- Csatlakozzunk az első munkamenethez a PID-jével:

- Térjünk vissza a `top` ablakhoz. Osszuk fel az ablakot vízszintesen, és lépjünk az új üres régióba:

- Listáztassuk ki az összes ablakot a `screen` segítségével és válasszuk ki a `vi`-t, hogy jelenjen meg az üres régióban:

- Osszuk fel függőlegesen a jelenlegi régiót, lépjünk át az újonnan létrehozott üres régióba és társítsuk azt egy vadonatúj ablakhoz:

- Terminálju az összes régiót, kivéve az aktuálisat (ne feledjük, attól, hogy a régiókat eltüntetjük, az ablakok megmaradnak)! Ezután lépjünk ki az aktuális munkamenet összes ablakából, amíg maga a munkamenet nem terminálódik:

- Végül, még egyszer listáztassuk ki a `screen` segítségével az összes munkamenet, terminálju a megmaradt `ssh` munkamenetat PID-del és ellenőrizzük, hogy nem maradt több munkamanet:

3. Telepítsük a `tmux`-ot (csomagnév: `tmux`) és végezzük el az alábbi feladatokat:
   - Indítsuk el a programot! Milyen parancsot kell használnunk?
   - Indítsuk el a `top`-ot (vegyük észre, hogy — néhány másodpercen belül — az ablak neve megváltozik `top`-ra az állapotsoron):
   - A tmux prefixét használva nyissunk meg egy új ablakot, majd hozzuk létre a `~/.tmux.conf` fájlt a `nano` segítségével:
   - Osszuk fel az ablakot függőlegesen, és csökkentsük az újonnan létrehozott panel méretét néhányszor:
   - Változtassuk meg az aktuális ablak nevét `text editing`-re; majd listáztassuk ki a `tmux` összes munkamenetét:
   - Ugorjunk át a `top` ablakra, majd vissza az aktuálisra, ugyanazon billentyűkombináció használatával:
   - Csatlakozzunk le az aktuális munkamenetről és hozzunk létre egy újat, aminek a neve `ssh`, az ablakának a neve pedig `ssh window`:
   - Csatlakozzunk le az `ssh` munkamenetről és a `tmux` jelenítse meg újra az összes munkamenetét:

NOTE

Ettől a ponttól kezdve a gyakorlat megköveteli, hogy egy *remote* (távoli) gépet használjunk a helyi géphez történő `ssh` kapcsolódásokhoz (egy virtuális gép is tökéletesen megfeel és nagyon praktikus is lehet). Győződjünk meg róla, hogy a helyi gépen telepítve van és fut az `openssh-server`, valamint arról, hogy a távoli gépen legalább az `openssh-client` telepítve van.

- Most indítsuk el a távoli gépet és csatlakozzunk hozzá `ssh`-val a helyi gépről. Ha a kapcsolat létrejött, ellenőrizzük a `tmux` munkameneteket:

- A távoli gépen csatlakozzunk az `ssh` munkamenethez a nevével:

- A saját gépünkön csatlakozzunk az `ssh` munkamenethez a nevével úgy, hogy terminálódjon először a távoli kapcsolat:

- Jelenítsük meg az összes munkamenetet és lépjünk az első munkamenetre (`[0]`). Ha már ott vagyunk, terminálják az `ssh` munkamenetet a nevével:

- Végül csatlakozzunk le az aktuális munkamenetről és terminálják a nevével:

Gondolkodtató feladatok

1. A `screen` és a `tmux` is beléphet parancssori módba a _command prefix + `:` segítségével (a `tmux` esetében már láttunk egy rövid példát). Végezzünk némi kutatást és hajtsuk végre a következő feladatokat parancssori üzemmódban:
   - A `screen` lépjen be a copy módba:
   - A `tmux` nevezze át az aktuális ablakot:
   - A `screen` zárja be az összes ablakot és terminálja a munkamenetet:
   - A `tmux` válassza ketté a panelt:
   - A `tmux` zárja be az aktuális ablakot:
2. Amikor a `screen` másolási módba lép, nem csak a nyílbillentyűkkel és a `PgUP` vagy `PgDown` billentyűkkel navigálhatunk az aktuális ablakban és a scrollback pufferben. Lehetőség van egy vi-szerű teljes képernyős szerkesztő használatára is. Ezzel a szerkesztővel a következő feladatokat végezhetjük el:
   - Echo `supercalifragilisticexpialidocious` a `screen` terminálban:
   - Most másoljuk az öt egymást követő karaktert (balról jobbra) a kurzor fölötti sorba:
   - Végül illesszük vissza a kijelölést (`stice`) a parancssorba:
3. Tegyük fel, hogy egy `tmux` munkamenetet (a mi munkamenetünket) szeretnénk megosztani egy másik felhasználóval. Létrehoztuk a socketet (`/tmp/our_socket`) a megfelelő jogosultságokkal, hogy mind mi, mind a másik felhasználó írhasson és olvashasson. Milyen további két

feltételnek kell teljesülnie ahhoz, hogy a második felhasználó sikeresen csatolni tudja a munkamenetet az `tmux -S /tmp/our_socket a -t our_session` segítségével?

Összefoglalás

Ebben a leckében a *terminális multiplexerekről* tanultunk általánosságban, különösen a GNU Screenről és a tmuxról. A fontos fogalmak, amelyekre emlékeznünk kell:

- Parancs prefix: `screen` esetén: `Ctrl` + `a` + *karakter*; `tmux`, `Ctrl` + `b` + *karakter*.
- A munkamenetek, ablakok és ablakfelosztások (régiók vagy panelek) szerkezete.
- Másolási mód.
- Munkamenet leválasztás: a multiplexerek egyik leghatékonyabb funkciója

A leckében használt parancsok:

`screen`
: Elindít egy screen munkamenetet.

`tmux`
: Elindít egy `tmux` munkamenetet.

Válaszok a gyakorló feladatokra

1. Jelöljük meg, hogy a következő utasítások/funkciók a GNU Screenhez, a tmuxhoz vagy mindkettőhöz tartoznak-e:

Funkció/utasítás	**GNU Screen**	**tmux**
Az alapértelmezett prefix: `Ctrl` + `a`	x	
Kliens-szerver modell		x
A panelek pszeudo-terminálok		x
Egy régió terminálása nem terminálja a hozzátartozó ablako(ka)t	x	
A munkamenetekbe beletartoznak az ablakok	x	x
A munkamenetek leválaszthatók	x	x

2. Telepítsük a GNU Screen-t (csomag neve: `screen`) és végezzük el a következő feladatokat:
   - Indítsuk el a programot! Milyen parancsot használunk?

     `screen`

   - Indítsuk el a `top`-ot:

     `top`

   - A prefix segítségével nyissunk meg egy új ablakot, majd nyissuk meg a `/etc/screenrc`-t a `vi` használatával:

     `Ctrl` + `a` - `c`

     `sudo vi /etc/screenrc`

   - Listázzuk ki az ablakokat a képernyő alján:

     `Ctrl` + `a` - `w`

   - Változtassuk meg az aktuális ablak nevét `vi`-ra:

`Ctrl` + `a`-`A`. Majd íruk be a `vi`-t és nyomjuk meg az `enter`-t.

- A fennmaradó ablak nevét változtassuk meg `top`-ra. Ehhez először jelenítsük meg az összes ablak listáját, hogy fel-le mozoghassunk és kiválaszthassuk a megfelelőt:

  Először: `Ctrl` + `a`-`"`. Majd a nyilak segítségével válasszuk ki azt, amelyik `0 bash` és nyomjuk meg az `enter`-t. Végül: `Ctrl` + `a`-`A`, írjuk be, hogy `top` és `enter`.

- Ellenőrizzük, hogy a nevek megváltoztak-e, úgy, hogy az ablaknevek ismét megjelennek a képernyő alján:

  `Ctrl` + `a`-`w`

- Válasszuk le a munkamenetet és a `screen` hozzon létre egy újat `ssh` néven:

  `Ctrl` + `a`-`d` `screen -S "ssh"` majd `enter`.

- Válasszuk le az `ssh`-t is és listázzuk ki a munkamenetet a `screen` segítségével:

  `Ctrl` + `a`-`d` `screen -list` vagy `screen -ls`.

- Csatlakozzunk az első munkamenethez a PID-jével:

  ```
  screen -r PID-OF-SESSION
  ```

- Térjünk vissza a `top` ablakhoz. Osszuk fel az ablakot vízszintesen, és lépjünk az új üres régióba:

  `Ctrl` + `a`-`S`

  `Ctrl` + `a`-`Tab`

- Listáztassuk ki az összes ablakot a `screen` segítségével és válasszuk ki a `vi`-t, hogy jelenjen meg az üres régióban:

  Használjuk a `Ctrl` + `a`-`"` parancsot, hogy az összes ablakot megjelenítsük kiválasztásra, jelöljük ki a `vi`-t és nyomjuk meg az `enter` billentyűt.

- Osszuk fel függőlegesen a jelenlegi régiót, lépjünk át az újonnan létrehozott üres régióba és társítsuk azt egy vadonatúj ablakhoz:

  `Ctrl` + `a`-`|`

  `Ctrl` + `a`-`Tab`

`Ctrl` + `a` - `c`

- Terminálják az összes régiót, kivéve az aktuálisat (ne feledjük, attól, hogy a régiókat eltüntetjük, az ablakok megmaradnak)! Ezután lépjünk ki az aktuális munkamenet összes ablakából, amíg maga a munkamenet nem terminálódik:

  `Ctrl` + `a` - `Q`. `exit` (a Bash-ból való kilépéshez). `Shift` + `:`, majd írjuk be, hogy `quit`, majd `enter` (a `vi` -ból való kilépéshez). Ezután `exit` (az alapértelmezett Bash shellből való kilépéshez) `q` (a `top` terminálása); majd `exit` (az alapértelmezett Bash shellből való kilépéshez).

- Végül, még egyszer listáztassuk ki a `screen` segítségével az összes munkamenet, terminálják a megmaradt `ssh` munkamenetat PID-del és ellenőrizzük, hogy nem maradt több munkamanet:

  `screen -list` vagy `screen -ls`

  `screen -S PID-OF-SESSION -X quit`

  `screen -list` vagy `screen -ls`

3. Telepítsük a `tmux`-ot (csomagnév: `tmux`) és fejezzük be az alábbi feladatokat:

   - Indítsuk el a programot! Milyen parancsot kell használnunk?

     `tmux`

   - Indítsuk el a `top`-ot (vegyük észre, hogy — néhány másodpercen belül — az ablak neve megváltozik `top`-ra az állapotsoron):

     `top`

   - A tmux prefixét használva nyissunk meg egy új ablakot, majd hozzuk létre a `~/.tmux.conf` fájlt a `nano` segítségével:

     `Ctrl` + `b` - `c` `nano ~/.tmux.conf`

   - Osszuk fel az ablakot függőlegesen, és csökkentsük az újonnan létrehozott panel méretét néhányszor:

     `Ctrl` + `b` - `"`

     `Ctrl` + `b` - `Ctrl` + `↓`

   - Változtassuk meg az aktuális ablak nevét `text editing`-re; majd listáztassuk ki a `tmux` összes munkamenetét:

`Ctrl` + `b`-`,`. Majd adjuk meg az új nevet és `enter`. `Ctrl` + `b`-`s` vagy `tmux ls`.

- Ugorjunk át a `top` ablakra, majd vissza az aktuálisra, ugyanazon billentyűkombináció használatával:

  `Ctrl` + `b`-`n` vagy `Ctrl` + `b`-`p`

- Csatlakozzunk le az aktuális munkamenetről és hozzunk létre egy újat, aminek a neve `ssh`, az ablakának a neve pedig `ssh window`:

  `Ctrl` + `b`-`d` `tmux new -s "ssh" -n "ssh window"`

- Csatlakozzunk le az `ssh` munkamenetről és a `tmux` jelenítse meg újra az összes munkamenetét:

  `Ctrl` + `b`-`d` `tmux ls`

> **NOTE** Ettől a ponttól kezdve a gyakorlat megköveteli, hogy egy *remote* (távoli) gépet használjunk a helyi géphez történő `ssh` kapcsolódásokhoz (egy virtuális gép is tökéletesen megfeel és nagyon praktikus is lehet). Győződjünk meg róla, hogy a helyi gépen telepítve van és fut az `openssh-server`, valamint arról, hogy a távoli gépen legalább az `openssh-client` telepítve van.

- Most indítsuk el a távoli gépet és csatlakozzunk hozzá `ssh`-val a helyi gépről. Ha a kapcsolat létrejött, ellenőrizzük a `tmux` munkameneteket:

  Távoli hoszton: `ssh local-username@local-ipaddress`. Ha csatlakozott a lokális géphez: `tmux ls`.

- A távoli gépen csatlakozzunk az `ssh` munkamenethez a nevével:

  `tmux a -t ssh` (az `a` helyett lehet `at` vagy `attach` is).

- A saját gépünkön csatlakozzunk az `ssh` munkamenethez a nevével úgy, hogy először terminálódjon a távoli kapcsolat:

  `tmux a -d -t ssh` (az `a` helyett lehet `at` vagy `attach` is).

- Jelenítsük meg az összes munkamenetet és lépjünk az első munkamenetre (`[0]`). Ha már ott vagyunk, terminálju az `ssh` munkamenetet a nevével:

  Először `Ctrl` + `b`-`s`, majd a nyilak segítségével megkeressük a `0` munkamenetet és `enter` `tmux kill-session -t ssh`.

- Végül csatlakozzunk le az aktuális munkamenetről és terminálju k a nevével:

  `Ctrl` + `b` - `d` `tmux kill-session -t 0`.

Válaszok a gondolkodtató feladatokra

1. A `screen` és a `tmux` is beléphet parancssori módba a _command prefix + `:` segítségével (a `tmux` esetében már láttunk egy rövid példát). Végezzünk némi kutatást és hajtsuk végre a következő feladatokat parancssori üzemmódban:
   - A `screen` lépjen be a copy módba:

     `Ctrl` + `a` - `:` — majd íruk be, hogy `copy`.
   - A `tmux` nevezze át az aktuális ablakot:

     `Ctrl` + `b` - `:` — majd íruk be, hogy `rename-window`.
   - A `screen` zárja be az összes ablakot és terminálja a munkamenetet:

     `Ctrl` + `a` - `:` — majd íruk be, hogy `quit`.
   - A `tmux` válassza ketté a panelt:

     `Ctrl` + `b` - `:` — majd íruk be, hogy `split-window`.
   - A `tmux` zárja be az aktuális ablakot:

     `Ctrl` + `b` - `:` — majd íruk be, hogy `kill-window`.
2. Amikor a `screen` másolási módba lép, nem csak a nyílbillentyűkkel és a `PgUP` vagy `PgDown` billentyűkkel navigálhatunk az aktuális ablakban és a scrollback pufferben. Lehetőség van egy vi-szerű teljes képernyős szerkesztő használatára is. Ezzel a szerkesztővel a következő feladatokat végezhetjük el:
   - Echo `supercalifragilisticexpialidocious` a `screen` terminálban:

     ```
     echo supercalifragilisticexpialidocious
     ```
   - Most másoljuk az öt egymást követő karaktert (balról jobbra) a kurzor fölötti sorba:

     Belépés a copy módba: `Ctrl` + `a` - `[` vagy `Ctrl` + `a` - `:` majd íruk be, hogy `copy`. Ezután a `k`-val lépjünk a fölső sorba és a `space` lenyomásával kezdjük el a kijelölést. Végül menjünk végig a négy karakteren a `l` segítségével és nyomjuk meg a `space` billentyűt ismét a kijelölés befejezéséhez.
   - Végül illesszük vissza a kijelölést (`stice`) a parancssorba:

     `Ctrl` + `a` - `]`

3. Tegyük fel, hogy egy `tmux` munkamenetet (a mi munkamenetünket) szeretnénk megosztani egy másik felhasználóval. Létrehoztuk a socketet (`/tmp/our_socket`) a megfelelő jogosultságokkal, hogy mind mi, mind a másik felhasználó írhasson és olvashasson. Milyen további két feltételnek kell teljesülnie ahhoz, hogy a második felhasználó sikeresen csatolni tudja a munkamenetet az `tmux -S /tmp/our_socket a -t our_session` segítségével?

   Mindkét felhasználónak rendelkeznie kell egy közös csoporttal, pl. `multiplexer`. Ezután a socketet is át kell helyeznünk ebbe a csoportba: `chgrp multiplexer /tmp/our_socket`.

103.6 A folyamatok végrehajtási prioritásának módosítása

Hivatkozás az LPI célkitűzésre

LPIC-1 v5, Exam 101, Objective 103.6

Súlyozás

2

Kulcsfontosságú ismeretek

- A létrehozott job alapértelmezett prioritásának ismerete.
- Program futtatása az alapértelmezettnél magasabb vagy alacsonyabb prioritással.
- Egy futó folyamat prioritásának módosítása.

A használt fájlok, kifejezések és segédprogramok listája

- `nice`
- `ps`
- `renice`
- `top`

Linux Professional Institute

103.6 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	103 GNU és Unix parancsok
Fejezet:	103.6 A folyamatok végrehajtási prioritásának módosítása
Lecke:	1/1

Bevezetés

Az egynél több folyamat egyidejű futtatására képes operációs rendszereket többfeladatos vagy többprocesszoros rendszereknek nevezzük. Míg a valódi egyidejűség csak akkor valósul meg, ha egynél több feldolgozóegység (processing unit) áll rendelkezésre, az egyprocesszoros rendszerek is képesek szimulálni az egyidejűséget azáltal, hogy nagyon gyorsan váltanak a folyamatok között. Ezt a technikát alkalmazzák a sok egyenértékű processzorral rendelkező rendszerekben, vagy *szimmetrikus többprocesszoros (SMP)* rendszerekben is, mivel a lehetséges egyidejű folyamatok száma jóval meghaladja a rendelkezésre álló processzoregységek számát.

Valójában egyszerre csak egy folyamat irányíthatja a CPU-t. A legtöbb folyamat tevékenysége azonban *rendszerhívás* (system call), azaz a futó folyamat átadja a CPU-irányítást az operációs rendszer valamelyik folyamatának, hogy az elvégezze a kért műveletet. A rendszerhívások felelősek minden eszközök közötti kommunikációért, például a memóriafoglalásokért, a fájlrendszerekből történő olvasásért és írásért, a képernyőre történő szövegmegjelenítésért, a felhasználói interakciókért, a hálózati átvitelért, stb. A CPU-vezérlés átadása egy rendszerhívás során lehetővé teszi az operációs rendszer számára annak eldöntését, hogy visszaadja-e a CPU-vezérlést az előző folyamatnak, vagy átadja-e azt egy másiknak. Mivel a modern CPU-k sokkal

gyorsabban képesek utasításokat végrehajtani, mint ahogy a legtöbb külső hardver kommunikálni tud egymással, egy új vezérlőfolyamat sok CPU-munkát végezhet el, miközben a korábban kért hardverválaszok még mindig nem állnak rendelkezésre. A CPU maximális kihasználásának biztosítása érdekében a többprocesszoros operációs rendszerek dinamikus sorban tartják az aktív folyamatokat, amelyek CPU-időre várnak.

Bár ezek lehetővé teszik a CPU-idő kihasználtságának jelentős javítását, a folyamatok közötti váltás kizárólag a rendszerhívásokra támaszkodva nem elegendő a kielégítő többfeladatos teljesítmény eléréséhez. Egy olyan folyamat, amely nem hajt végre rendszerhívásokat, a végtelenségig irányíthatja a CPU-t. Ezért a modern operációs rendszerek *preemptívek* is, azaz egy futó folyamatot vissza lehet tenni a sorba, hogy egy fontosabb folyamat irányíthassa a CPU-t, még akkor is, ha a futó folyamat nem hajtott végre rendszerhívást.

A Linux ütemezője

A Linux, mint preemptív többprocesszoros operációs rendszer, egy ütemezőt (scheduler) valósít meg, amely a folyamatok sorát szervezi. Pontosabban az ütemező dönti el azt is, hogy melyik sorba állított *szál* (thread) kerüljön végrehajtásra — egy folyamat több független szálra is elágazhat — de a folyamat és a szál ebben a kontextusban felcserélhető fogalmak. Minden folyamatnak két predikátuma van, amely beavatkozik az ütemezésébe: az *ütemezési irányelv* (scheduling policy) és az *ütemezési prioritás* (scheduling priority).

Az ütemezési irányelveknek két fő típusa van: *valós idejű irányelvek* és *általános irányelvek*. A valós idejű irányelvek alá tartozó folyamatok ütemezése közvetlenül a prioritási értékük alapján történik. Ha egy fontosabb folyamat készen áll a futtatásra, egy kevésbé fontos futó folyamatot megelőz és a magasabb prioritású folyamat átveszi a CPU feletti irányítást. Egy alacsonyabb prioritású folyamat csak akkor kapja meg a CPU irányítását, ha a magasabb prioritású folyamatok üresjáratban vannak vagy hardveres válaszra várnak.

Minden valós idejű folyamat magasabb prioritással rendelkezik, mint egy normál folyamat. Általános célú operációs rendszerként a Linux csak néhány valós idejű folyamatot futtat. A legtöbb folyamat, beleértve a rendszer- és felhasználói programokat is, normál ütemezési szabályok szerint fut. A normál folyamatok általában azonos prioritási értékkel rendelkeznek, de a normál házirendek egy másik folyamatpredikátummal: a *nice value* (jó érték) segítségével határozhatják meg a végrehajtási prioritási szabályokat. A nice értékekből származtatott dinamikus prioritásokkal való összetévesztés elkerülése érdekében az ütemezési prioritásokat általában *statikus* ütemezési prioritásoknak nevezik.

A Linux ütemezője sokféleképpen konfigurálható, és a prioritások megállapításának még bonyolultabb módjai is léteznek, de ezek az általános fogalmak mindig érvényesek. A folyamatok ütemezésének vizsgálatakor és hangolásakor fontos szem előtt tartani, hogy csak a normál

ütemezési irányelvek alá tartozó folyamatokat érinti.

Prioritások ellenőrzése

A Linux 0 és 99 közötti statikus prioritásokat tart fenn a valós idejű folyamatok számára, a normál folyamatok pedig 100 és 139 közötti statikus prioritásokat kapnak, ami azt jelenti, hogy a normál folyamatok számára 39 különböző prioritási szint létezik. Az alacsonyabb értékek magasabb prioritást jelentenek. Egy aktív folyamat statikus prioritása a `sched` fájlban található, a `/proc` fájlrendszeren belül a megfelelő mappában:

```
$ grep ^prio /proc/1/sched
prio                  :       120
```

Ahogy a példában látható, a `prio` kezdetű sor a folyamat prioritási értékét adja meg (a PID 1 folyamat a *init* vagy *systemd* folyamat, az első folyamat, amelyet a kernel a rendszer inicializálása során elindít). A normál folyamatok szabványos prioritása 120, így ez 100-ra csökkenthető vagy 139-re növelhető. Az összes futó folyamat prioritása ellenőrizhető a `ps -Al` vagy `ps -el` paranccsal:

```
$ ps -el
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 -  9292 -      ?        00:00:00 systemd
4 S     0    19     1  0  80   0 -  8817 -      ?        00:00:00 systemd-journal
4 S   104    61     1  0  80   0 - 64097 -      ?        00:00:00 rsyslogd
4 S     0    63     1  0  80   0 -  7244 -      ?        00:00:00 cron
1 S     0   126     1  0  80   0 -  4031 -      ?        00:00:00 dhclient
4 S     0   154     1  0  80   0 -  3937 -      pts/0    00:00:00 agetty
4 S     0   155     1  0  80   0 -  3937 -      pts/1    00:00:00 agetty
4 S     0   156     1  0  80   0 -  3937 -      pts/2    00:00:00 agetty
4 S     0   157     1  0  80   0 -  3937 -      pts/3    00:00:00 agetty
4 S     0   158     1  0  80   0 -  3937 -      console  00:00:00 agetty
4 S     0   160     1  0  80   0 - 16377 -      ?        00:00:00 sshd
4 S     0   280     0  0  80   0 -  5301 -      ?        00:00:00 bash
0 R     0   392   280  0  80   0 -  7221 -      ?        00:00:00 ps
```

A `PRI` oszlop a kernel által kiosztott statikus prioritást jelzi. Figyeljük meg azonban, hogy a `ps` által megjelenített prioritás értéke eltér az előző példában kapott értéktől. Történelmi okok miatt a `ps` által megjelenített prioritások alapértelmezés szerint -40 és 99 között mozognak, így a tényleges prioritást úgy kapjuk meg, hogy hozzáadunk 40-et (80 + 40 = 120).

Lehetőség van a Linux kernel által jelenleg kezelt folyamatok folyamatos megfigyelésére is a `top` programmal. A `ps`-hez hasonlóan a `top` is másképp jeleníti meg a prioritás értékét. A valós idejű folyamatok könnyebb azonosítása érdekében a `top` a prioritás értékét 100-zal csökkenti, így minden valós idejű prioritás negatív lesz, és egy negatív szám vagy *rt* jelöli őket. Ezért a `top` által megjelenített normál prioritások 0-tól 39-ig terjednek.

NOTE

A `ps` parancs még részletesebb lehet, ha további kapcsolókat használunk. Hasonlítsuk össze ennek a parancsnak a kimenetét az előző példánk kimenetével:

```
$ ps -e -o user,uid,comm,tty,pid,ppid,pri,pmem,pcpu --sort=-pcpu | head
```

Processzek jósága (niceness)

Minden normál folyamat alapértelmezett 0-s nice értékkel (120-as prioritás) indul. A *nice* név abból az elképzelésből ered, hogy a "szebb" folyamatok lehetővé teszik, hogy más folyamatok előttük fussanak egy adott végrehajtási sorban. A nice számok -20-tól (kevésbé nice, magas prioritás) 19-ig (magasabb nice, alacsony prioritás) terjednek. A Linux azt is lehetővé teszi, hogy ugyanazon folyamaton belül különböző nice értékeket rendeljünk a szálakhoz. A `ps` kimenet `NI` oszlopa a *nice* számot jelzi.

Csak a root felhasználó csökkentheti a folyamatok nice értékét nulla alá. Lehetőség van arra, hogy egy folyamatot nem szabványos prioritással indítsunk el a `nice` paranccsal. Alapértelmezés szerint a `nice` a niceness-t 10-re változtatja, de ez a `-n` kapcsolóval megadható:

```
$ nice -n 15 tar czf home_backup.tar.gz /home
```

Ebben a példában a `tar` parancsot 15-ös niceness-szel hajtjuk végre. A `renice` paranccsal egy futó folyamat prioritását lehet megváltoztatni. A `-p` kapcsoló a célfolyamat PID-számát adja meg. Például:

```
# renice -10 -p 2164
2164 (process ID) old priority 0, new priority -10
```

A `-g` és `-u` kapcsolók egy adott csoport vagy felhasználó összes folyamatának módosítására szolgálnak. A `renice +5 -g users` paranccsal a *users* csoport felhasználóinak tulajdonában lévő folyamatok nice értéke öttel nő.

A `renice` mellett a folyamatok prioritása más programokkal is módosítható, például a `top` programmal. A top főképernyőn a folyamatok nice értékét az `r`, majd a folyamat PID-számának

megadásával lehet módosítani:

```
top - 11:55:21 up 23:38,  1 user,  load average: 0,10, 0,04, 0,05
Tasks:  20 total,   1 running,  19 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0,5 us,  0,3 sy,  0,0 ni, 99,0 id,  0,0 wa,  0,2 hi,  0,0 si,  0,0 st
KiB Mem :  4035808 total,   774700 free,  1612600 used,  1648508 buff/cache
KiB Swap:  7999828 total,  7738780 free,   261048 used.  2006688 avail Mem
PID to renice [default pid = 1]
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    1 root      20   0   74232   7904   6416 S 0,000 0,196   0:00.12 systemd
   15 root      20   0   67436   6144   5568 S 0,000 0,152   0:00.03 systemd-journal
   21 root      20   0   61552   5628   5000 S 0,000 0,139   0:00.01 systemd-logind
   22 message+  20   0   43540   4072   3620 S 0,000 0,101   0:00.03 dbus-daemon
   23 root      20   0   45652   6204   4992 S 0,000 0,154   0:00.06 wickedd-dhcp4
   24 root      20   0   45648   6276   5068 S 0,000 0,156   0:00.06 wickedd-auto4
   25 root      20   0   45648   6272   5060 S 0,000 0,155   0:00.06 wickedd-dhcp6
```

A `PID to renice [default pid = 1]` üzenet jelenik meg, és alapértelmezés szerint az első felsorolt folyamat van kiválasztva. Egy másik folyamat prioritásának megváltoztatásához írjuk be annak PID-jét, és nyomjuk le az Enter billentyűt. Ekkor megjelenik a `Renice PID 1 to value` üzenet (a kért PID számmal), és egy új nice értéket lehet hozzárendelni.

Gyakorló feladatok

1. Mi történik egy preemptív többfeladatos rendszerben, amikor egy alacsonyabb prioritású folyamat foglalja le a processzort, és egy magasabb prioritású folyamat sorban áll a végrehajtásra?

2. Nézzük az alábbi `top`-ot:

```
top - 08:43:14 up 23 days, 12:29,  5 users,  load average: 0,13, 0,18, 0,21
Tasks: 240 total,   2 running, 238 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1,4 us,  0,4 sy,  0,0 ni, 98,1 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   7726,4 total,    590,9 free,   1600,8 used,   5534,7 buff/cache
MiB Swap:  30517,0 total,  30462,5 free,     54,5 used.   5769,4 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    1 root      20   0  171420  10668   7612 S   0,0   0,1   9:59.15 systemd
    2 root      20   0       0      0      0 S   0,0   0,0   0:02.76 kthreadd
    3 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_par_gp
    8 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 mm_percpu_wq
    9 root      20   0       0      0      0 S   0,0   0,0   0:49.06 ksoftirqd/0
   10 root      20   0       0      0      0 I   0,0   0,0  18:24.20 rcu_sched
   11 root      20   0       0      0      0 I   0,0   0,0   0:00.00 rcu_bh
   12 root      rt   0       0      0      0 S   0,0   0,0   0:08.17 migration/0
   14 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/0
   15 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/1
   16 root      rt   0       0      0      0 S   0,0   0,0   0:11.79 migration/1
   17 root      20   0       0      0      0 S   0,0   0,0   0:26.01 ksoftirqd/1
```

   Mely PID-eknek van valós idejű prioritása?

3. Nézzük az alábbi `ps -el` listát:

```
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 42855 -      ?        00:09:59 systemd
1 S     0     2     0  0  80   0 -     0 -      ?        00:00:02 kthreadd
1 I     0     3     2  0  60 -20 -     0 -      ?        00:00:00 rcu_gp
1 S     0     9     2  0  80   0 -     0 -      ?        00:00:49 ksoftirqd/0
1 I     0    10     2  0  80   0 -     0 -      ?        00:18:26 rcu_sched
```

```
1 I     0    11     2  0  80   0 -     0 -      ?        00:00:00 rcu_bh
1 S     0    12     2  0 -40   - -     0 -      ?        00:00:08 migration/0
1 S     0    14     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/0
5 S     0    15     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/1
```

Melyik PID-nek van magasabb prioritása?

4. Miután megpróbáltunk egy processzt renice-olni a `renice` segítségével, az alábbi hiba jelentkezik:

```
$ renice -10 21704
renice: failed to set priority for 21704 (process ID): Permission denied
```

Mi a hiba legvalószínűbb oka?

Gondolkodtató feladatok

1. A folyamatprioritások megváltoztatására általában akkor van szükség, ha egy folyamat túl sok CPU-időt foglal le. A `ps` standard kapcsolókkal történő használata az összes rendszerfolyamatot hosszú formátumban írja ki, melyik `--sort` flag rendezné a folyamatokat CPU kihasználtság szerint, növekvő sorrendben?

2. A `schedtool` paranccsal beállíthatjuk az összes CPU ütemezési paramétert, amire a Linux képes, vagy megjeleníthetjük az adott folyamatokra vonatkozó információkat. Hogyan lehet vele megjeleníteni az `1750` folyamat ütemezési paramétereit? Továbbá, hogyan lehet a `schedtool` segítségével a 1750-es folyamatot -90-es prioritású valós idejűvé változtatni (ahogyan azt a `top` mutatja)?

Összefoglalás

Ez a lecke azt tárgyalta, hogy a Linux hogyan osztja meg a CPU-időt a kezelt folyamatok között. A legjobb teljesítmény biztosítása érdekében a kritikusabb folyamatoknak meg kell előzniük a kevésbé kritikus folyamatokat. A leckében a következőkről volt szó:

- Alapfogalmak a többprocesszoros rendszerekről.
- Mi az a folyamatütemező és hogyan valósítja meg a Linux.
- Mik a Linux prioritások, nice számok és mi a céljuk.
- Hogyan olvassuk és értelmezzük a folyamatprioritásokat Linuxban.
- Hogyan változtathatjuk meg egy folyamat prioritását a végrehajtás előtt és közben.

Válaszok a gyakorló feladatokra

1. Mi történik egy preemptív többfeladatos rendszerben, amikor egy alacsonyabb prioritású folyamat foglalja le a processzort, és egy magasabb prioritású folyamat sorban áll a végrehajtásra?

   Az alacsonyabb prioritású folyamat szünetel, és helyette a magasabb prioritású folyamat kerül végrehajtásra.

2. Nézzük az alábbi `top`-ot:

```
top - 08:43:14 up 23 days, 12:29,  5 users,  load average: 0,13, 0,18, 0,21
Tasks: 240 total,   2 running, 238 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1,4 us,  0,4 sy,  0,0 ni, 98,1 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   7726,4 total,    590,9 free,   1600,8 used,   5534,7 buff/cache
MiB Swap:  30517,0 total,  30462,5 free,     54,5 used.   5769,4 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    1 root      20   0  171420  10668   7612 S   0,0   0,1   9:59.15 systemd
    2 root      20   0       0      0      0 S   0,0   0,0   0:02.76 kthreadd
    3 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_par_gp
    8 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 mm_percpu_wq
    9 root      20   0       0      0      0 S   0,0   0,0   0:49.06 ksoftirqd/0
   10 root      20   0       0      0      0 I   0,0   0,0  18:24.20 rcu_sched
   11 root      20   0       0      0      0 I   0,0   0,0   0:00.00 rcu_bh
   12 root      rt   0       0      0      0 S   0,0   0,0   0:08.17 migration/0
   14 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/0
   15 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/1
   16 root      rt   0       0      0      0 S   0,0   0,0   0:11.79 migration/1
   17 root      20   0       0      0      0 S   0,0   0,0   0:26.01 ksoftirqd/1
```

   Mely PID-eknek van valós idejű prioritása?

   12 és 16.

3. Nézzük az alábbi `ps -el` listát:

```
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 42855 -      ?        00:09:59 systemd
1 S     0     2     0  0  80   0 -     0 -      ?        00:00:02 kthreadd
1 I     0     3     2  0  60 -20 -     0 -      ?        00:00:00 rcu_gp
```



```
1 S     0     9     2  0  80   0 -     0 -      ?        00:00:49 ksoftirqd/0
1 I     0    10     2  0  80   0 -     0 -      ?        00:18:26 rcu_sched
1 I     0    11     2  0  80   0 -     0 -      ?        00:00:00 rcu_bh
1 S     0    12     2  0 -40   - -     0 -      ?        00:00:08 migration/0
1 S     0    14     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/0
5 S     0    15     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/1
```

Melyik PID-nek van magasabb prioritása?

12.

4. Miután megpróbáltunk egy processzt renice-olni a `renice` segítségével, az alábbi hiba jelentkezik:

```
$ renice -10 21704
renice: failed to set priority for 21704 (process ID): Permission denied
```

Mi a hiba legvalószínűbb oka?

Csak a root tudja a nice számokat nulla alá csökkenteni.

Válaszok a gondolkodtató feladatokra

1. A folyamatprioritások megváltoztatására általában akkor van szükség, ha egy folyamat túl sok CPU-időt foglal le. A `ps` standard kapcsolókkal történő használata az összes rendszerfolyamatot hosszú formátumban írja ki, melyik `--sort` flag rendezné a folyamatokat CPU kihasználtság szerint, növekvő sorrendben?

```
$ ps -el --sort=pcpu
```

2. A `schedtool` paranccsal beállíthatjuk az összes CPU ütemezési paramétert, amire a Linux képes, vagy megjeleníthetjük az adott folyamatokra vonatkozó információkat. Hogyan lehet vele megjeleníteni az `1750` folyamat ütemezési paramétereit? Továbbá, hogyan lehet a `schedtool` segítségével a 1750-es folyamatot -90-es prioritású valós idejűvé változtatni (ahogyan azt a `top` mutatja)?

```
$ schedtool 1750
```

```
$ schedtool -R -p 89 1750
```


103.7 Keresés szövegfájlokban reguláris kifejezések segítségével

Hivatkozás az LPI célkitűzésre

LPIC-1 v5, Exam 101, Objective 103.7

Súlyozás

3

Kulcsfontosságú ismeretek

- Több jelölőelemet tartalmazó egyszerű szabályos kifejezések létrehozása.
- Az alapvető és a kiterjesztett reguláris kifejezések közötti különbségek megértése.
- A speciális karakterek, karakterosztályok, kvantorok és horgonyok fogalmának megértése.
- A reguláris kifejezések eszközeinek használata a fájlrendszerben vagy a fájlok tartalmában történő keresések elvégzéséhez.
- Reguláris kifejezések használata szöveg törlésére, módosítására és helyettesítésére.

A használt fájlok, kifejezések és segédprogramok listája

- `grep`
- `egrep`
- `fgrep`
- `sed`
- `regex(7)`

Linux Professional Institute

103.7 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	103 GNU és Unix parancsok
Fejezet:	103.7 Keresés szövegfájlokban reguláris kifejezések segítségével
Lecke:	1/2

Bevezetés

A sztring-kereső algoritmusokat számos adatfeldolgozási feladat széles körben használja, olyannyira, hogy a Unix-szerű operációs rendszereknek saját, mindenütt jelenlévő implementációjuk van: *Reguláris kifejezések*, (regular expressions) gyakran rövidítve *REs*. A reguláris kifejezések olyan karaktersorozatokból állnak, amelyek egy általános mintát alkotnak, amelyet egy nagyobb karaktersorozatban lévő megfelelő sorozat megkeresésére és néha módosítására használnak. A reguláris kifejezések nagymértékben kibővítik a következő lehetőségeket:

- Elemzési szabályok írása HTTP-kiszolgálók, különösen az *nginx* kéréseihez.
- Szkriptek írása, amelyek szöveges alapú adathalmazokat konvertálnak más formátumba.
- Érdekes előfordulások keresése logokban vagy dokumentumokban.
- Markup dokumentumok szűrése, a szemantikus tartalom megtartása.

A legegyszerűbb reguláris kifejezés legalább egy *atom*-ot tartalmaz. Az atom, amely azért kapta ezt a nevet, mert ez a reguláris kifejezés alapeleme, nem más, mint egy karakter, amelynek lehet

speciális jelentése, de lehet, hogy nincs. A legtöbb közönséges karakter egyértelmű, megtartják szó szerinti jelentésüket, míg másoknak speciális jelentésük van:

`.` (pont)

Az atom illeszkedik bármely karakterre.

`^` (caret)

Az atom illeszkedik a sor elejével.

`$` (dollárjel)

Az atom illeszkedik a sor végével.

Például a `bc` reguláris kifejezés, amely a `b` és `c` literális atomokból áll, megtalálható az `abcd` karakterláncban, de nem található meg az `a1cd` karakterláncban. Másrészt a `.c` reguláris kifejezés megtalálható mind az `abcd`, mind az `a1cd` karakterláncban, mivel a `.` pont bármely karakterrel megegyezik.

A caret és a dollárjel atomokat akkor használjuk, ha csak a karakterlánc elején vagy végén lévő találatok érdekesek. Emiatt *anchors*-nak (horgony) is nevezik őket. Például a `cd` megtalálható az `abcd`-ben, de a `^cd` nem. Hasonlóképpen, az `ab` megtalálható az `abcd`-ben, de az `ab$` nem. A `^` caret szó szerinti karakter, kivéve, ha az elején áll, és a `$` is szó szerinti karakter, kivéve, ha a reguláris kifejezés végén áll.

Zárójeles kifejezés

Létezik egy másik atomtípus is, amelynek neve *bracket expression* (zárójeles kifejezés). Bár nem egyetlen karakter, a zárójelek `[]` (tartalmukkal együtt) egyetlen atomnak tekinthetők. A zárójeles kifejezés általában csak egy lista szó szerinti karakterekből, amelyeket `[]` zár be, így az atom a lista bármelyik karakterével megegyezik. Például a `[1b]` kifejezés megtalálható mind az `abcd`, mind az `a1cd` karakterláncban. Ha meg akarjuk adni, hogy az atom ne feleljen meg a karaktereknek, a listának `^`-vel kell kezdődnie, mint a `[^1b]`-ben. Karaktertartományok megadása zárójeles kifejezésekben is lehetséges. Például a `[0-9]` megfelel a 0-tól 9-ig terjedő számjegyeknek, az `[a-z]` pedig bármely kisbetűs betűnek. A tartományokat óvatosan kell használni, mivel előfordulhat, hogy nem konzisztensek a különböző helyi tartományokban.

A zárójeles kifejezéslisták az egyes karakterek és tartományok helyett osztályokat is elfogadnak. A hagyományos karakterosztályok a következők:

`[:alnum:]`

Alfanumerikus karaktert reprezentál.

`[:alpha:]`
Alfabetikus karaktert reprezentál.

`[:ascii:]`
Az ASCII karakterkészletbe illeszkedő karaktert reprezentál.

`[:blank:]`
Egy üres karaktert, tabot vagy szóközt reprezentál.

`[:cntrl:]`
Egy control karaktert reprezentál.

`[:digit:]`
Egy számjegyet (0-9 között) reprezentál.

`[:graph:]`
Bármilyen nyomtatható karaktert (a szóköz kivételével) reprezentál.

`[:lower:]`
Egy kisbetűs karaktert reprezentál.

`[:print:]`
Bármilyen nyomtatható karaktert (beleértve a szóközt is) reprezentál.

`[:punct:]`
Bármilyen nyomtatható karaktert (kivéve a szóközt és az alfanumerikus karaktereket) reprezentál.

`[:space:]`
White-space karaktereket: space, form-feed (`\f`), newline (`\n`), carriage return (`\r`), horizontal tab (`\t`), és vertical tab (`\v`) karaktereket reprezentál.

`[:upper:]`
Egy nagybetűs betűt reprezentál.

`[:xdigit:]`
Hexadecimális számokat (0 és F között) reprezentál.

A karakterosztályok egyedülálló karakterekkel és tartományokkal kombinálhatók, de nem használhatók tartomány végpontjaként. Továbbá a karakterosztályok csak zárójeles kifejezésekben használhatók, a zárójelen kívül önálló atomként nem.

Kvantorok

Egy atom elérést, akár egykarakteres atom, akár zárójeles, egy *atom quantifier* (atom kvantor) segítségével lehet beállítani. Az atom kvantorok atom *szekvenciákat* határoznak meg, vagyis akkor történik találat, ha az atomnak egy összefüggő ismétlődése található a karakterláncban. Az egyezésnek megfelelő részláncát *darabnak* nevezzük. Ettől függetlenül a kvantorokat és a reguláris kifejezések egyéb jellemzőit másképp kezelik attól függően, hogy melyik szabványt használják.

A POSIX által meghatározottak szerint a reguláris kifejezéseknek két formája létezik: “basic” (alap) és “extended” (kiterjesztett) reguláris kifejezések. A legtöbb szöveggel kapcsolatos program bármely hagyományos Linux-disztribúcióban mindkét formát támogatja, ezért fontos ismerni a különbségeket a kompatibilitási problémák elkerülése és a tervezett feladathoz legmegfelelőbb implementáció kiválasztása érdekében.

A `*` kvantornak ugyanaz a funkciója az alap és a kiterjesztett REs-ben (az atom nulla vagy több alkalommal fordul elő), és szó szerinti karakter, ha a reguláris kifejezés elején jelenik meg, vagy ha a `\` backslash előzi meg. A `` ` pluszjel kvantor olyan darabokat választ ki, amelyek egymás után egy vagy több atomot tartalmaznak. A kérdőjeles kvantorral `?` akkor lesz egyezés, ha a megfelelő atom egyszer jelenik meg, vagy ha egyáltalán nem jelenik meg. Ha egy `\` backslash előzi meg ezeket, akkor a speciális jelentésüket nem vesszük figyelembe. Az alapvető reguláris kifejezések támogatják a ` `` és `?` kvantorokat is, de ezeket is meg kell előznie a backslashnek. A kiterjesztett reguláris kifejezésektől eltérően a `+` és a `?` önmagukban szó szerinti karakterek az alap reguláris kifejezésekben.

Korlátok

A *bound* (korlát) egy olyan kvantor, amely, mint a neve is mutatja, lehetővé teszi a felhasználó számára, hogy pontos mennyiségi korlátokat adjon meg egy atomhoz. A kiterjesztett reguláris kifejezésekben a bound háromféle formában jelenhet meg:

`{i}`

Az atomnak pontosan `i`-szer kell megjelennie (`i` egy egész szám). Például a `\[{2}` pontosan két üres karakterrel egyezik.

`{i,}`

Az atomnak legalább `i`-szer kell megjelennie (`i` egy egész szám). Például a `\[{2,}` bármely két vagy több üres karakterből álló sorozattal megegyezik.

`{i,j}`

Az atomnak legalább `i`-szer és legfeljebb `j`-szer kell megjelennie (`i` és `j` egész számok, `j` nagyobb, mint `i`). Például az `xyz{2,4}` az `xy` karakterláncot követi kettő-négy `z` karakter.

Bármilyen esetben, amikor egy substring megfelel egy reguláris kifejezésnek, és egy hosszabb, ugyanott kezdődő substring is megfelel, akkor a hosszabb substringet vesszük figyelembe.

Az alap reguláris kifejezések is támogatják a kolrátokat, de a delimiterek előtt `\`-nek kell állnia: `\{` és `\}`. A `{` és `}` önmagukban szó szerinti karakterekként értelmezhetők. A `\{`, amelyet egy számjegyen kívüli szó szerinti karakter követ, nem pedig egy korlát kezdőbetűje.

Branchek és visszautalások

Az alap reguláris kifejezések egy másik fontos szempontban is különböznek a kiterjesztett reguláris kifejezésektől: egy kiterjesztett reguláris kifejezés *ágakra* (branchekre) osztható, amelyek mindegyike egy-egy független reguláris kifejezés. Az ágak `|`-vel vannak elválasztva, és a kombinált reguláris kifejezés minden olyan kifejezésre illik, amely megfelel bármelyik ágnak. Például a `he|him` akkor talál, ha a vizsgált karakterláncban megtalálható a `he` vagy a `him` részlánc. Az alapvető reguláris kifejezések a `|` karaktert szó szerinti karakterként értelmezik. A legtöbb program azonban, amely támogatja az alapvető reguláris kifejezéseket, engedélyezi a `\|` ágakat.

A kiterjesztett reguláris kifejezést `()`-be zártan lehet használni egy *visszautalásban* (back reference). Például a `([[:digit:]])\1` minden olyan reguláris kifejezésre illik, amely legalább egyszer megismétli önmagát, mivel a kifejezésben szereplő `\1` az első zárójeles részkifejezés által lefedett darab visszautalása. Ha a szabályos kifejezésben egynél több zárójeles részkifejezés van, akkor így hivatkozhatunk rájuk: `\2`, `\3` és így tovább.

Az alapvető REs-ek esetében a részkifejezéseket `\(` és `\)` karakterekkel kell körülvenni, a `(` és `)` pedig önmagukban közönséges karakterek. A visszautalás jelzőjét a kiterjesztett reguláris kifejezésekhez hasonlóan használjuk.

Keresés a reguláris kifejezésekkel

A reguláris kifejezések közvetlen előnye a fájlrendszerekben és a szöveges dokumentumokban való keresés javítása. A `find` parancs `-regex` opciója lehetővé teszi, hogy a mappastruktúra minden útvonalát egy reguláris kifejezéssel teszteljük. Például,

```
$ find $HOME -regex '.*/\..*' -size +100M
```

100 megabájtnál (100 egységnyi 1048576 bájt) nagyobb fájlokat keres, de csak a felhasználó home mappáján belüli olyan elérési utakban, amelyek tartalmaznak egyezést a `.*/\..*'-vel, azaz egy `/.`, amelyet bármilyen más karakter vesz körül. Más szóval, csak a rejtett fájlok vagy a rejtett mappákban lévő fájlok kerülnek listázásra, függetlenül attól, hogy a `/.` milyen pozícióban van a megfelelő elérési útvonalban. Nagy- és kisbetű-független reguláris kifejezések esetén ehelyett az `-iregex` opciót kell használni:

```
$ find /usr/share/fonts -regextype posix-extended -iregex
'.*(dejavu|liberation).*sans.*(italic|oblique).*'
/usr/share/fonts/dejavu/DejaVuSansCondensed-BoldOblique.ttf
/usr/share/fonts/dejavu/DejaVuSansCondensed-Oblique.ttf
/usr/share/fonts/dejavu/DejaVuSans-BoldOblique.ttf
/usr/share/fonts/dejavu/DejaVuSans-Oblique.ttf
/usr/share/fonts/dejavu/DejaVuSansMono-BoldOblique.ttf
/usr/share/fonts/dejavu/DejaVuSansMono-Oblique.ttf
/usr/share/fonts/liberation/LiberationSans-BoldItalic.ttf
/usr/share/fonts/liberation/LiberationSans-Italic.ttf
```

Ebben a példában a reguláris kifejezés olyan elágazásokat tartalmaz (*kiterjesztett* stílusban írva), amelyek csak a `/usr/share/fonts` mappastruktúrában található betűtípusfájlokat listázzák. A kiterjesztett reguláris kifejezések alapértelmezés szerint nem támogatottak, de a `find` lehetővé teszi, hogy a `-regextype posix-extended` vagy `-regextype egrep` kapcsolóval engedélyezzük őket. A `find` alapértelmezett RE szabványa a *findutils-default*, ami gyakorlatilag egy alapvető reguláris kifejezés klón.

Gyakran szükséges, hogy a program kimenetét átadjuk a `less` parancsnak, ha az nem fér ki a képernyőre. A `less` parancs a bemenetet oldalakra bontja, egy-egy képernyőnyi részre, így a felhasználó könnyen navigálhat a szövegben fel és le. Ezenkívül a `less` lehetővé teszi a felhasználó számára a reguláris kifejezéseken alapuló keresést is. Ez a funkció különösen fontos, mivel a `less` az alapértelmezett lapozó, amelyet sok mindennapi feladathoz használnak, például a logfájlok ellenőrzéséhez vagy a kézikönyvek oldalainak megtekintéséhez. Egy kézikönyvoldal olvasásakor például a `/` billentyű megnyomásával egy keresési prompt nyílik meg. Ez egy tipikus forgatókönyv, amelyben a reguláris kifejezések hasznosak, mivel a kézikönyvoldal általános elrendezésében a parancsopciók közvetlenül az oldal margója után szerepelnek. Azonban ugyanaz az opció többször is előfordulhat a szövegben, ami a szó szerinti keresést kivitelezhetetlenné teszi. Ettől függetlenül a `^\\[*-o` — vagy még egyszerűbben: `^ *-o` — beírása a keresési promptban az Enter megnyomása után azonnal az `-o` szakasz opciójára ugrik (ha létezik), így gyorsabban meg lehet keresni egy opció leírását.

Gyakorló feladatok

1. Milyen kiterjesztett reguláris kifejezésnek felelne meg bármelyik e-mail cím, például az `info@example.org`?

2. Milyen kiterjesztett reguláris kifejezésnek felelnének meg a szabványos formátumú IPv4 címek, mint például a `192.168.15.1`?

3. Hogyan lehet a `grep` parancsot használni az `/etc/services` fájl tartalmának listázására, az összes kommentet (a `#-val kezdődő sorokat) elhagyva?

4. A `domains.txt` fájl tartalmazza a tartománynevek listáját, soronként egyet-egyet. Hogyan lehetne az `egrep` parancs segítségével csak a `.org` vagy csak a `.com` tartományokat listázni?

Gondolkodtató feladatok

1. Az aktuális mappában hogyan használhatnánk a `find` parancsban egy kiterjesztett reguláris kifejezést az összes olyan fájl keresésére, amely nem tartalmaz szabványos fájlvégződéseket (például a `.txt` vagy `.c` végződésű fájlokat)?

2. A `less` parancs az alapértelmezett lapozó a hosszú szöveges fájlok megjelenítéséhez a shell környezetben. A `/` beírásával egy reguláris kifejezést adhatunk meg a keresési promptban, hogy az első megfelelő találatra ugorjon. Ahhoz, hogy a dokumentum aktuális pozíciójában maradjon, és csak a megfelelő találatokat emelje ki, milyen billentyűkombinációt kell beírni a keresési promptba?

3. A `less`-ben hogyan lehetne szűrni a kimenetet, hogy csak a reguláris kifejezésnek megfelelő sorok jelenjenek meg?

Összefoglalás

Ez a lecke a reguláris kifejezések általános Linux-támogatásával foglalkozott, amely egy széles körben használt szabvány, amelynek mintaillesztési képességeit a legtöbb szöveghez kapcsolódó program támogatja. A lecke a következő lépéseken ment keresztül:

- Mi a reguláris kifejezés.
- A reguláris kifejezések fő összetevői.
- A különbségek a reguláris és a kiterjesztett reguláris kifejezések között.
- Hogyan végezhetünk egyszerű szöveges- és fájlkereséseket a reguláris kifejezések segítségével.

Válaszok a gyakorló feladatokra

1. Milyen kiterjesztett reguláris kifejezésnek felelne meg bármelyik e-mail cím, például az `info@example.org`?

```
egrep "\S+@\S+\.\S+"
```

2. Milyen kiterjesztett reguláris kifejezésnek felelnének meg a szabványos formátumú IPv4 címek, mint például a `192.168.15.1`?

```
egrep "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}"
```

3. Hogyan lehet a `grep` parancsot használni az `/etc/services` fájl tartalmának listázására, az összes kommentet (a `#-val kezdődő sorokat) elhagyva?

```
grep -v ^# /etc/services
```

4. A `domains.txt` fájl tartalmazza a tartománynevek listáját, soronként egyet-egyet. Hogyan lehetne az `egrep` parancs segítségével csak a `.org` vagy csak a `.com` tartományokat listázni?

```
egrep ".org$|.com$" domains.txt
```

Válaszok a gondolkodtató feladatokra

1. Az aktuális mappában hogyan használhatnánk a `find` parancsban egy kiterjesztett reguláris kifejezést az összes olyan fájl keresésére, amely nem tartalmaz szabványos fájlvégződéseket (például a `.txt` vagy `.c` végződésű fájlokat)?

   ```
   find . -type f -regextype egrep -not -regex '.*\.[[:alnum:]]{1,}$'
   ```

2. A `less` parancs az alapértelmezett lapozó a hosszú szöveges fájlok megjelenítéséhez a shell környezetben. A `/` beírásával egy reguláris kifejezést adhatunk meg a keresési promptban, hogy az első megfelelő találatra ugorjon. Ahhoz, hogy a dokumentum aktuális pozíciójában maradjon, és csak a megfelelő találatokat emelje ki, milyen billentyűkombinációt kell beírni a keresési promptba?

   A `Ctrl` + `K` lenyomásával, mielőtt beírnánk a keresési kifejezést.

3. A `less`-ben hogyan lehetne szűrni a kimenetet, hogy csak a reguláris kifejezésnek megfelelő sorok jelenjenek meg?

   A `&` lenyomásával, majd a keresési kifejezés megadásával.

Linux Professional Institute

103.7 Lecke 2

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	103 GNU és Unix parancsok
Fejezet:	103.7 Keresés szövegfájlokban reguláris kifejezések segítségével
Lecke:	2/2

Bevezetés

Az adatok csővezetékes parancsok láncolatán keresztül történő áramoltatása lehetővé teszi a reguláris kifejezéseken alapuló összetett szűrők alkalmazását. A reguláris kifejezések nemcsak a rendszergazdai munka során, hanem az *adatbányászatban* és a kapcsolódó területeken is fontos technikát jelentenek. Két parancs kifejezetten alkalmas fájlok és szöveges adatok szabályos kifejezésekkel történő manipulálására: a `grep` és a `sed`. A `grep` egy mintakereső, a `sed` pedig egy folyamszerkesztő. Önmagukban is hasznosak, de más folyamatokkal együtt dolgozva kiemelkedőek.

A mintakereső: grep

A `grep` egyik legáltalánosabb felhasználási módja a hosszú fájlok vizsgálatának megkönnyítése, a reguláris kifejezést az egyes sorokra alkalmazott szűrőként használva. Használható arra, hogy csak a bizonyos kifejezéssel kezdődő sorokat jelenítse meg. Például a `grep` használható a kernelmodulok konfigurációs fájljának vizsgálatára, csak az opciós sorokat listázva:

```
$ grep '^options' /etc/modprobe.d/alsa-base.conf
options snd-pcsp index=-2
options snd-usb-audio index=-2
options bt87x index=-2
options cx88_alsa index=-2
options snd-atiixp-modem index=-2
options snd-intel8x0m index=-2
options snd-via82xx-modem index=-2
```

A pipe `|` karakterrel egy parancs kimenete átirányítható közvetlenül a `grep` bemenetére. A következő példa egy zárójeles kifejezést használ az `fdisk -l` kimenetéből azon sorok kiválasztására, amik `Disk /dev/sda`-val vagy `Disk /dev/sdb`-val kezdődnek:

```
# fdisk -l | grep '^Disk /dev/sd[ab]'
Disk /dev/sda: 320.1 GB, 320072933376 bytes, 625142448 sectors
Disk /dev/sdb: 7998 MB, 7998537728 bytes, 15622144 sectors
```

Az egyezéseket tartalmazó sorok puszta kiválasztása nem biztos, hogy megfelelő egy adott feladathoz, így a `grep` viselkedését a kapcsolókkal kell módosítani. Például a `-c` vagy `--count` kapcsoló arra utasítja a `grep`-et, hogy írja ki, hány sorban volt találat:

```
# fdisk -l | grep '^Disk /dev/sd[ab]' -c
2
```

A kapcsolót a reguláris kifejezés előtt vagy után lehet elhelyezni. Más fontos `grep` kapcsolók a következők:

`-c` or `--count`

Ahelyett, hogy megjelenítené a keresési eredményeket, csak az összesített számot jeleníti meg, hogy hányszor fordul elő az egyezés bármely adott fájlban.

`-i` or `--ignore-case`

A kis- és nagybetű érzékeny keresés bekapcsolása.

`-f FILE` or `--file=FILE`

Adjuk meg a használni kívánt reguláris kifejezést tartalmazó fájlt.

`-n` or `--line-number`

A sor számának megjelenítése.

`-v` or `--invert-match`

Minden sor kijelölése, kivéve az egyezéseket tartalmazó sorokat.

`-H` or `--with-filename`

A sort tartalmazó fájl nevének kiírása.

`-z` or `--null-data`

Ahelyett, hogy a `grep` a bemeneti és kimeneti adatfolyamokat külön sorokként kezelné (alapértelmezés szerint a *newline* használatával), a bemenetet vagy a kimenetet sorok sorozatának tekinti. Ha a `find` parancs kimenetét a `-print0` kapcsolót használva kombináljuk a `grep` paranccsal, akkor a `-z` vagy a `--null-data` kapcsolót kell használni a folyam azonos módon történő feldolgozásához.

Bár alapértelmezés szerint több fájl elérési útvonalának megadásakor aktiválva van, a `-H` kapcsoló nem aktiválódik egyetlen fájl esetén. Ez speciális helyzetekben kritikus lehet, például amikor a `grep`-et közvetlenül a `find` hívja meg:

```
$ find /usr/share/doc -type f -exec grep -i '3d modeling' "{}" \; | cut -c -100
artistic aspects of 3D modeling. Thus this might be the application you are
This major approach of 3D modeling has not been supported
oce is a C++ 3D modeling library. It can be used to develop CAD/CAM softwares, for instance
[FreeCad
```

Ebben a példában a `find` felsorol minden fájlt a `/usr/share/doc` alatt, majd átadja mindegyiket a `grep`-nek, amely viszont a `3d modeling` szóra keres a fájlban, nem érzékelve a nagy- és kisbetűket. A pipe a `cut`-hoz csak azért van, hogy a kimenet hosszát 100 oszlopra korlátozzuk. Megjegyzendő azonban, hogy nem lehet tudni, hogy a sorok melyik fájlból származnak. Ez a probléma megoldható a `-H` hozzáadásával a `grep`-hez:

```
$ find /usr/share/doc -type f -exec grep -i -H '3d modeling' "{}" \; | cut -c -100
/usr/share/doc/openscad/README.md:artistic aspects of 3D modeling. Thus this might be the
applicatio
/usr/share/doc/opencsg/doc/publications.html:This major approach of 3D modeling has not been
support
```

Most már lehetséges azonosítani azokat a fájlokat, ahol az egyezések vannak. A lista még informatívabbá tétele érdekében a találatokat tartalmazó sorokhoz elő- és utósorok adhatók:

```
$ find /usr/share/doc -type f -exec grep -i -H -1 '3d modeling' "{}" \; | cut -c -100
/usr/share/doc/openscad/README.md-application Blender), OpenSCAD focuses on the CAD aspects
```

```
rather t
/usr/share/doc/openscad/README.md:artistic aspects of 3D modeling. Thus this might be the
applicatio
/usr/share/doc/openscad/README.md-looking for when you are planning to create 3D models of
machine p
/usr/share/doc/opencsg/doc/publications.html-3D graphics library for Constructive Solid
Geometry (CS
/usr/share/doc/opencsg/doc/publications.html:This major approach of 3D modeling has not been
support
/usr/share/doc/opencsg/doc/publications.html-by real-time computer graphics until recently.
```

Az `-1` opció arra utasítja a `grep`-t, hogy a talált egyezés elé és után is beszúrjon egy sort. Ezeket az extra sorokat *context lines*-nak (kontextus sor) nevezzük, és a kimeneten a fájlnév után egy mínusz jel jelzi őket. Ugyanezt az eredményt kaphatjuk a `-C 1` vagy a `--context=1` kapcsolóval, és más mennyiséget is megadhatunk.

A grep-nek két kiegészítő programja van: az `egrep` és az `fgrep`. Az `egrep` program a `grep -E` paranccsal egyenértékű, amely az alapvető reguláris kifejezéseken kívül más extra funkciókat is tartalmaz. Például az `egrep` programmal lehetőség van a kiterjesztett reguláris kifejezések olyan funkcióinak használatára, mint az elágazás:

```
$ find /usr/share/doc -type f -exec egrep -i -H -1 '3d (modeling|printing)' "{}" \; | cut -c
-100
/usr/share/doc/openscad/README.md-application Blender), OpenSCAD focuses on the CAD aspects
rather t
/usr/share/doc/openscad/README.md:artistic aspects of 3D modeling. Thus this might be the
applicatio
/usr/share/doc/openscad/README.md-looking for when you are planning to create 3D models of
machine p
/usr/share/doc/openscad/RELEASE_NOTES.md-* Support for using 3D-Mouse / Joystick / Gamepad
input dev
/usr/share/doc/openscad/RELEASE_NOTES.md:* 3D Printing support: Purchase from a print
service partne
/usr/share/doc/openscad/RELEASE_NOTES.md-* New export file formats: SVG, 3MF, AMF
/usr/share/doc/opencsg/doc/publications.html-3D graphics library for Constructive Solid
Geometry (CS
/usr/share/doc/opencsg/doc/publications.html:This major approach of 3D modeling has not been
support
/usr/share/doc/opencsg/doc/publications.html-by real-time computer graphics until recently.
```

Ebben a példában a `3D modeling` vagy a `3D printing` felel meg a kifejezésnek, a nagy- és kisbetűket nem figyelembe véve. Ha a szövegfolyamnak csak azokat a részeit szeretnénk

megjeleníteni, amelyek megfelelnek az `egrep` által használt kifejezésnek, használjuk az `-o` kapcsolót.

Az `fgrep` program a `grep -F` programmal egyenértékű, azaz nem elemzi a reguláris kifejezéseket. Hasznos egyszerű kereséseknél, ahol a cél egy szó szerinti kifejezésre való illesztés. Ezért az olyan speciális karaktereket, mint a dollárjel és a pont, szó szerint veszi, nem pedig a reguláris kifejezés szerinti jelentésben.

A folyamszerkesztő: sed

A `sed` program célja a szöveges adatok nem interaktív módon történő módosítása. Ez azt jelenti, hogy minden szerkesztés előre meghatározott utasítások alapján történik, nem pedig a képernyőn megjelenő szövegbe való önkényes közvetlen beírás útján. Modern kifejezéssel élve a `sed` egy sablonfeldolgozóként értelmezhető: egy szöveget bemenetként kapva, egyéni tartalmat helyez el előre meghatározott helyekre, vagy ha talál egyezést egy reguláris kifejezésre.

A sed, ahogy a neve is mutatja, jól alkalmazható a csővezetékeken keresztül áramló szövegek feldolgozására. Alapvető szintaxisa a `sed -f SCRIPT`, ha a szerkesztési utasításokat a `SCRIPT` fájlban tároljuk, vagy a `sed -e COMMANDS` a `COMMANDS` közvetlenül a parancssorból történő végrehajtásához. Ha sem az `-f`, sem az `-e` nincs megadva, a `sed` az első nem opciós paramétert használja szkriptfájlként. Az is lehetséges, hogy egy fájlt használjunk bemenetként, ha annak elérési útját adjuk meg a `sed` argumentumaként.

A `sed` utasítások egyetlen karakterből állnak, amelyet egy cím előzhet meg, vagy egy vagy több kapcsoló követhet, és minden egyes sorra egyszerre alkalmazandók. A címek lehetnek egyetlen sorszám, reguláris kifejezés vagy sortartomány. Például egy szövegfolyam első sora törölhető az `1d` paranccsal, ahol az `1` azt a sort adja meg, amelyre az `d` törlési parancsot alkalmazzuk. A `sed` használatának tisztázásához vegyük a `factor `seq 12`` parancs kimenetét, amely az 1-től 12-ig terjedő számok prímtényezőit adja vissza:

```
$ factor `seq 12`
1:
2: 2
3: 3
4: 2 2
5: 5
6: 2 3
7: 7
8: 2 2 2
9: 3 3
10: 2 5
```

```
11: 11
12: 2 2 3
```

Az első sor törlése a `sed`-del az `1d`-vel történik.:

```
$ factor `seq 12` | sed 1d
2: 2
3: 3
4: 2 2
5: 5
6: 2 3
7: 7
8: 2 2 2
9: 3 3
10: 2 5
11: 11
12: 2 2 3
```

A sorok tartományát vesszővel elválasztva lehet megadni:

```
$ factor `seq 12` | sed 1,7d
8: 2 2 2
9: 3 3
10: 2 5
11: 11
12: 2 2 3
```

Egynél több utasítás is használható ugyanabban a végrehajtásban, pontosvesszővel elválasztva. Ebben az esetben azonban fontos, hogy zárójelbe zárjuk őket, hogy a pontosvesszőt a shell ne értelmezze:

```
$ factor `seq 12` | sed "1,7d;11d"
8: 2 2 2
9: 3 3
10: 2 5
12: 2 2 3
```

Ebben a példában két törlési utasítást hajtottunk végre, először az 1-től 7-ig terjedő sorokban, majd a 11. sorban. A cím lehet reguláris kifejezés is, így csak az egyező sorokat érinti az utasítás:

```
$ factor `seq 12` | sed "1d;/:.*2.*/d"
3: 3
5: 5
7: 7
9: 3 3
11: 11
```

A `:.*2.*` reguláris kifejezés a 2-es szám bármelyik előfordulására illeszkedik, bárhol a kettőspont után, a 2-es számmal rendelkező sorok törlését eredményezve. A `sed` használatával minden, ami a perjelek (`/`) közé kerül, reguláris kifejezésnek minősül, és alapértelmezés szerint minden alapvető RE támogatott. Például a `sed -e "/^#/d" /etc/services` az `/etc/services` fájl tartalmát mutatja meg a `#`-vel kezdődő sorok (kommentek) nélkül.

Az `d` törlési utasítás csak egy a sok szerkesztési utasítás közül, amelyet a `sed` biztosít. Ahelyett, hogy törölne egy sort, a `sed` egy adott szövegre is kicserélheti azt:

```
$ factor `seq 12` | sed "1d;/:.*2.*/c REMOVED"
REMOVED
3: 3
REMOVED
5: 5
REMOVED
7: 7
REMOVED
9: 3 3
REMOVED
11: 11
REMOVED
```

A `c REMOVED` utasítás egyszerűen a `REMOVED` szöveggel helyettesíti a sort. A példa esetében a `c REMOVED` utasítás minden olyan sort érint, amelynek részláncai megfelelnek a `:.*2.*` reguláris kifejezésnek. Az `a TEXT` utasítás a `TEXT` által jelzett szöveget az egyezéssel rendelkező sor utáni új sorba másolja. Az `r FILE` utasítás ugyanezt teszi, de a `FILE` által megjelölt fájl tartalmát másolja. A `w` utasítás az `r` utasítás ellenkezőjét teszi, azaz a sort a megadott fájlhoz csatolja.

A messze leggyakrabban használt `sed` utasítás a `s/FIND/REPLACE/`, amely arra szolgál, hogy a `FIND` reguláris kifejezéssel való egyezést a `REPLACE` által jelzett szöveggel helyettesítse. Például a `s/hda/sda/` utasítás a szó szerinti RE `hda` szócikkre illeszkedő részláncot `sda`-val helyettesíti. Csak a sorban talált első egyezés lesz kicserélve, kivéve, ha az utasítás után a `g` jelzőt helyezzük el, mint a `s/hda/sda/g`-ban.

Egy reálisabb esettanulmány segít a `sed` jellemzőinek illusztrálásában. Tegyük fel, hogy egy orvosi klinika szöveges üzeneteket szeretne küldeni ügyfeleinek, hogy emlékeztesse őket a következő napra tervezett időpontokra. Egy tipikus megvalósítási forgatókönyv egy professzionális azonnali üzenetküldő szolgáltatásra támaszkodik, amely API-t biztosít az üzenetek kézbesítéséért felelős rendszer eléréséhez. Ezek az üzenetek általában ugyanabból a rendszerből származnak, amely az ügyfelek időpontját ellenőrző alkalmazást futtatja, és amelyeket egy adott napszak vagy más esemény indít el. Ebben a hipotetikus helyzetben az alkalmazás létrehozhatna egy `appointments.csv` nevű fájlt, amely táblázatos adatokat tartalmazna a következő napra vonatkozó összes időponttal, majd a `sed` segítségével megjeleníthetné a szöveges üzeneteket a `template.txt` nevű sablonfájlból. A CSV-fájl az adatbázis-lekérdezésekből származó adatok exportálásának szabványos módja, így a minta-időpontokat a következőképpen adhatjuk meg:

```
$ cat appointments.csv
"NAME","TIME","PHONE"
"Carol","11am","55557777"
"Dave","2pm","33334444"
```

Az első sor tartalmazza az egyes oszlopok címkéit, amelyeket a minta sablonfájlban lévő címkékkel való egyezéshez használunk:

```
$ cat template.txt
Helló <NAME>, ne felejtse el az időpontját holnap <TIME>-kor.
```

A kisebb, mint `<` és nagyobb, mint `>` jelek csak azért kerültek a címkék köré, hogy segítsék a címkék azonosítását. A következő Bash szkript elemzi az összes beállított találkozót a `template.txt` mint üzenetsablon használatával:

```
#! /bin/bash

TEMPLATE=`cat template.txt`
TAGS=(`sed -ne '1s/^"//;1s/","/\n/g;1s/"$//p' appointments.csv`)
mapfile -t -s 1 ROWS < appointments.csv
for (( r = 0; r < ${#ROWS[*]}; r++ ))
do
  MSG=$TEMPLATE
  VALS=(`sed -e 's/^"//;s/","/\n/g;s/"$//' <<<${ROWS[$r]}`)
  for (( c = 0; c < ${#TAGS[*]}; c++ ))
  do
    MSG=`sed -e "s/<${TAGS[$c]}>/${VALS[$c]}/g" <<<"$MSG"`
  done
```

```
  echo curl --data message=\"$MSG\" --data phone=\"${VALS[2]}\" https://mysmsprovider/api
done
```

Egy valódi, gyártásra szánt szkript a hitelesítést, a hibaellenőrzést és a naplózást is kezelné, de ez a példa csak az alapfunkciókkal rendelkezik. A `sed` által végrehajtott első utasítások csak az első sorra vonatkoznak — az `1` címre az `1s/^"//;1s/","/\n/g;1s/"$//p`-ben --, hogy eltávolítsa a kezdő és a záró idézőjeleket — `1s/^"//` és `1s/"$//` — és a mezőelválasztó karaktereket újsorral helyettesítse: `1s/","/\n/g`. Az oszlopnevek betöltéséhez csak az első sorra van szükség, így a nem megfelelő sorokat a `-n` kapcsoló elnyomja, így az utolsó `sed` parancs után a `p` flaget kell elhelyezni a megfelelő sor kiírásához. A címkéket ezután a `TAGS` változóban tároljuk, mint egy Bash tömböt. Egy másik Bash tömbváltozót hoz létre a `mapfile` parancs az időpontokat tartalmazó sorok tárolására a `ROWS` tömbváltozóban.

A `for` ciklus a `ROWS`-ban található minden egyes időpontot tartalmazó sor feldolgozására szolgál. Ezután az időpontban lévő idézőjeleket és elválasztójeleket — az időpont a `${ROWS[$r]}` változóban van, amelyet *here string*-ként használunk — a `sed` helyettesíti, hasonlóan a címkék betöltésére használt parancsokhoz. Az időpont elválasztott értékeit ezután a `VALS` tömbváltozóban tároljuk, ahol a tömb 0, 1 és 2 indexei a `NAME`, `TIME` és `PHONE` értékeknek felelnek meg.

Végül egy egymásba ágyazott `for` ciklus végigmegy a `TAGS` tömbön, és minden egyes, a sablonban található taget kicserél a megfelelő értékkel a `VALS`-ban. Az `MSG` változó a renderelt sablon másolatát tartalmazza, amelyet a `s/<${TAGS[$c]}>/${VALS[$c]}/g` helyettesítési parancs frissít a `TAGS` minden egyes cikluson való áthaladásakor.

Ez egy ilyen megjelenített üzenetet eredményez: "Helló Carol, ne felejtse el az időpontját holnap 11-kor."` A megjelenített üzenet ezután elküldhető paraméterként egy HTTP-kérésen keresztül a `curl` segítségével, e-mailként vagy bármilyen más hasonló módszerrel.

A grep és a sed kombinálása

A `grep` és a `sed` parancsok együttesen is használhatók, ha összetettebb szövegbányászati eljárásokra van szükség. Rendszergazdaként például megvizsgálhatjuk az összes bejelentkezési kísérletet egy szerverre. A `/var/log/wtmp` fájl az összes bejelentkezést és kijelentkezést rögzíti, míg a `/var/log/btmp` fájl a sikertelen bejelentkezési kísérleteket. Ezek bináris formátumban íródnak, és a `last`, illetve `lastb` parancsokkal olvashatók.

A `lastb` kimenete nem csak a rossz bejelentkezési kísérletben használt felhasználónevet mutatja, hanem az IP-címet is:

```
# lastb -d -a -n 10 --time-format notime
user     ssh:notty       (00:00)     81.161.63.251
```

```
nrostagn ssh:notty       (00:00)     vmd60532.contaboserver.net
pi       ssh:notty       (00:00)     132.red-88-20-39.staticip.rima-tde.net
pi       ssh:notty       (00:00)     132.red-88-20-39.staticip.rima-tde.net
pi       ssh:notty       (00:00)     46.6.11.56
pi       ssh:notty       (00:00)     46.6.11.56
nps      ssh:notty       (00:00)     vmd60532.contaboserver.net
narmadan ssh:notty       (00:00)     vmd60532.contaboserver.net
nominati ssh:notty       (00:00)     vmd60532.contaboserver.net
nominati ssh:notty       (00:00)     vmd60532.contaboserver.net
```

A `d` kapcsoló az IP-címet a megfelelő hosztnévre fordítja. A hosztnév nyomokat adhat az ISP-ről vagy a rossz bejelentkezési kísérletekhez használt tárhelyszolgáltatóról. Az `-a` opció a hosztnevet az utolsó oszlopba helyezi, ami megkönnyíti a még alkalmazandó szűrést. A `--time-format notime` opció elnyomja a bejelentkezési kísérlet időpontját. A `lastb` parancs befejezése eltarthat egy ideig, ha túl sok rossz bejelentkezési kísérlet történt, ezért a kimenetet tíz bejegyzésre korlátoztuk a `-n 10` kapcsolóval.

Nem minden távoli IP-hez tartozik hosztnév, így a fordított DNS nem vonatkozik rájuk, és figyelmen kívül lehet őket hagyni. Bár írhatnánk egy olyan reguláris kifejezést, amely megfelel a sor végén lévő hosztnév elvárt formátumának, valószínűleg egyszerűbb olyan reguláris kifejezést írni, amely vagy az ábécé egy betűjével, vagy a sor végén lévő egyetlen számjeggyel egyezik meg. A következő példa azt mutatja, hogy a `grep` parancs hogyan veszi a standard bemeneten lévő listát, és hogyan távolítja el a hosztnév nélküli sorokat:

```
# lastb -d -a --time-format notime | grep -v '[0-9]$' | head -n 10
nvidia   ssh:notty       (00:00)     vmd60532.contaboserver.net
n_tonson ssh:notty       (00:00)     vmd60532.contaboserver.net
nrostagn ssh:notty       (00:00)     vmd60532.contaboserver.net
pi       ssh:notty       (00:00)     132.red-88-20-39.staticip.rima-tde.net
pi       ssh:notty       (00:00)     132.red-88-20-39.staticip.rima-tde.net
nps      ssh:notty       (00:00)     vmd60532.contaboserver.net
narmadan ssh:notty       (00:00)     vmd60532.contaboserver.net
nominati ssh:notty       (00:00)     vmd60532.contaboserver.net
nominati ssh:notty       (00:00)     vmd60532.contaboserver.net
nominati ssh:notty       (00:00)     vmd60532.contaboserver.net
```

A `grep` parancs `-v` opciója csak azokat a sorokat mutatja meg, amelyek nem egyeznek a megadott reguláris kifejezéssel. A számmal végződő sorokra (pl. `[0-9]$`) illeszkedő reguláris kifejezés csak a hosztnév nélküli bejegyzéseket fogja rögzíteni. Ezért a `grep -v '[0-9]$'` csak a hosztnévvel végződő sorokat fogja megmutatni.

A kimenet még tovább szűrhető, ha csak a domainnevet tartjuk meg, és minden sorból eltávolítjuk a többi részt. A `sed` parancs ezt egy helyettesítési paranccsal teszi meg, amely az egész sort a benne lévő domainnévre való visszahivatkozással helyettesíti:

```
# lastb -d -a --time-format notime | grep -v '[0-9]$' | sed -e 's/.* \(.*\)$/\1/' | head -n
10
vmd60532.contaboserver.net
vmd60532.contaboserver.net
vmd60532.contaboserver.net
132.red-88-20-39.staticip.rima-tde.net
132.red-88-20-39.staticip.rima-tde.net
vmd60532.contaboserver.net
vmd60532.contaboserver.net
vmd60532.contaboserver.net
vmd60532.contaboserver.net
vmd60532.contaboserver.net
```

A feloldott zárójel `.* \(.*\)$` azt mondja a `sed`-nek, hogy a sornak ezt a részét jegyezze meg, vagyis az utolsó szóköz és a sor vége közötti részt. A példában erre a részre a `\1`-el hivatkozunk, és a teljes sor helyettesítésére használjuk.

Nyilvánvaló, hogy a legtöbb távoli állomás többször is megpróbál bejelentkezni, így ugyanaz a tartománynév ismétlődik. Az ismétlődő bejegyzések elnyomása érdekében először rendezni kell őket (a `sort` paranccsal), majd átadni a `uniq` parancsnak:

```
# lastb -d -a --time-format notime | grep -v '[0-9]$' | sed -e 's/.* \(.*\)$/\1/' | sort |
uniq | head -n 10
116-25-254-113-on-nets.com
132.red-88-20-39.staticip.rima-tde.net
145-40-33-205.power-speed.at
tor.laquadrature.net
tor.momx.site
ua-83-226-233-154.bbcust.telenor.se
vmd38161.contaboserver.net
vmd60532.contaboserver.net
vmi488063.contaboserver.net
vmi515749.contaboserver.net
```

Ez azt mutatja, hogy a különböző parancsok hogyan kombinálhatók a kívánt eredmény elérése érdekében. A hosztnévlista ezután felhasználható blokkoló tűzfalszabályok írására vagy más intézkedések megtételére a szerver biztonságának megerősítése érdekében.

Gyakorló feladatok

1. A `last` parancs megjeleníti az utoljára bejelentkezett felhasználók listáját, beleértve az IP-jüket is. Hogyan lehetne az `egrep` paranccsal szűrni a `last` kimenetét, hogy csak az IPv4-címek előfordulását mutassa, és a megfelelő sorban lévő további információkat elvesse?

2. Milyen kapcsolót kell megadni a `grep` parancsnak ahhoz, hogy a `-print0` opcióval végrehajtott `find` parancs által generált kimenetet helyesen szűrje?

3. Az `uptime -s` parancs megmutatja a rendszer legutóbbi bekapcsolásának dátumát, például: `2019-08-05 20:13:22`. Mi lesz az `uptime -s | sed -e 's/(.*) (.*)/\1/'` parancs eredménye?

4. Milyen kapcsolót kell megadni a `grep`-nek, hogy a megfelelő sorokat megszámolja ahelyett, hogy megjelenítené őket?

Gondolkodtató feladatok

1. Egy HTML fájl alapvető szerkezete a `html`, `head` és `body` elemekkel kezdődik, például:

```
<html>
<head>
  <title>Hírportál</title>
</head>
<body>
  <h1>Főcím</h1>
  <p>Érdekes információ.</p>
</body>
</html>
```

   Hogyan lehet a címeket a `sed`-ben úgy használni, hogy csak a `body` elemet és annak tartalmát jelenítse meg?

2. Milyen `sed` kifejezés távolítja el az összes taget egy HTML dokumentumból, csak a megjelenítendő szöveget megtartva?

3. Az `.ovpn` kiterjesztésű fájlok nagyon népszerűek a VPN kliensek konfigurálásánál, mivel nemcsak a beállításokat, hanem a kliens kulcsainak és tanúsítványainak tartalmát is tartalmazzák. Ezek a kulcsok és tanúsítványok eredetileg különálló fájlokban vannak, ezért azokat be kell másolni az `.ovpn` fájlba. Adott a következő részlet egy `.ovpn` sablonból:

```
client
dev tun
remote 192.168.1.155 1194
<ca>
ca.crt
</ca>
<cert>
client.crt
</cert>
<key>
client.key
</key>
<tls-auth>
ta.key
```

```
</tls-auth>
```

Feltételezve, hogy a `ca.crt`, `client.crt`, `client.key` és `ta.key` fájlok az aktuális mappában vannak, hogyan módosítaná a sablon konfigurációját a `sed`, hogy az egyes fájlneveket a tartalmukkal helyettesítse?

Összefoglalás

Ez a lecke a reguláris kifejezésekkel kapcsolatos két legfontosabb Linux-paranccsal foglalkozott: `grep` és `sed`. A szkriptek és az összetett parancsok a szövegszűrési és elemzési feladatok széles skálájához támaszkodnak a `grep`-re és a `sed`-re. A lecke a következő témákat járta körül:

- Hogyan használjuk a `grep`-et és változatait, mint például az `egrep`-et és az `fgrep`-et.
- Hogyan használjuk a `sed`-et és annak belső utasításait a szöveg manipulálására.
- Példák a reguláris kifejezések használatára `grep` és `sed` esetén.

Válaszok a gyakorló feladatokra

1. A `last` parancs megjeleníti az utoljára bejelentkezett felhasználók listáját, beleértve az IP-jüket is. Hogyan lehetne az `egrep` paranccsal szűrni a `last` kimenetét, hogy csak az IPv4-címek előfordulását mutassa, és a megfelelő sorban lévő további információkat elvesse?

   ```
   last -i | egrep -o '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}'
   ```

2. Milyen kapcsolót kell megadni a `grep` parancsnak ahhoz, hogy a `-print0` opcióval végrehajtott `find` parancs által generált kimenetet helyesen szűrje?

   A `-z` vagy `--null-data` kapcsolót, mint például `find . -print0 | grep -z expression`.

3. Az `uptime -s` parancs megmutatja a rendszer legutóbbi bekapcsolásának dátumát, például: `2019-08-05 20:13:22`. Mi lesz az `uptime -s | sed -e 's/(.*) (.*)/\1/'` parancs eredménye?

   Hiba lép fel. Alapértelmezés szerint a zárójeleket fel kell oldani, hogy a `sed`-ben a visszautalásokat használhassuk.

4. Milyen kapcsolót kell megadni a `grep`-nek, hogy a megfelelő sorokat megszámolja ahelyett, hogy megjelenítené őket?

   A `-c` kapcsolót.

Válaszok a gondolkodtató feladatokra

1. Egy HTML fájl alapvető szerkezete a `html`, `head` és `body` elemekkel kezdődik, például:

```
<html>
<head>
  <title>Hírportál</title>
</head>
<body>
  <h1>Főcím</h1>
  <p>Érdekes információ.</p>
</body>
</html>
```

   Hogyan lehet a címeket a `sed`-ben úgy használni, hogy csak a `body` elemet és annak tartalmát jelenítse meg?

   Ahhoz, hogy csak a `body`-t lássuk, a címeknek `/<body>/,/<\/body>/`-nak kell lenniük, mint például `sed -n -e '/<body>/,/<\/body>/p'`. A `-n`kapcsoló van megadva a `sed`-nek, így alapértelmezés szerint nem jelenít meg sorokat, ezért a `sed` kifejezés végén lévő `p` parancs a megfelelő sorok kiírására szolgál.

2. Milyen `sed` kifejezés távolítja el az összes taget egy HTML dokumentumból, csak a megjelenítendő szöveget megtartva?

   A `sed` `s/<[^>]*>//g` kifejezése a `<>` közé zárt tartalmakat üres sztringre cseréli.

3. Az `.ovpn` kiterjesztésű fájlok nagyon népszerűek a VPN kliensek konfigurálásánál, mivel nemcsak a beállításokat, hanem a kliens kulcsainak és tanúsítványainak tartalmát is tartalmazzák. Ezek a kulcsok és tanúsítványok eredetileg különálló fájlokban vannak, ezért azokat be kell másolni az `.ovpn` fájlba. Adott a következő részlet egy `.ovpn` sablonból:

```
client
dev tun
remote 192.168.1.155 1194
<ca>
ca.crt
</ca>
<cert>
client.crt
</cert>
<key>
```

```
client.key
</key>
<tls-auth>
ta.key
</tls-auth>
```

Feltételezve, hogy a `ca.crt`, `client.crt`, `client.key` és `ta.key` fájlok az aktuális mappában vannak, hogyan módosítaná a sablon konfigurációját a `sed`, hogy az egyes fájlneveket a tartalmukkal helyettesítse?

A parancs

```
sed -r -e 's/(^[^.]*)\.(crt|key)$/cat \1.\2/e' < client.template > client.ovpn
```

minden `.crt` vagy `.key` végződésű sort egy olyan fájl tartalmával helyettesít, amelynek neve megegyezik a sor nevével. Az `-r` kapcsoló azt mondja a `sed`-nek, hogy kibővített reguláris kifejezéseket használjon, míg az `e` a kifejezés végén azt, hogy a találatokat a `cat \1.\2` parancs kimenetével helyettesítse. A `\1` és `\2` visszautalások megfelelnek az egyezésben található fájlnévnek és kiterjesztésnek.

103.8 A fájlszerkesztés alapjai

Hivatkozás az LPI célkitűzésre

LPIC-1 v5, Exam 101, Objective 103.8

Súlyozás

3

Kulcsfontosságú ismeretek

- Navigate a document using vi.
- Understand and use vi modes.
- Insert, edit, delete, copy and find text in vi.
- Awareness of Emacs, nano and vim.
- Configure the standard editor.

A használt fájlok, kifejezések és segédprogramok listája

- `vi`
- `/`, `?`
- `h`, `j`, `k`, `l`
- `i`, `o`, `a`
- `d`, `p`, `y`, `dd`, `yy`
- `ZZ`, `:w!`, `:q!`
- `EDITOR`

103.8 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	103 GNU és Unix parancsok
Fejezet:	103.8 A fájlszerkesztés alapjai
Lecke:	1/1

Bevezetés

A legtöbb Linux disztribúcióban a `vi` — a “vizuális” rövidítése — előre telepítve van, és ez a shell környezet standard szerkesztője. A vi egy interaktív szövegszerkesztő, a fájl tartalmát szerkesztés közben megjeleníti a képernyőn. Mint ilyen, lehetővé teszi a felhasználó számára, hogy a dokumentumon belül bárhol mozogjon és módosításokat végezzen. A grafikus asztali vizuális szerkesztőprogramokkal ellentétben azonban a `vi` szerkesztő egy shell alkalmazás, amely minden szerkesztési feladathoz billentyűparancsokat tartalmaz.

A `vi` alternatíváját, a `vim`-et (*vi improved*) néha a `vi` modern helyettesítőjeként használják. A `vim` egyéb fejlesztések mellett támogatja a szintaxisok kiemelését, a többszintű undo/redo műveleteket és több dokumentum szerkesztését. Bár a `vim` több erőforrással rendelkezik, teljesen kompatibilis a `vi`-vel, így a legtöbb feladat esetén megkülönböztethetetlen egymástól a kettő.

A `vi` indításának szabványos módja, hogy paraméterként megadjuk egy fájl elérési útvonalát. Ha közvetlenül egy adott sorra akarunk ugrani, akkor a sorszámát plusz jellel kell közölni, mint például a `vi +9 /etc/fstab` a `/etc/fstab/` megnyitásához, és a kurzor 9. sorra helyezéséhez. Szám nélkül a pluszjel önmagában az utolsó sorra helyezi a kurzort.

A `vi` kezelőfelülete nagyon egyszerű: a terminálablakban rendelkezésre álló összes helyet elfoglalja, hogy a felhasználónak megjelenítsen egy fájlt, amelyet általában parancsargumentumként kap meg. Az egyetlen vizuális támpont a kurzor aktuális pozícióját mutató lábléc és a fájl végét jelző tilde `~`. A `vi`-nak különböző végrehajtási módjai vannak, amelyekben a program viselkedése megváltozik. A leggyakoribbak a következők: *insert mode* (beszúrási mód) és *normal mode* (normál mód).

Insert Mode

Az insert mód egyszerű: a szöveg úgy jelenik meg a képernyőn, ahogyan azt a billentyűzeten beírjuk. Ez az a fajta interakció, amit a legtöbb felhasználó elvár egy szövegszerkesztőtől, de a `vi` először nem így jeleníti meg a dokumentumot. Az insert módba való belépéshez a felhasználónak a normál módban kell végrehajtania egy beszúrási parancsot. A `Esc` billentyűvel befejeződik az insert mód, és visszatér a normál módba, a `vi` alapértelmezett módjába.

NOTE

Ha többet szeretnénk megtudni a többi futtatási módról, nyissuk meg a `vi`-t és írjuk be:

```
:help vim-modes-intro
```

Normal Mode

Alapértelmezés szerint a `vi` normál módban - más néven parancs módban - indul. Ebben az üzemmódban a billentyűzetbillentyűk a navigációs és szövegkezelési feladatokhoz tartozó parancsokhoz vannak rendelve. Ebben az üzemmódban a legtöbb parancs egyedi billentyű. Néhány billentyű és funkciójuk a normál módban a következő:

`0`, **`$`**
: A sor elejére vagy végére ugrás.

`1G`, **`G`**
: A dokumentum elejére vagy végére ugrás.

`(`, **`)`**
: A mondat elejére vagy végére ugrás.

`{`, **`}`**
: A bekezdés elejére vagy végére ugrás.

`w`, `W`

Szó vagy szó és írásjel átugrása.

`h`, `j`, `k`, `l`

Balra, le, fel, jobbra.

`e` or `E`

Az aktuális szó végére ugrás.

`/`, `?`

Előre vagy hátra keresés.

`i`, `I`

Belépés az insert módba az aktuális kurzorpozíció előtt vagy az aktuális sor elején.

`a`, `A`

Belépés az insert módba az aktuális kurzorpozíció után vagy az aktuális sor végén.

`o`, `O`

Új sor hozzáadása és belépés az insert módba a következő sorban vagy az előző sorban.

`s`, `S`

A kurzor alatti karakter vagy az egész sor törlése és belépés az insert módba.

`c`

A kurzor alatti karakter(ek) megváltoztatása.

`r`

A kurzor alatti karakter cseréje.

`x`

A kurzor alatt kiválasztott karakter(ek) törlése.

`v`, `V`

Új kijelölés indítása az aktuális karakterrel vagy az egész sorral.

`y`, `yy`

A karakter(ek) vagy az egész sor másolása (yanks).

`p`, `P`

A kijelölt tartalom beillesztése az aktuális pozíció mögé vagy elé.

`u`

Az utolsó művelet visszavonása.

`Ctrl-R`

Az utolsó művelet megismétlése.

`ZZ`

Bezárás és mentés.

`ZQ`

Bezárás mentés nélkül.

Ha a parancs előtt egy szám áll, a parancs annyiszor kerül végrehajtásra. Például a `3yy` megnyomása az aktuális sort és a következő kettőt másolja, a `d5w` megnyomásával az aktuális szót és a következő 4 szót törli, és így tovább.

A legtöbb szerkesztési feladat több parancs kombinációja. Például a `vey` billentyűsorozatot az aktuális pozíciótól kezdődően az aktuális szó végéig tartó kijelölés másolására használjuk. A parancsismétlés kombinációkban is használható, így a `v3ey` az aktuális pozíciótól kezdődő kijelölést másolná az onnan induló harmadik szó végéig.

A `vi` képes a másolt szöveget regiszterekbe rendezni, lehetővé téve a különböző tartalmak egyidejű megtartását. Egy regisztert egy `"` karakterrel kell megadni, és ha egyszer létrehoztuk, akkor az az aktuális munkamenet végéig megmarad. A `"ly` billentyűsorozat létrehoz egy regisztert, amely az aktuális szekciót tartalmazza, és amely az `l` billentyűvel lesz elérhető. Ezután az `l` regisztert az `"lp` billentyűvel lehet beilleszteni.

Lehetőség van arra is, hogy a szöveg mentén tetszőleges pozíciókban egyéni jelöléseket állítsunk be, megkönnyítve ezzel a gyors ugrást közöttük. A jelek létrehozása az `m` billentyű, majd az aktuális pozíciót megcímző billentyű lenyomásával történik. Ha ez megtörtént, akkor a kurzor a `'` és a kiválasztott billentyű lenyomásával visszatér a megjelölt pozícióba.

Any key sequence can be recorded as a macro for future execution. A macro can be recorded, for example, to surround a selected text in double-quotes. First, a string of text is selected and the key `q` is pressed, followed by a register key to associate the macro with, like `d`. The line `recording @d` will appear in the footer line, indicating that the recording is on. It is assumed that some text is already selected, so the first command is `x` to remove (and automatically copy) the selected text. The key `i` is pressed to insert two double quotes at the current position, then `Esc` returns to normal mode. The last command is `P`, to re-insert the deleted selection just before the last double-quote. Pressing `q` again will end the recording. Now, a macro consisting of key sequence `x`, `i`, `""`, `Esc` and `P` will execute every time keys `@d` are pressed in normal mode, where `d` is the register key associated

with the macro.

A makró azonban csak az aktuális munkamenet alatt lesz elérhető. Ahhoz, hogy a makrók tartósak legyenek, a konfigurációs fájlban kell tárolni őket. Mivel a legtöbb modern disztribúció a *vim*-et használja *vi* kompatibilis szerkesztőként, a felhasználó konfigurációs fájlja a `~/.vimrc`. A `~/.vimrc` állományon belül a `let @d = 'xi""^[P'` a `d` regisztert az idézőjelek között lévő billentyűsorozatra állítja. Ugyanaz a korábban egy makróhoz rendelt regiszter használható a billentyűsorozat beillesztésére.

Colon Commands

A normál mód a `vi` parancsok egy másik csoportját is támogatja: a *kettőspont-parancsokat* (colon commands). A kettőspont-parancsok, mint a neve is mutatja, a normál módban a `:` kettőspont billentyű lenyomása után kerülnek végrehajtásra. A kettőspont-parancsok lehetővé teszik a felhasználó számára a keresést, a mentést, a kilépést, a shell parancsok futtatását, a `vi` beállítások módosítását stb. A normál üzemmódba való visszatéréshez a `:visual` parancsot kell végrehajtani, vagy az Enter billentyűt kell megnyomni minden parancs nélkül. A leggyakoribb kettőspont-parancsok közül néhányat itt jelölünk (a kezdőbetű nem része a parancsnak):

`:s/REGEX/TEXT/g`

A `REGEX` reguláris kifejezés összes előfordulását helyettesíti a `TEXT` kifejezéssel az aktuális sorban. Elfogadja a `sed` parancs szintaxisát, beleértve a címeket is.

`:!`

A következő shell parancs futtatása.

`:quit` or **`:q`**

Kilépés a programból.

`:quit!` or **`:q!`**

Kilépés a programoból mentés nélkül.

`:wq`

Mentés és kilépés.

`:exit` or **`:x`** or **`:e`**

Mentés és kilépés, ha szükséges.

`:visual`

Vissza navigációs módba.

A szabványos `vi` program alkalmas a legtöbb szövegszerkesztési feladat elvégzésére, de bármilyen más, nem grafikus szerkesztőprogram is használható a szöveges fájlok szerkesztésére a shellben.

TIP

A kezdő felhasználóknak nehézséget okozhat a `vi` parancsbillentyűinek egyszerre történő megjegyzése. A `vim`-et alkalmazó disztribúciókban a `vimtutor` parancs is megtalálható, amely a magát a `vim`-et használja, hogy lépésről-lépésre megnyissa a fő tevékenységek útmutatóját. A fájl egy szerkeszthető példány, amely a parancsok gyakorlásához és fokozatos megszokásukhoz használható.

Alternatív szerkesztők

A `vi`-t nem ismerő felhasználóknak nehézséget okozhat az alkalmazkodás, mivel működése nem intuitív. Egyszerűbb alternatíva a GNU nano, egy kis szövegszerkesztő, amely minden alapvető szövegszerkesztési funkciót kínál, mint például visszavonás/ismétlés (undo/redo), szintaxisszínezés, interaktív keresés és helyettesítés, automatikus behúzás, sorszámozás, szótömörítés, fájlzár, biztonsági mentés és nemzetköziesítés támogatás. A `vi`-al ellentétben minden billentyűleütés csak beillesztésre kerül a szerkesztett dokumentumba. A nano-ban a parancsokat a `Ctrl` billentyűvel vagy a Meta billentyűvel adhatjuk meg (rendszertől függően a Meta a `Alt` vagy a `◻`).

`Ctrl-6` vagy `Meta-A`

Új kijelölés indítása. Kijelölés létrehozására a Shift billentyű lenyomásával és a kurzor mozgatásával is lehetőség van.

`Meta-6`

Az aktuális kijelölés másolása.

`Ctrl-K`

Az aktuális kijelölés kivágása.

`Ctrl-U`

A kimásolt tartalom beillesztése.

`Meta-U`

Visszavonás (undo).

`Meta-E`

Ismétlés (redo).

`Ctrl-\`

Szöveg kicserélése a kijelölésnél.

`Ctrl-T`

A dokumentum vagy az aktuális kijelölés helyesírás-ellenőrzési munkamenetének elindítása.

Az Emacs egy másik nagyon népszerű szövegszerkesztő a shellhez. Míg a szöveg beillesztése a `nano` programhoz hasonlóan csak gépeléssel történik, addig a dokumentumban való navigációt billentyűparancsok segítik, mint a `vi` programban. Az Emacs számos olyan funkciót tartalmaz, amely többé teszi, mint egy egyszerű szövegszerkesztőt. Ez egy IDE (*integrated development environment* (integrált fejlesztőkörnyezet)) is, amely képes programok fordítására, futtatására és tesztelésére. Az Emacs beállítható e-mail, hír- vagy RSS-kliensként is, így valódi produktivitási csomaggá válik.

Maga a shell minden alkalommal, amikor szükséges, lefuttat egy alapértelmezett szövegszerkesztőt, általában a `vi`-t. Ez a helyzet például akkor, amikor a `crontab -e` parancsot hajtjuk végre a *cronjobs* szerkesztésére. A Bash a `VISUAL` vagy `EDITOR` munkamenetváltozókat használja a shell alapértelmezett szövegszerkesztőjének kiderítésére. Például a `export EDITOR=nano` parancs a `nano`-t határozza meg alapértelmezett szövegszerkesztőként az aktuális shell munkamenetben. Ahhoz, hogy ez a módosítás a munkamenetek között tartós legyen, a parancsot a `~/.bash_profile` állományba kell felvenni.

Gyakorló feladatok

1. A `vi`-t leginkább konfigurációs fájlok és forráskódok szerkesztőjeként használják, ahol a behúzás segít a szövegrészek azonosításában. Egy kijelölés a `<` megnyomásával balra, a `>` megnyomásával pedig jobbra behúzható. Milyen billentyűket kell megnyomni normál módban ahhoz, hogy az aktuális kijelölést három lépéssel balra behúzza?

2. Egy teljes sor kiválasztható a `V` gomb megnyomásával a `vi` normál üzemmódjában. Azonban a befejező újsor karakter is szerepel a kijelölésben. Milyen billentyűket kell megnyomni normál üzemmódban a kezdő karaktertől az újsor karakterig, de nem beleértve az újsor karaktert?

3. Hogyan kell a `vi`-t a parancssorban meghívni, hogy megnyíljon a `~/.bash_profile` és egyenesen az utolsó sorba ugorjon?

4. Milyen billentyűket kell megnyomni a `vi` normál módjában, hogy a kurzor aktuális pozíciójától a következő pont karakterig töröljük a karaktereket?

Gondolkodtató feladatok

1. A `vim` lehetővé teszi tetszőleges szélességű szövegblokkok kijelölését, nem csak egész sorokat tartalmazó szakaszokét. Normál módban a `Ctrl` + `V` billentyűkombináció megnyomásával a kurzor felfelé, lefelé, balra és jobbra mozgatásával történik a kijelölés. Ezzel a módszerrel hogyan lehetne törölni az aktuális sor első karakterétől kezdődő, a következő nyolc oszlopot és öt sornyi szöveget tartalmazó blokkot?

2. Egy `vi` munkamenet megszakadt egy váratlan áramkimaradás miatt. A fájl újbóli megnyitásakor a `vi` megkérdezi a felhasználót, hogy vissza akarja-e állítani a swap fájlt (a `vi` által készített automatikus másolatot). Mit tegyen a felhasználó a swap-fájl elvetéséhez?

3. Egy `vim` munkamenetben egy sor korábban az `l` regiszterbe lett másolva. Milyen billentyűkombinációval rögzítenénk egy makrót az `a` regiszterben, hogy az `l` regiszterben lévő sort közvetlenül az aktuális sor elé illesszük be?

Összefoglalás

Ez a lecke a Linux shell környezet standard szövegszerkesztőjével foglalkozott: a `vi` szerkesztővel. Bár a `vi` az ismeretlen felhasználó számára félelmetes, olyan tulajdonságokkal rendelkezik, amelyek jó választássá teszik a technikai és nem technikai jellegű szövegszerkesztéshez. A lecke a következő lépéseken ment keresztül:

- `vi` alapvető használat és hasznos funkciók.
- Mi a `vim` — a továbbfejlesztett `vi` — és más alternatív szerkesztők.
- Hogyan határozzuk meg a shell környezet alapértelmezett szövegszerkesztőjét.

A tárgyalt parancsok és eljárások a következők voltak:

- A `vi` szerkesztő és annak továbbfejlesztett változata a `vim`.
- Alapvető szövegszerkesztés a `vi`-ban.
- Alternatív szerkesztők `emacs` és `nano`.

Válaszok a gyakorló feladatokra

1. A `vi`-t leginkább konfigurációs fájlok és forráskódok szerkesztőjeként használják, ahol a behúzás segít a szövegrészek azonosításában. Egy kijelölés a `<` megnyomásával balra, a `>` megnyomásával pedig jobbra behúzható. Milyen billentyűket kell megnyomni normál módban ahhoz, hogy az aktuális kijelölést három lépéssel balra behúzza?

   A `3<` billentyűk jelentik a 3 lépés balra behúzását.

2. Egy teljes sor kiválasztható a `V` gomb megnyomásával a `vi` normál üzemmódjában. Azonban a befejező újsor karakter is szerepel a kijelölésben. Milyen billentyűket kell megnyomni normál üzemmódban a kezdő karaktertől az újsor karakterig, de nem beleértve az újsor karaktert?

   A `0v$h` billentyűk, ahol a `0` (“sor elejére ugrás”), `v` (“karakterkijelölés kezdete”), `$` (“sor végére ugrás”) és `h` (“egy pozíció vissza”).

3. Hogyan kell a `vi`-t a parancssorban meghívni, hogy megnyíljon a `~/.bash_profile` és egyenesen az utolsó sorba ugorjon?

   A `vi + ~/.bash_profile` parancs meg fogja nyitni a fájlt és a kurzort az utolsó sorba helyezi.

4. Milyen billentyűket kell megnyomni a `vi` normál módjában, hogy a kurzor aktuális pozíciójától a következő pont karakterig töröljük a karaktereket?

   A `dt.` billentyűket, ahol a `d` (“törlés kezdése”), `t` (“következő karakterre ugrás”) és `.` (időszakos karakter).

Válaszok a gondolkodtató feladatokra

1. A `vim` lehetővé teszi tetszőleges szélességű szövegblokkok kijelölését, nem csak egész sorokat tartalmazó szakaszokét. Normál módban a `Ctrl` + `V` billentyűkombináció megnyomásával a kurzor felfelé, lefelé, balra és jobbra mozgatásával történik a kijelölés. Ezzel a módszerrel hogyan lehetne törölni az aktuális sor első karakterétől kezdődő, a következő nyolc oszlopot és öt sornyi szöveget tartalmazó blokkot?

   A `0`, a `Ctrl-V` és a `8l5jd` kijelöli és törli a megfelelő blokkot.

2. Egy `vi` munkamenet megszakadt egy váratlan áramkimaradás miatt. A fájl újbóli megnyitásakor a `vi` megkérdezi a felhasználót, hogy vissza akarja-e állítani a swap fájlt (a `vi` által készített automatikus másolatot). Mit tegyen a felhasználó a swap-fájl elvetéséhez?

   Nyomjuk meg a `d` billentyűt, amikor a `vi` kéri.

3. Egy `vim` munkamenetben egy sor korábban az `l` regiszterbe lett másolva. Milyen billentyűkombinációval rögzítenénk egy makrót az `a` regiszterben, hogy az `l` regiszterben lévő sort közvetlenül az aktuális sor elé illesszük be?

   A `qa"lPq` kombinációval, ahol a `q` (“makrórögzítés elkezdése”), `a` (“` az `a` regiszter hozzárendelése a makróhoz`”), `"l` (“` `l` regiszterben lévő szöveg kijelölése `"), `P` (“aktuális sor elé történő beillesztés”) és `q` (“makrórögzítés vége”).

Linux Professional Institute

Témakör 104: Eszközök, Linux fájlrendszerek, Fájlrendszer-hierarchia szabvány

104.1 Partíciók és fájlrendszerek létrehozása

Hivatkozás az LPI célkitűzésre

LPIC-1 v5, Exam 101, Objective 104.1

Súlyozás

2

Kulcsfontosságú ismeretek

- MBR és GPT partíciós táblák kezelése
- Különböző mkfs parancsok használata különböző fájlrendszerek létrehozásához, mint például:
  - ext2/ext3/ext4
  - XFS
  - VFAT
  - exFAT
- Alapvető ismeretek a Btrfs funkcióiról, beleértve a többeszközös fájlrendszereket, a tömörítést és az alkötegeket.

A használt fájlok, kifejezések és segédprogramok listája

- `fdisk`
- `gdisk`
- `parted`
- `mkfs`
- `mkswap`

104.1 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	104 Eszközök, Linux fájlrendszerek, Fájlrendszer-hierarchia szabvány
Fejezet:	104.1 Partíciók és fájlrendszerek létrehozása
Lecke:	1/1

Bevezetés

Bármely operációs rendszerben a lemezt partícionálni kell, mielőtt használni lehetne. A partíció a fizikai lemez egy logikai részhalmaza, és a partíciókról szóló információk egy partíciós táblázatban vannak tárolva. Ez a táblázat tartalmazza a partíció első és utolsó szektorára és típusára vonatkozó információkat, valamint az egyes partíciók további részleteit.

Az operációs rendszer általában minden partíciót külön "lemezként" kezel, még akkor is, ha mind ugyanazon a fizikai adathordozón található. A Windows rendszerekben olyan betűket kapnak, mint a `C:` (történelmi okok miatt ez a fő (main)), `D:` és így tovább. Linuxon minden partíciót a `/dev` alatt található mappához rendelnek, például `/dev/sda1` vagy `/dev/sda2`.

Ebben a leckében megtanuljuk, hogyan hozhatunk létre, törölhetünk, állíthatunk vissza partíciókat és hogyan változtathatjuk meg méretüket a három leggyakoribb segédprogrammal (`fdisk`, `gdisk` és `parted`), hogyan hozhatunk létre fájlrendszert rajtuk, és hogyan hozhatunk létre és állíthatunk be egy *swap partíciót* vagy *swap fájlt*, amelyet virtuális memóriaként használhatunk.

NOTE Történelmi okokból ebben a leckében a tárolóeszközökre “lemezek”-ként fogunk hivatkozni, annak ellenére, hogy a modern tárolórendszerek, például az SSD-k és a flash tárolók egyáltalán nem tartalmaznak “lemezeket”.

Az MBR és GPT megértése

A merevlemezeken a partíciós információk tárolásának két fő módja van. Az első az MBR (*Master Boot Record*), a második pedig a GPT (*GUID Partition Table*).

MBR

Ez az MS-DOS (pontosabban az 1983-as PC-DOS 2.0) korai időszakából maradt fenn, és évtizedekig ez volt a PC-k szabványos particionálási sémája. A partíciós tábla a lemez első szektorában, az úgynevezett *Boot Sector*-ban található, a bootloaderrel együtt, amely a Linux rendszereken általában a *GRUB*. Az MBR-nek azonban van egy sor olyan korlátja, amely akadályozza a modern rendszereken való használatát, mint például az, hogy nem képes 2 TB-nál nagyobb méretű lemezek címzésére, és hogy lemezenként csak 4 elsődleges partíciót tartalmaz.

GUID

Olyan particionálási rendszer, amely az MBR számos korlátját kiküszöböli. A lemez méretének nincs gyakorlati korlátja, és a partíciók maximális számát csak maga az operációs rendszer korlátozza. Ez a rendszer gyakrabban megtalálható a modernebb gépeken, amelyek a régi PC BIOS helyett UEFI-t használnak.

A rendszergazdai feladatok során nagyon valószínű, hogy mindkét séma használatban lesz, ezért fontos tudni, hogyan kell használni az egyes sémákhoz tartozó eszközöket a partíciók létrehozásához, törléséhez vagy módosításához.

MBR partíciók menedzselése FDISK-el

Az MBR partíciók kezelésének szabványos segédprogramja Linuxon az `fdisk`. Ez egy interaktív, menüvezérelt segédprogram. Használatához írjuk be az `fdisk` parancsot, majd a szerkeszteni kívánt lemeznek megfelelő eszköz nevét. Például a

```
# fdisk /dev/sda
```

parancs a rendszer első SATA-csatlakozású eszközének (`sda`) partíciós tábláját szerkeszti. Ne feledjük, hogy a fizikai lemeznek megfelelő eszközt kell megadni, nem pedig annak egyik partícióját (például `/dev/sda1`).

NOTE Ebben a leckében minden lemezzel kapcsolatos műveletet a `root` (a rendszergazda) felhasználóként kell elvégezni, vagy root jogosultságokkal a `sudo` használatával.

Meghíváskor az `fdisk` egy üdvözlést, majd egy figyelmeztetést jelenít meg, és várja a parancsokat.

```
# fdisk /dev/sda
Welcome to fdisk (util-linux 2.33.1). (Üdvözöllek a fdisk (util-linux 2.33.1)-en)
Changes will remain in memory only, until you decide to write them. (A változtatások a csak
memóriában vannak jelen, amíg úgy nem döntünk, hogy beírjuk azokat.)
Be careful before using the write command. (Legyünk óvatosak a write parancs használata
előtt.)

Command (m for help):
```

A figyelmeztetés fontos. Bármikor létrehozhatunk, szerkeszthetünk vagy törölhetünk partíciókat, de a lemezre *nem íródik semmi*, hacsak nem használjuk a write (`w`) parancsot. Tehát `gyakorolhatunk` anélkül, hogy az adatvesztés veszélyét kockáztatnánk, amíg távoltartjuk magunkat a `w` billentyűtől. Az `fdisk`-ből a változtatások mentése nélkül a `q` paranccsal léphetünk ki.

NOTE Ennek ellenére soha nem szabad fontos lemezen gyakorolni, mivel mindig vannak kockázatok. Használjunk inkább egy tartalék külső lemezt vagy USB flash meghajtót.

Az aktuális partíciós tábla kiíratása

A `p` parancs az aktuális partíciós tábla megjelenítésére szolgál. A kimenet valami ilyesmi lesz:

```
Command (m for help): p
Disk /dev/sda: 111.8 GiB, 120034123776 bytes, 234441648 sectors
Disk model: CT120BX500SSD1
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x97f8fef5

Device     Boot     Start       End   Sectors   Size Id Type
/dev/sda1             4096 226048942 226044847 107.8G 83 Linux
/dev/sda2       226048944 234437550   8388607     4G 82 Linux swap / Solaris
```

Itt vannak az oszlopok jelentései:

`Device`

A partícióhoz rendelt eszköz.

`Boot`

Megmutatja, hogy a partíció “bootable”-e vagy sem.

`Start`

A szektor, ahol a partíció elkezdődik.

`End`

A szektor, ahol a partíció végződik.

`Sectors`

A partícióban lévő szektorok teljes száma. Szorozzuk meg a szektor méretével, hogy megkapjuk a partíció méretét bájtban.

`Size`

A partíció mérete “ember által olvasható” formátumban. A fenti példában az értékek gigabájtban vannak megadva.

`Id`

A partíció típusát reprezentáló numerikus érték.

`Type`

A partíciótípus leírása.

Elsődleges vs kibővített partíciók

Egy MBR lemezen 2 fő partíciótípus lehet, a *primary* (elsődleges) és a *extended* (kibővített). Mint már említettük, a lemezen csak 4 elsődleges partíció lehet, és ha a lemezt “bootolhatóvá” akarjuk tenni, akkor az első partíciónak elsődlegesnek kell lennie.

Ezt a korlátozást úgy lehet megkerülni, hogy létrehozunk egy kiterjesztett partíciót, amely a *logikai* partíciók tárolójaként szolgál. Lehet például egy elsődleges partíció, egy kiterjesztett partíció, amely a lemezterület fennmaradó részét foglalja el, és ezen belül öt logikai partíció.

Egy olyan operációs rendszer esetében, mint a Linux, az elsődleges és a kiterjesztett partíciókat pontosan ugyanúgy kezelik, így nincsenek “előnyei” az egyiknek a másikkal szemben.

Partíció létrehozása

Partíció létrehozásához használjuk az `n` parancsot. Alapértelmezés szerint a partíciókat a lemez ki nem osztott területének kezdetén hozza létre a rendszer. A parancs megkérdezi a partíció típusát (elsődleges vagy kiterjesztett), valamint az első és az utolsó szektort.

Az első szektor esetében általában elfogadhatjuk az `fdisk` által javasolt alapértelmezett értéket, kivéve, ha a partíciónak egy adott szektorban kell kezdődnie. Az utolsó szektor megadása helyett megadhatunk egy méretet, amelyet a `K`, `M`, `G`, `T` vagy `P` (Kilo, Mega, Giga, Tera vagy Peta) betűk követnek. Ha tehát 1 GB-os partíciót szeretnénk létrehozni, akkor az `Last sector`-ként megadhatjuk a `+1G` értéket, és az `fdisk` ennek megfelelően méretezi a partíciót. Az elsődleges partíció létrehozására íme egy példa:

```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-3903577, default 2048): 2048
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-3903577, default 3903577): +1G
```

Allokálatlan hely keresése

Ha nem tudjuk, mennyi szabad hely van a lemezen, az `F` paranccsal megjeleníthetjük a ki nem osztott helyeket, például így:

```
Command (m for help): F
Unpartitioned space /dev/sdd: 881 MiB, 923841536 bytes, 1804378 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes

  Start     End Sectors  Size
2099200 3903577 1804378  881M
```

Partíciók törlése

Egy partíció törléséhez használjuk a `d` parancsot. Az `fdisk` megkérdezi a törlendő partíció számát, *hacsak* nem_csak_ egy partíció van a lemezen. Ebben az esetben ez a partíció lesz *kiválasztva és azonnal törölve*.

Ne feledjük, hogy ha egy kiterjesztett partíciót törlünk, a benne lévő összes logikai partíció is törlődik!

Mind the Gap! - Ügyeljünk a résekre a partíciók közt!

Ne feledjük, hogy amikor új partíciót hozunk létre az `fdisk` segítségével, a maximális méret a lemezen lévő *összefüggő* ki nem osztott hely maximális méretére korlátozódik. Tegyük fel, hogy például a következő partíciótérképünk van:

```
Device     Boot   Start     End Sectors  Size Id Type
/dev/sdd1          2048 1050623 1048576  512M 83 Linux
/dev/sdd2       1050624 2099199 1048576  512M 83 Linux
/dev/sdd3       2099200 3147775 1048576  512M 83 Linux
```

Majd töröljük a 2-es partíciót és megnézzük a szabad helyet:

```
Command (m for help): F
Unpartitioned space /dev/sdd: 881 MiB, 923841536 bytes, 1804378 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes

  Start     End Sectors  Size
1050624 2099199 1048576  512M
3147776 3903577  755802  369M
```

Ha összeadjuk a fel nem osztott tárhely méretét, elméletileg 881 MB áll rendelkezésünkre. De nézzük meg, mi történik, ha megpróbálunk létrehozni egy 700 MB-os partíciót:

```
Command (m for help): n
Partition type
   p   primary (2 primary, 0 extended, 2 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (2,4, default 2): 2
First sector (1050624-3903577, default 1050624):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (1050624-2099199, default 2099199): +700M
Value out of range.
```

Ez azért történt, mert a lemezen a legnagyobb összefüggő, ki nem osztott terület a 2. partícióhoz tartozó 512 MB-os blokk. Az új partíció nem tud “átnyúlni” a 3. partíció fölött, hogy az utána lévő

fel nem osztott terület egy részét használja.

Partíció típusának megváltoztatása

Esetenként szükség lehet a partíció típusának megváltoztatására, különösen akkor, ha olyan lemezekről van szó, amelyeket más operációs rendszereken és platformokon is használni szeretnénk. Ezt a `t` paranccsal tehetjük meg, amelyet a megváltoztatni kívánt partíció száma követ.

A partíció típusát a megfelelő hexadecimális kóddal kell megadni, az összes érvényes kód listáját az `l` parancs segítségével nézhetjük meg.

Ne keverjük össze a partíció típusát a rajta használt fájlrendszerrel. Bár kezdetben volt köztük kapcsolat, ma már nem feltételezhetjük, hogy ez igaz. Egy Linux partíció például bármilyen Linux-natív fájlrendszert tartalmazhat, például *ext4* vagy *ReiserFS*.

TIP A Linux partíciók típusa `83` (Linux). A swap partícióké `82` (Linux Swap).

GUID partíciók menedzselése a GDISK segítségével

A `gdisk` segédprogram az `fdisk` megfelelője, ha GPT partícionált lemezekkel dolgozunk. Valójában az interfész az `fdisk` mintájára készült, interaktív prompttal és ugyanazokkal (vagy nagyon hasonló) parancsokkal.

Az aktuális partíciós tábla kiíratása

Az `p` parancs az aktuális partíciós tábla kiíratására szolgál. A kimenet valami ilyesmi lesz:

```
Command (? for help): p
Disk /dev/sdb: 3903578 sectors, 1.9 GiB
Model: DataTraveler 2.0
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): AB41B5AA-A217-4D1E-8200-E062C54285BE
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 3903544
Partitions will be aligned on 2048-sector boundaries
Total free space is 1282071 sectors (626.0 MiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         2099199   1024.0 MiB  8300  Linux filesystem
   2         2623488         3147775   256.0 MiB   8300  Linux filesystem
```

Már az elején észreveszünk néhány különböző dolgot:

- Minden lemez egyedi lemezazonosítóval (GUID) rendelkezik. Ez egy 128 bites hexadecimális szám, amely a partíciós tábla létrehozásakor véletlenszerűen kerül kiosztásra. Mivel ennek a számnak $3{,}4 \times 10^{38}$ lehetséges értéke van, elég kicsi az esélye annak, hogy két véletlenszerű lemeznek ugyanaz a GUID-je legyen. A GUID használható annak azonosítására, hogy indításkor melyik fájlrendszert (és hova) kell csatlakoztatni, így nem szükséges az eszköz elérési útvonalát használni (mint például a `/dev/sdb`).
- Látjuk a `Partition table holds up to 128 entries` kifejezést? Így van, egy GPT lemezen akár 128 partíció is lehet. Emiatt nincs szükség *primary* és *extended* partíciókra.
- A szabad hely az utolsó sorban szerepel, így nincs szükség az fdisk `F` parancsának megfelelőjére.

Partíció létrehozása

A partíció létrehozásának parancsa az `n`, ugyanúgy, mint az `fdisk`-ben. A fő különbség az, hogy a partíció száma, valamint az első és utolsó szektor (vagy mérete) mellett a létrehozás során megadhatjuk a partíció típusát is. A GPT partíciók sokkal több típust támogatnak, mint az MBR. A támogatott típusok listáját az `l` parancs segítségével ellenőrizhetjük.

Partíció törlése

Egy partíció törléséhez írjuk be a `d`-t és a partíció számát. Az `fdisk`-től eltérően az első partíció nem lesz automatikusan kiválasztva, ha ez az egyetlen partíció a lemezen.

A GPT lemezeken a partíciókat könnyen újra lehet rendszerezni vagy “rendezni”, hogy elkerülhetők legyenek a számozási sorrendben lévő hiányosságok. Ehhez egyszerűen használjuk az `s` parancsot. Képzeljünk el például egy lemezt a következő partíciós táblával:

```
Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         2099199   1024.0 MiB  8300  Linux filesystem
   2         2099200         2361343   128.0 MiB   8300  Linux filesystem
   3         2361344         2623487   128.0 MiB   8300  Linux filesystem
```

Ha a második partíciót töröljük, a tábla ilyen lesz:

```
Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         2099199   1024.0 MiB  8300  Linux filesystem
   3         2361344         2623487   128.0 MiB   8300  Linux filesystem
```

Ha használjuk az `s` parancsot, ilyen:

```
Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         2099199   1024.0 MiB  8300  Linux filesystem
   2         2361344         2623487   128.0 MiB   8300  Linux filesystem
```

Vegyük észre, hogy a harmadik partíció lett a második.

Rés? Milyen Rés?

Az MBR lemezekkel ellentétben a GPT lemezeken a partíció létrehozásakor a partíció méretét nem korlátozza a *összefüggő* ki nem osztott terület maximális mennyisége. A szabad szektor minden egyes bitjét felhasználhatjuk, függetlenül attól, hogy az a lemezen hol található.

Helyreállítási lehetőségek

A GPT lemezek a GPT fejléc és a partíciós tábla biztonsági másolatát tárolják, így a lemezek könnyen helyreállíthatók, ha ezek az adatok megsérültek. A `gdisk` olyan funkciókat biztosít, amelyek segítik ezeket a helyreállítási feladatokat, és az `r` paranccsal érhető el.

A sérült GPT fő fejlécet vagy partíciós táblát a `b` és `c` paranccsal állíthatjuk helyre, vagy a fő fejlécet és táblát használhatjuk a biztonsági mentés helyreállításához a `d` és `e` paranccsal. Egy MBR-t GPT-vé alakíthatunk át az `f` paranccsal, és az ellenkezőjét is megtehetjük a `g` paranccsal, egyéb műveletek mellett. Írjuk be a helyreállítási menübe a `?` parancsot, hogy megkapjuk a rendelkezésre álló helyreállítási parancsok listáját és a leírást arról, hogy mit csinálnak.

Fájlrendszerek létrehozása

A lemez particionálása csak az első lépés a lemez használatához. Ezt követően a partíciót fájlrendszerrel kell formázni, mielőtt adatok tárolására használnánk.

A fájlrendszer szabályozza az adatok tárolását és elérését a lemezen. A Linux számos fájlrendszert támogat, némelyik natív, mint például az ext (Extended Filesystem) család, míg mások más operációs rendszerekből származnak, mint például a FAT az MS-DOS-ból, az NTFS a Windows NT-ből, a HFS és HFS+ a Mac OS-ből stb.

A Linuxon a fájlrendszerek létrehozására használt szabványos eszköz az `mkfs`, amely sokféle “ízben” létezik, aszerint, hogy milyen fájlrendszerrel kell dolgoznunk.

ext2/ext3/ext4 fájlrendszer létrehozása

A *Extended Filesystem* (ext) volt a Linux első fájlrendszere, amelyet az évek során új változatok váltottak fel, az ext2, ext3 és ext4, utóbbi jelenleg is sok Linux-disztribúció alapértelmezett fájlrendszere.

Az `mkfs.ext2`, `mkfs.ext3` és `mkfs.ext4` segédprogramok az ext2, ext3 és ext4 fájlrendszerek létrehozására szolgálnak. Valójában ezek a "segédprogramok" csak szimbolikus linkként léteznek egy másik segédprogramra, az `mke2fs`-re. Az `mke2fs` az alapértelmezett beállításait aszerint változtatja meg, hogy milyen néven hívják. Mint ilyenek, mindegyikük viselkedése és parancssori paraméterei megegyeznek.

A legegyszerűbb felhasználási mód:

```
# mkfs.ext2 TARGET
```

Ahol a `TARGET` annak a partíciónak a neve, ahol a fájlrendszert létre kell hozni. Például egy ext3 fájlrendszer létrehozásához a `/dev/sdb1` partíción a parancs a következő:

```
# mkfs.ext3 /dev/sdb1
```

Ahelyett, hogy a létrehozni kívánt fájlrendszernek megfelelő parancsot használnánk, átadhatjuk a `-t` paramétert a `mke2fs`-nek, amelyet a fájlrendszer neve követ. Például a következő parancsok egyenértékűek, és egy ext4 fájlrendszert hoznak létre a `/dev/sdb1` lemezen.

```
# mkfs.ext4 /dev/sdb1
# mke2fs -t ext4 /dev/sdb1
```

Parancssori paraméterek

Az `mke2fs` parancssori paraméterek és opciók széles skáláját támogatja. Íme néhány a legfontosabbak közül. Ezek mindegyike alkalmazható az `mkfs.ext2`, `mkfs.ext3` és `mkfs.ext4` esetében is:

`-b SIZE`

Az eszközben lévő adatblokkok méretét a `SIZE` értékre állítja be, ami lehet 1024, 2048 vagy 4096 bájt blokkonként.

`-c`

A fájlrendszer létrehozása előtt ellenőrzi a céleszközön a rossz blokkokat. Alapos, de sokkal lassabb ellenőrzést végezhetünk, ha ezt a paramétert kétszer adjuk meg, mint az `mkfs.ext4 -c -c TARGET` paraméterben.

`-d DIRECTORY`

A megadott mappa tartalmát az új fájlrendszer gyökerébe másolja. Hasznos, ha a lemezt "előzetesen" fel kell tölteni előre meghatározott fájlokkal.

`-F`

Veszély, Will Robinson! Ez az opció *kényszeríti* az mke2fs-t, hogy létrehozzon egy fájlrendszert, még akkor is, ha a neki vagy a célnak átadott egyéb opciók veszélyesek vagy értelmetlenek. Ha kétszer adjuk meg (mint az `-F -F`), akkor még arra is használható, hogy létrehozzunk egy fájlrendszert egy olyan eszközön, ami be van csatolva vagy használatban van, ami nagyon, *nagyon* rossz dolog.

`-L VOLUME_LABEL`

A kötet címkéjét a `VOLUME_LABEL` paraméterben megadott címkéjére állítja be. Ez a címke legfeljebb 16 karakter hosszú lehet.

`-n`

Ez egy igazán hasznos opció, amely szimulálja a fájlrendszer létrehozását, és megmutatja, hogy mi történne, ha az `n` opció nélkül hajtanánk végre. Gondoljunk rá úgy, mint egy "próba" üzemmódra. Jó a dolgok ellenőrzésére, mielőtt bármilyen változtatást ténylegesen a lemezre rögzítenénk.

`-q`

Csendes üzemmód. Az `mke2fs` normálisan fut, de nem ad ki semmilyen kimenetet a terminálra. Hasznos, ha az `mke2fs`-t egy szkriptből futtatjuk.

`-U ID`

Ez a partíció UUID-jét (*Universally Unique Identifier*) az ID-ként megadott értékre állítja be. Az UUID-k 128 bites, hexadecimális jelölésű számok, amelyek arra szolgálnak, hogy a partíciót az operációs rendszer számára egyedileg azonosítsák. Ezt a számot 32 számjegyű sztringként kell megadni 8-4-4-4-12 formátumban, azaz 8 számjegy, kötőjel, 4 számjegy, kötőjel, 4 számjegy, kötőjel, 4 számjegy, kötőjel, 12 számjegy, például `D249E380-7719-45A1-813C-35186883987E`. Az azonosító helyett olyan paramétereket is megadhatunk, mint a `clear` a fájlrendszer UUID-jének törléséhez, a `random` egy véletlenszerűen generált UUID használatához, vagy a `time` egy időalapú UUID létrehozásához.

`-V`

Verbose mód, a szokásosnál sokkal több információt ír ki működés közben. Hasznos hibakeresési célokra.

XFS fájlrendszer létrehozása

Az XFS egy nagy teljesítményű fájlrendszer, amelyet eredetileg a Silicon Graphics fejlesztett ki 1993-ban az IRIX operációs rendszerhez. Teljesítménye és megbízhatósági jellemzői miatt általában szerverekhez és más olyan környezetekhez használják, amelyek nagy (vagy garantált) fájlrendszer-sávszélességet igényelnek.

Az XFS fájlrendszerek kezelésére szolgáló eszközök az `xfsprogs` csomag részét képezik. Ezt a csomagot esetleg manuálisan kell telepíteni, mivel néhány Linux-disztribúcióban alapértelmezés szerint nem szerepel. Mások, mint például a Red Hat Enterprise Linux 7, az XFS-t használják alapértelmezett fájlrendszerként.

Az XFS fájlrendszerek legalább két részre oszlanak, egy *log szekcióra*, ahol a fájlrendszer összes műveletének naplóját (általában *Journal_nak nevezik) vezetik, és az _adat szekcióra*. A naplórész az adatrészen belül is elhelyezkedhet (ez az alapértelmezett viselkedés), vagy akár egy külön lemezen is, a jobb teljesítmény és megbízhatóság érdekében.

Az XFS fájlrendszer létrehozásának legalapvetőbb parancsa az `mkfs.xfs TARGET`, ahol a `TARGET` az a partíció, amelyen a fájlrendszert létrehozni szeretnénk. Például: `mkfs.xfs /dev/sda1`.

Az `mke2fs`-hez hasonlóan az `mkfs.xfs` is számos parancssori opciót támogat. Íme néhány a leggyakoribbak közül.

`-b size=VALUE`

A fájlrendszerben lévő blokkméretet bájtban a `VALUE`-ban megadott értékre állítja be. Az alapértelmezett érték 4096 bájt (4 KiB), a minimális érték 512, a maximális pedig 65536 (64 KiB).

`-m crc=VALUE`

Az `-m`-el kezdődő paraméterek metaadat opciók. Ez engedélyezi (ha a `VALUE` értéke `1`) vagy letiltja (ha a `VALUE` értéke `0`) a CRC32c ellenőrzés használatát a lemezen lévő metaadatok integritásának ellenőrzésére. Ez lehetővé teszi a jobb hibaérzékelést és a hardverproblémákhoz kapcsolódó összeomlásokból való helyreállítást, ezért alapértelmezés szerint be van kapcsolva. Ennek az ellenőrzésnek a teljesítményre gyakorolt hatása minimális, így általában nincs ok a kikapcsolására.

`-m uuid=VALUE`

A partíció UUID-jét a VALUE-ként megadottra állítja be. Ne feledjük, hogy az UUID-k 32 karakteres (128 bites) számok hexadecimális alapon, 8, 4, 4, 4 és 12 számjegyű, kötőjellel elválasztott csoportokban, például `1E83E3A3-3AE9-4AAC-BF7E-29DFFECD36C0`.

`-f`

Kényszeríti egy fájlrendszer létrehozását a céleszközön, még akkor is, ha a céleszközön már van fájlrendszer.

`-l logdev=DEVICE`

Ez a fájlrendszer naplószekcióját a megadott eszközön helyezi el az adatszekció helyett.

`-l size=VALUE`

Ez a naplószekció méretét a `VALUE` értékben megadott méretre állítja be. A méret megadható bájtokban, és olyan utótagok is használhatók, mint az `m` vagy `g`. Az `-l size=10m` például 10 megabájtra korlátozza a naplószekciót.

`-q`

Csendes üzemmód. Ebben az üzemmódban az `mkfs.xfs` nem fogja kiírni a létrehozandó fájlrendszer paramétereit.

`-L LABEL`

Beállítja a fájlrendszer címkéjét, amely legfeljebb 12 karakter hosszú lehet.

`-N`

Hasonlóan az `mke2fs -n` paraméteréhez, az `mkfs.xfs` kiírja az összes paramétert a fájlrendszer létrehozásához, anélkül, hogy ténylegesen létrehozná azt.

FAT vagy VFAT fájlrendszer létrehozása

A FAT fájlrendszer az MS-DOS-ból származik, és az évek során számos átdolgozása jelent meg, amelyek 1996-ban a Windows 95 OSR2-vel kiadott FAT32 formátumban csúcsosodtak ki.

A VFAT a FAT16 formátum kiterjesztése a hosszú (legfeljebb 255 karakteres) fájlnevek támogatásával. Mindkét fájlrendszert ugyanaz a segédprogram, az `mkfs.fat` kezeli. Az `mkfs.vfat` ennek egy alias változata.

A FAT fájlrendszernek jelentős hátrányai vannak, amelyek korlátozzák a nagy lemezeken való használatát. A FAT16 például legfeljebb 4 GB-os köteteket és legfeljebb 2 GB-os fájlméretet támogat. A FAT32 a kötetméretet 2 PB-ra, a maximális fájlméretet pedig 4 GB-ra növeli. Emiatt a FAT fájlrendszereket manapság inkább a kis méretű (legfeljebb 2 GB-os) flash meghajtókon vagy

memóriakártyákon, illetve a fejlettebb fájlrendszereket nem támogató elavult eszközökön és operációs rendszereken használják.

A FAT fájlrendszer létrehozásának legalapvetőbb parancsa az `mkfs.fat TARGET`, ahol a `TARGET` az a partíció, amelyen a fájlrendszert létre akarjuk hozni. Például: `mkfs.fat /dev/sdc1`.

Más segédprogramokhoz hasonlóan az `mkfs.fat` is számos parancssori opciót támogat. Az alábbiakban a legfontosabbak következnek. Az összes opció teljes listája és leírása a segédprogram manualjában olvasható, a `man mkfs.fat` parancs segítségével.

`-c`

A fájlrendszer létrehozása előtt ellenőrzi a céleszközt, hogy vannak-e rossz blokkjai.

`-C FILENAME BLOCK_COUNT`

Létrehozza a `FILENAME`-ben megadott fájlt, majd létrehoz egy FAT fájlrendszert benne, gyakorlatilag létrehozva egy üres "lemezképet" (disk image), amely később egy eszközre írható egy olyan segédprogrammal, mint a `dd`, vagy csatolható loopback eszközként. Ha ezt az opciót használjuk, az eszköz neve után meg kell adni a fájlrendszerben lévő blokkok számát (`BLOCK_COUNT`).

`-F SIZE`

A FAT (*File Allocation Table*) méretének kiválasztása 12, 16 vagy 32, azaz FAT12, FAT16 vagy FAT32 között. Ha nincs megadva, az `mkfs.fat` a fájlrendszer mérete alapján választja ki a megfelelő opciót.

`-n NAME`

Beállítja a fájlrendszer kötetcímkéjét vagy nevét. Ez legfeljebb 11 karakter hosszú lehet, és alapértelmezés szerint név nélküli.

`-v`

Szöveges üzemmód. A szokásosnál sokkal több információt jelenít meg, ami hasznos a hibakereséshez.

NOTE Az `mkfs.fat` *nem* tud "bootolható" fájlrendszert létrehozni. A kézikönyv oldala szerint "ez nem olyan egyszerű, mint gondolnád", és nem lesz implementálva.

exFAT fájlrendszer létrehozása

Az exFAT a Microsoft által 2006-ban létrehozott fájlrendszer, amely a FAT32 egyik legfontosabb korlátját, a fájl- és lemezméretet kezeli. Az exFAT rendszeren a maximális fájlméret 16 exabájt (a FAT32 4 GB-járól növelve), a maximális lemezméret pedig 128 petabájt.

Mivel mindhárom nagy operációs rendszer (Windows, Linux és mac OS) jól támogatja, jó választás ott, ahol interoperabilitásra van szükség, például nagy kapacitású flash meghajtókon, memóriakártyákon és külső lemezeken. Valójában ez az *SD Association* által meghatározott alapértelmezett fájlrendszer a 32 GB-nál nagyobb SDXC memóriakártyákhoz.

Az exFAT fájlrendszerek létrehozására szolgáló alapértelmezett segédprogram az `mkfs.exfat`, amely az `mkexfatfs` linkje. A legalapvetőbb parancs az `mkfs.exfat TARGET`, ahol a `TARGET` az a partíció, amelyen a fájlrendszert létre akarjuk hozni. Például: `mkfs.exfat /dev/sdb2`.

A leckében tárgyalt többi segédprogrammal ellentétben az `mkfs.exfat` nagyon kevés parancssori opcióval rendelkezik. Ezek a következők:

`-i VOL_ID`

A kötet azonosítóját a `VOL_ID` paraméterben megadott értékre állítja. Ez egy 32 bites hexadecimális szám. Ha nincs megadva, akkor az aktuális időn alapuló azonosító kerül beállításra.

`-n NAME`

A kötet címkéjének vagy nevének beállítása. Ez legfeljebb 15 karakter lehet, és az alapértelmezés szerint nincs neve.

`-p SECTOR`

Megadja a lemez első partíciójának első szektorát. Ez egy opcionális érték, és az alapértelmezés szerint nulla.

`-s SECTORS`

Meghatározza a fizikai szektorok számát kiosztási klaszterenként. Ennek kettő hatványának kell lennie, például 1, 2, 4, 8, stb.

A Btrfs fájlrendszer megismerése

A Btrfs (hivatalosan *B-Tree Filesystem*, kiejtése: “Butter FS”, “Better FS” vagy akár “Butterfuss”) egy olyan fájlrendszer, amelyet 2007 óta fejleszt az Oracle Corporation és más vállalatok, többek között a Fujitsu, a Red Hat, az Intel és a SUSE, kifejezetten Linuxhoz.

A Btrfs-nek számos olyan tulajdonsága van, amely vonzóvá teszi a modern rendszerekben, ahol gyakori a hatalmas mennyiségű tárolóhely. Ezek közé tartozik a több eszköz támogatása (beleértve a csíkozást (striping), tükrözést (mirroring) és csíkozás+tükrözést, mint egy RAID beállításban), az átlátható tömörítés, az SSD optimalizálás, inkrementális mentések, pillanatképek, online töredezettségmentesítés (defragmentálás), offline ellenőrzések, alkötegek támogatása (kvótákkal), deduplikáció és még sok más.

Mivel ez egy *copy-on-write* fájlrendszer, nagyon ellenálló az összeomlásokkal szemben. Ráadásul a Btrfs egyszerűen használható, és sok Linux disztribúció jól támogatja. Néhányan közülük, mint például a SUSE, alapértelmezett fájlrendszerként használják.

NOTE Egy hagyományos fájlrendszeren, amikor egy fájl egy részét felül szeretnénk írni, az új adat közvetlenül a régi adat fölé kerül, amelyet felcserél. Egy *copy-on-write* fájlrendszerben az új adatot a lemezen lévő szabad helyre írjuk, majd a fájl eredeti metaadatait frissítjük, hogy az új adatokra hivatkozzanak, és csak ezután szabadítjuk fel a régi adatokat, mivel már nincs rájuk szükség. Ez csökkenti az adatvesztés esélyét egy összeomlás esetén, mivel a régi adatok csak akkor kerülnek törlésre, ha a fájlrendszer teljesen biztos benne, hogy már nincs rájuk szükség, és az új adatok a helyükön vannak.

Btrfs fájlrendszer létrehozása

Az `mkfs.btrfs` segédprogramot a Btrfs fájlrendszer létrehozására használjuk. A parancs opciók nélküli használata egy Btrfs fájlrendszert hoz létre egy adott eszközön, a következőképpen:

```
# mkfs.btrfs /dev/sdb1
```

TIP Ha nincs meg az `mkfs.btrfs` segédprogram a rendszerünkben, a `btrfs-progs` programot kell keresni a disztribúció csomagkezelőjében.

A `-L` segítségével megadhatunk egy címkét (vagy nevet) a fájlrendszernek. A Btrfs címkék legfeljebb 256 karakteresek lehetnek, kivéve az újsorokat:

```
# mkfs.btrfs /dev/sdb1 -L "New Disk"
```

TIP Zárjuk a címkét idézőjelek közé (mint fentebb), ha szóközöket tartalmaz.

Figyeljük meg a Btrfs sajátos tulajdonságát: az `mkfs.btrfs` parancsnak *több* eszközt is átadhatunk. Ha egynél több eszközt adunk meg, akkor a fájlrendszer az összes eszközre kiterjed, ami hasonló egy RAID vagy LVM beállításhoz. A metaadatok lemeztömbön belüli elosztásának módját az `-m` paraméterrel adhatjuk meg. Az érvényes paraméterek a következők: `raid0`, `raid1`, `raid5`, `raid6`, `raid10`, `single` és dup.

Például a `/dev/sdb1` és `/dev/sdc1` fájlrendszer létrehozásához, a két partíciót egyetlen nagy partícióvá fűzve, használjuk a következőt:

```
# mkfs.btrfs -d single -m single /dev/sdb /dev/sdc
```

WARNING A több partíciót átfogó fájlrendszerek, mint a fentiek, elsőre előnyösnek tűnhetnek, de adatbiztonsági szempontból nem jó ötlet, mivel a tömb egyetlen lemezének meghibásodása biztos adatvesztést jelent. A kockázat annál nagyobb, minél több lemezt használunk, mivel több lehetséges hibapont is van.

Sobvolumek kezelése

Az subvolumek (alkötetek) olyanok, mint a fájlrendszerek a fájlrendszereken belül. Gondoljunk rájuk úgy, mint egy mappára, amely külön fájlrendszerként csatolható (és kezelhető). Az alkötetek megkönnyítik a szervezést és a rendszeradminisztrációt, mivel mindegyikükhöz külön kvóták vagy pillanatkép-szabályok tartozhatnak.

NOTE A subvolumek nem partíciók. A partíció egy rögzített helyet jelöl ki a meghajtón. Ez a későbbiekben problémákhoz vezethet, például ahhoz, hogy az egyik partíció kifogy a helyéből, miközben egy másiknak még bőven van helye. Nem így van ez a subvolumekkel, mivel ezek “megosztják” a gyökérfájlrendszerük szabad helyét, és szükség szerint növekednek.

Tegyük fel, hogy van egy Btrfs fájlrendszerünk, amelyet az `/mnt/disk` lemezre csatoltunk fel, és szeretnénk létrehozni egy subvolumet benne a biztonsági mentések tárolására. Nevezzük ezt `BKP` -nek:

```
# btrfs subvolume create /mnt/disk/BKP
```

Ezután felsoroljuk az `/mnt/disk` fájlrendszer tartalmát. Látni fogjuk, hogy van egy új mappánk, amely a subvolumeről kapta a nevét.

```
$ ls -lh /mnt/disk/
total 0
drwxr-xr-x 1 root   root     0 jul 13 17:35 BKP
drwxrwxr-x 1 carol carol 988 jul 13 17:30 Images
```

NOTE Igen, a subvolumekhez ugyanúgy hozzá lehet férni, mint bármely más mappához.

A következő paranccsal ellenőrizhetjük, hogy a subvolume aktív-e:

```
# btrfs subvolume show /mnt/disk/BKP/
    Name:           BKP
    UUID:           e90a1afe-69fa-da4f-9764-3384f66fa32e
    Parent UUID:        -
    Received UUID:      -
    Creation time:      2019-07-13 17:35:40 -0300
    Subvolume ID:       260
    Generation:         23
    Gen at creation:    22
    Parent ID:      5
    Top level ID:       5
    Flags:          -
    Snapshot(s):
```

Az alkötetet a `/mnt/BKP` lemezre csatolhatjuk a `-t btrfs -o subvol=NAME` paraméter `mount` parancsnak történő átadásával:

```
# mount -t btrfs -o subvol=BKP /dev/sdb1 /mnt/bkp
```

NOTE A `-t` paraméter megadja a csatolandó fájlrendszer típusát.

Munka a pillanatfelvételekkel

A pillanatfelvételek (snapshot) ugyanolyanok, mint a subvolumek, de előre feltöltődnek annak a kötetnek a tartalmával, amelyről a pillanatfelvétel készült.

Létrehozásakor a pillanatfelvétel és az eredeti kötet tartalma pontosan megegyezik. De ettől az időponttól kezdve eltérnek egymástól. Az eredeti kötetben végrehajtott változtatások (például hozzáadott, átnevezett vagy törölt fájlok) nem fognak tükröződni a pillanatfelvételen, és fordítva.

Ne feledjük, hogy a pillanatfelvétel nem duplikálja a fájlokat, és kezdetben szinte semmilyen lemezterületet nem foglal. Egyszerűen csak megkettőzi a fájlrendszer fáját, miközben az eredeti adatokra mutat.

A pillanatkép létrehozására használt parancs ugyanaz, mint a subvolumek létrehozására, csak a `btrfs subvolume` után a `snapshot` paramétert kell hozzáadni. Az alábbi parancs a `/mnt/disk` -re csatolt Btrfs fájlrendszer pillanatképét hozza létre a `/mnt/disk/snap`-ban:

```
# btrfs subvolume snapshot /mnt/disk /mnt/disk/snap
```

Most képzeljük el, hogy az `/mnt/disk`-ben az alábbi tartalmak vannak:

```
$ ls -lh
total 2,8M
-rw-rw-r-- 1 carol carol 109K jul 10 16:22 Galaxy_Note_10.png
-rw-rw-r-- 1 carol carol 484K jul  5 15:01 geminoid2.jpg
-rw-rw-r-- 1 carol carol 429K jul  5 14:52 geminoid.jpg
-rw-rw-r-- 1 carol carol 467K jul  2 11:48 LG-G8S-ThinQ-Mirror-White.jpg
-rw-rw-r-- 1 carol carol 654K jul  2 11:39 LG-G8S-ThinQ-Range.jpg
-rw-rw-r-- 1 carol carol  94K jul  2 15:43 Mimoji_Comparativo.jpg
-rw-rw-r-- 1 carol carol 112K jul 10 16:20 Note10Plus.jpg
drwx------ 1 carol carol  366 jul 13 17:56 snap
-rw-rw-r-- 1 carol carol 118K jul 11 16:36 Twitter_Down_20190711.jpg
-rw-rw-r-- 1 carol carol 324K jul  2 15:22 Xiaomi_Mimoji.png
```

Figyeljük meg a pillanatfelvételt tartalmazó snap mappát! Most távolítsunk el néhány fájlt, és ellenőrizzük a mappa tartalmát:

```
$ rm LG-G8S-ThinQ-*
$ ls -lh
total 1,7M
-rw-rw-r-- 1 carol carol 109K jul 10 16:22 Galaxy_Note_10.png
-rw-rw-r-- 1 carol carol 484K jul  5 15:01 geminoid2.jpg
-rw-rw-r-- 1 carol carol 429K jul  5 14:52 geminoid.jpg
-rw-rw-r-- 1 carol carol  94K jul  2 15:43 Mimoji_Comparativo.jpg
-rw-rw-r-- 1 carol carol 112K jul 10 16:20 Note10Plus.jpg
drwx------ 1 carol carol  366 jul 13 17:56 snap
-rw-rw-r-- 1 carol carol 118K jul 11 16:36 Twitter_Down_20190711.jpg
-rw-rw-r-- 1 carol carol 324K jul  2 15:22 Xiaomi_Mimoji.png
```

Ha azonban megnézzük a snap mappán belül, a törölt fájlok még mindig ott vannak és szükség esetén visszaállíthatók.

```
$ ls -lh snap/
total 2,8M
-rw-rw-r-- 1 carol carol 109K jul 10 16:22 Galaxy_Note_10.png
-rw-rw-r-- 1 carol carol 484K jul  5 15:01 geminoid2.jpg
-rw-rw-r-- 1 carol carol 429K jul  5 14:52 geminoid.jpg
-rw-rw-r-- 1 carol carol 467K jul  2 11:48 LG-G8S-ThinQ-Mirror-White.jpg
-rw-rw-r-- 1 carol carol 654K jul  2 11:39 LG-G8S-ThinQ-Range.jpg
-rw-rw-r-- 1 carol carol  94K jul  2 15:43 Mimoji_Comparativo.jpg
```

```
-rw-rw-r-- 1 carol carol 112K jul 10 16:20 Note10Plus.jpg
-rw-rw-r-- 1 carol carol 118K jul 11 16:36 Twitter_Down_20190711.jpg
-rw-rw-r-- 1 carol carol 324K jul  2 15:22 Xiaomi_Mimoji.png
```

Lehetőség van csak olvasható pillanatfelvételek létrehozására is. Ezek pontosan úgy működnek, mint az írhatók, azzal a különbséggel, hogy a tartalmuk nem változtatható meg, azok időben "befagyasztva" vannak. Csak adjuk hozzá a pillanatfelvétel létrehozásakor az `-r` paramétert:

```
# btrfs subvolume snapshot -r /mnt/disk /mnt/disk/snap
```

Néhány szó a tömörítésről

A Btrfs támogatja az átlátható fájltömörítést, három különböző algoritmus áll a felhasználó rendelkezésére. Ez automatikusan, fájlonként történik, amennyiben a fájlrendszer a `-o compress` opcióval van csatlakoztatva. Az algoritmusok elég okosak ahhoz, hogy felismerjék a tömöríthetetlen fájlokat, és nem próbálják meg tömöríteni őket, így takarékoskodva a rendszer erőforrásainak felhasználásával. Így egyetlen mappában lehetnek tömörített és tömörítetlen fájlok együtt. Az alapértelmezett tömörítési algoritmus a ZLIB, de elérhető az LZO (gyorsabb, rosszabb tömörítési arány) vagy a ZSTD (gyorsabb, mint a ZLIB, hasonló tömörítés), többféle tömörítési szinttel (lsd. a mount opciókról szóló fejezetet).

Partíciók menedzselése a GNU Parted segítségével

A *GNU Parted* egy nagyon hatékony partíciószerkesztő (partition editor, innen ered a neve), amely partíciók létrehozására, törlésére, áthelyezésére, méretének módosítására, mentésére és másolására használható. GPT és MBR lemezekkel egyaránt képes dolgozni, és szinte minden lemezkezelési igényt lefed.

Számos grafikus front-end létezik, amelyek megkönnyítik a `parted`-del való munkát, mint például a *GParted* a GNOME-alapú asztali környezetekhez és a *KDE Partition Manager* a KDE asztali számítógépekhez. Azonban érdemes megtanulni a `parted` parancssori használatát, mivel szerver környezetben soha nem számíthatunk arra, hogy grafikus asztali környezet áll rendelkezésre.

WARNING

Az `fdisk` és `gdisk` parancsokkal ellentétben a parted a parancs kiadása után *azonnal* változtat a lemezen, anélkül, hogy megvárna egy másik parancsot a változtatások lemezre írásához. Gyakorláskor célszerű ezt egy üres vagy tartalék lemezen vagy pendrive-on végezni, így hibázás esetén nem áll fenn az adatvesztés veszélye.

A legegyszerűbben úgy kezdhetjük el használni a parted-et, hogy beírjuk a `parted DEVICE`

parancsot, ahol a `DEVICE` a kezelni kívánt eszköz (`parted /dev/sdb`). A program elindít egy interaktív parancssori felületet, mint az `fdisk` és `gdisk`, a `(parted)` prompttal, ahol parancsokat írhatunk be.

```
# parted /dev/sdb
GNU Parted 3.2
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.

(parted)
```

WARNING Óvatosan! Ha nem adunk meg eszközt, a `parted` automatikusan kiválasztja az elsődleges lemezt (általában `/dev/sda`), amivel dolgozni fog.

Lemezek kiválasztása

Ha a parancssorban megadottól eltérő lemezre szeretnénk váltani, használhatjuk a `select` parancsot, amelyet az eszköz neve követ:

```
(parted) select /dev/sdb
Using /dev/sdb
```

Információk megszerzése

A `print` paranccsal több információt kaphatunk egy adott partícióról vagy akár a rendszerhez csatlakoztatott összes blokkeszközről (lemezről).

Ha információt szeretnénk kapni az aktuálisan kiválasztott partícióról, csak írjuk be azt, hogy `print`:

```
(parted) print
Model: ATA CT120BX500SSD1 (scsi)
Disk /dev/sda: 120GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End    Size    Type     File system     Flags
 1      2097kB  116GB  116GB   primary  ext4
 2      116GB   120GB  4295MB  primary  linux-swap(v1)
```

A rendszerhez csatlakoztatott összes blokkeszköz listáját a `print devices` paranccsal kaphatjuk meg:

```
(parted) print devices
/dev/sdb (1999MB)
/dev/sda (120GB)
/dev/sdc (320GB)
/dev/mapper/cryptswap (4294MB)
```

Ha egyszerre szeretnénk információt kapni az összes csatlakoztatott eszközről, használhatjuk a `print all` parancsot. Ha azt szeretnénk tudni, hogy mennyi szabad hely van az egyes eszközökön, akkor a `print free` parancs segít:

```
(parted) print free
Model: ATA CT120BX500SSD1 (scsi)
Disk /dev/sda: 120GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system     Flags
        32.3kB  2097kB  2065kB           Free Space
 1      2097kB  116GB   116GB   primary  ext4
        116GB   116GB   512B             Free Space
 2      116GB   120GB   4295MB  primary  linux-swap(v1)
        120GB   120GB   2098kB           Free Space
```

Partíciós tábla létrehozása üres lemezen

Ha partíciós táblát szeretnénk létrehozni egy üres lemezen, használjuk az `mklabel` parancsot, majd a használni kívánt partíciós tábla típusát.

Számos támogatott partíciós táblatípus létezik, de a főbb típusok, amelyeket ismerni kell, az `msdos`, amelyet itt az MBR partíciós táblára használunk, és az `gpt`, amely a GPT partíciós táblára utal. MBR partíciós tábla létrehozásához írjuk be a következőt:

```
(parted) mklabel msdos
```

GPT partíciós tábla létrehozásához a parancs pedig:

```
(parted) mklabel gpt
```

Partíció létrehozása

Partíció létrehozásához az `mkpart` parancsot kell használni, az `mkpart PARTTYPE FSTYPE START END` szintaxissal, ahol:

`PARTTYPE`

A partíció típusa, amely lehet `primary`, `logical` vagy `extended` MBR partíciós tábla használata esetén.

`FSTYPE`

Megadja, hogy melyik fájlrendszert használja a partíció. Vegyük figyelembe, hogy a `parted` *nem* hozza létre a fájlrendszert. Csak beállít egy flaget a partícióra, ami megmondja az operációs rendszernek, hogy milyen adatokat várjon tőle.

`START`

Megadja az eszköz pontos pontját, ahol a partíció kezdődik. A pont megadásához különböző mértékegységeket is használhatunk. A `2s` a lemez második szektorára utalhat, míg az `1m` a lemez első megabájtjának kezdetére. Más gyakori egységek a `B` (bájt) és a `%` (a lemez százalékos aránya).

`END`

Megadja a partíció végét. Vegyük figyelembe, hogy ez *nem* a partíció mérete, hanem *a lemez azon pontja, ahol a partíció véget ér*. Ha például a `100m` értéket adjuk meg, a partíció 100 MB-tal a lemez kezdete után ér véget. Ugyanazokat az egységeket használhatjuk, mint a `START` paraméterben.

Tehát, a parancs:

```
(parted) mkpart primary ext4 1m 100m
```

Létrehoz egy `ext4` típusú elsődleges partíciót, amely a lemez első megabájtjánál kezdődik és a 100. megabájt után ér véget.

Partíció eltávolítása

Egy partíció eltávolításához használjuk az `rm` parancsot, majd a partíció számát, amelyet a `print` paranccsal tudunk megjeleníteni. Tehát az `rm 2` a második partíciót távolítja el az aktuálisan

kiválasztott lemezen.

Partíciók helyreállítása

A `parted` helyre tud állítani törölt partíciókat. Képzeljük el az alábbi partíció-struktúrát:

```
Number  Start   End     Size    File system  Name     Flags
 1      1049kB  99.6MB  98.6MB  ext4         primary
 2      99.6MB  200MB   100MB   ext4         primary
 3      200MB   300MB   99.6MB  ext4         primary
```

Véletlenül eltávolítottuk a 2. partíciót az `rm 2` használatával. Visszaállításához használhatjuk a `rescue` parancsot a `rescue START END` szintaxissal, ahol a `START` a partíció kezdetének hozzávetőleges helye, a `END` pedig a partíció végének hozzávetőleges helye.

A `parted` átvizsgálja a lemezt a partíciók után kutatva, és felajánlja, hogy a megtalált partíciókat visszaállítja. A fenti példában a `2` partíció 99,6 MB-nál kezdődött és 200 MB-nál végződött. A partíció helyreállításához tehát a következő parancsot használhatjuk:

```
(parted) rescue 90m 210m
Information: A ext4 primary partition was found at 99.6MB -> 200MB.
Do you want to add it to the partition table?

Yes/No/Cancel? y
```

Ez helyreállítja a partíciót és annak tartalmát. Vegyük figyelembe, hogy a `rescue` csak olyan partíciókat tud helyreállítani, amelyekre fájlrendszer van telepítve. Az üres partíciókat nem ismeri fel.

ext2/3/4 partíciók átméretezése

A `parted` a partíciók méretének megváltoztatására is használható, hogy nagyobbá vagy kisebbé tegyük őket. Van azonban néhány megkötés:

- A partíció átméretezése során a partíciót nem szabad használni és le kell csatolni.
- Elég szabad helyre van szükség *a partíció után* ahhoz, hogy a kívánt méretre növelhessük azt.

A parancs a `resizepart`, amelyet a partíció száma és a partíció vége követ. Például, ha a következő partíciós táblával rendelkezünk:

```
Number  Start   End     Size    File system  Name     Flags
 1      1049kB  99.6MB  98.6MB  ext4         primary
 2      99.6MB  200MB   100MB   ext4
 3      200MB   300MB   99.6MB  ext4         primary
```

Ha az `resizepart` segítségével próbálnánk megnövelni az `1` partíciót, hibaüzenetet kapnánk, mivel az új méret esetén az `1` partíció átfedésben lenne a `2` partícióval. A `3` partíció azonban átméretezhető, mivel van szabad hely utána, amit a `print free` paranccsal ellenőrizhetünk:

```
(parted) print free
Model: Kingston DataTraveler 2.0 (scsi)
Disk /dev/sdb: 1999MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

 Number  Start   End     Size    File system  Name     Flags
         17.4kB  1049kB  1031kB  Free Space
 1       1049kB  99.6MB  98.6MB  ext4         primary
 2       99.6MB  200MB   100MB   ext4
 3       200MB   300MB   99.6MB  ext4         primary
         300MB   1999MB  1699MB  Free Space
```

Tehát az alábbi parancsot használhatjuk a partíció 3-ról 350 MB-re történő növeléséhez:

```
(parted) resizepart 3 350m

(parted) print
Model: Kingston DataTraveler 2.0 (scsi)
Disk /dev/sdb: 1999MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name     Flags
 1      1049kB  99.6MB  98.6MB  ext4         primary
 2      99.6MB  200MB   100MB   ext4
 3      200MB   350MB   150MB   ext4         primary
```

Ne feledjük, hogy az új végpontot a lemez elejétől számítva adja meg. Tehát, mivel a `3` partíció 300 MB-nál ért véget, most 350 MB-nál kell véget érnie.

A partíció átméretezése azonban csak egy része a feladatnak. A rajta található fájlrendszert is át kell méretezni. Az ext2/3/4 fájlrendszerek esetében ez a `resize2fs` paranccsal történik. A fenti példa esetében a 3. partíció még mindig a "régi" méretet mutatja, amikor felcsatoljuk:

```
$ df -h /dev/sdb3
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb3        88M  1.6M   80M   2% /media/carol/part3
```

A méret módosításához a `resize2fs DEVICE SIZE` parancs használható, ahol a `DEVICE` az átméretezni kívánt partíciónak felel meg, a `SIZE` pedig az új méret. Ha elhagyjuk a size paramétert, akkor a program a partíció teljes rendelkezésre álló helyét felhasználja. A méretváltoztatás előtt ajánlatos a partíciót leválasztani.

A fenti példában:

```
$ sudo resize2fs /dev/sdb3
resize2fs 1.44.6 (5-Mar-2019)
Resizing the filesystem on /dev/sdb3 to 146212 (1k) blocks.
The filesystem on /dev/sdb3 is now 146212 (1k) blocks long.

$ df -h /dev/sdb3
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb3       135M  1.6M  123M   2% /media/carol/part3
```

Egy partíció *zsugorításához* a folyamatot fordított sorrendben kell elvégezni. Először a fájlrendszert kell átméretezni az új, kisebb méretűre, majd magát a partíciót kell átméretezni a `parted` segítségével.

WARNING A partíciók zsugorításakor figyeljünk oda! Ha elrontjuk a sorrendet, elveszítjük az adatokat!

A példánkban:

```
# resize2fs /dev/sdb3 88m
resize2fs 1.44.6 (5-Mar-2019)
Resizing the filesystem on /dev/sdb3 to 90112 (1k) blocks.
The filesystem on /dev/sdb3 is now 90112 (1k) blocks long.

# parted /dev/sdb3
(parted) resizepart 3 300m
Warning: Shrinking a partition can cause data loss, are you sure
```

```
you want to continue?

Yes/No? y

(parted) print
Model: Kingston DataTraveler 2.0 (scsi)
Disk /dev/sdb: 1999MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name     Flags
 1      1049kB  99.6MB  98.6MB  ext4         primary
 2      99.6MB  200MB   100MB   ext4
 3      200MB   300MB   99.7MB  ext4         primary
```

TIP Az új méret megadása helyett használhatjuk a `resize2fs -M` paraméterét, hogy beállítsuk a fájlrendszer méretét, hogy az éppen elég nagy legyen a rajta lévő fájloknak.

Swap partíciók létrehozása

Linuxon a rendszer szükség szerint memóriaoldalakat tud cserélni a RAM-ból a lemezre, és ezeket egy külön helyen tárolja, amelyet általában a lemezen külön partícióként, *swap partíciónak* vagy egyszerűen *swapnak* neveznek. Ennek a partíciónak meghatározott típusúnak kell lennie, és egy megfelelő segédprogrammal (`mkswap`) kell beállítani, mielőtt használni lehetne.

A swap partíció létrehozásához az `fdisk` vagy `gdisk` használata esetén úgy kell eljárnunk, mintha egy hagyományos partíciót hoznánk létre, ahogyan azt már korábban megtettük. Az egyetlen különbség az, hogy a partíció típusát *Linux swap*-ra kell változtatni.

- Az `fdisk` esetén használjuk a `t` parancsot! Válasszuk ki a használni kívánt partíciót, és változtassuk meg a típusát `82`-re! Írjuk a változtatásokat a lemezre, és lépjünk ki a `w` paranccsal!
- A `gdisk`-nél a partíció típusának megváltoztatására szolgáló parancs szintén `t`, de a kód `8200`. Írjuk a változtatásokat a lemezre, és lépjünk ki a `w` paranccsal!

A `parted` használata esetén a partíciót swap partícióként kell azonosítani a létrehozás során, ehhez csak a `linux-swap` fájlrendszer típust kell használnunk. Például egy 500 MB-os swap partíció létrehozásához a következő parancsot kell kiadni, a lemezen 300 MB-ról indulva:

```
(parted) mkpart primary linux-swap 301m 800m
```

Miután a partíciót létrehoztuk és megfelelően azonosítottuk, csak használjuk az `mkswap` parancsot, majd a használni kívánt partíciót reprezentáló eszközt, például:

```
# mkswap /dev/sda2
```

Ha engedélyezni szeretnénk a swapot ezen a partíción, használjuk a `swapon` parancsot, amelyet az eszköz neve követ:

```
# swapon /dev/sda2
```

Hasonlóképpen, a `swapoff`, amelyet az eszköz neve követ, letiltja a swapot az adott eszközön.

A Linux támogatja a swap *fájlok* használatát is a partíciók helyett. Csak hozzunk létre egy üres, a kívánt méretű fájlt a `dd` segítségével, majd használjuk az `mkswap` és a `swapon` parancsokat, és ezt a fájlt használjuk célként.

A következő parancsok létrehoznak egy 1 GB-os, nullákkal teli `myswap` nevű fájlt az aktuális mappában, majd beállítják és engedélyezik, mint swap fájlt.

Swap fájl létrehozása:

```
$ dd if=/dev/zero of=myswap bs=1M count=1024
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 7.49254 s, 143 MB/s
```

`if=` a bemeneti (input) fájl, a fájlba írandó adatok forrása. Ebben az esetben ez a `/dev/zero` eszköz, amely annyi `NULL` karaktert ad, amennyit kérünk. `of=` a kimeneti fájl, a létrehozandó fájl. `bs=` az adatblokkok mérete, itt megabájtban van megadva, és `count=` a kimenetre írandó blokkok száma. 1024 darab, egyenként 1 MB-os blokk 1 GB-nak felel meg.

```
# mkswap myswap
Setting up swapspace version 1, size = 1024 MiB (1073737728 bytes)
no label, UUID=49c53bc4-c4b1-4a8b-a613-8f42cb275b2b

# swapon myswap
```

A fenti parancsok használatával ez a swap-fájl csak az aktuális rendszermunkamenet alatt lesz használható. Ha a gép újraindul, a fájl továbbra is elérhető lesz, de nem töltődik be

automatikusan. Ezt automatizálhatjuk az új swap fájl hozzáadásával az `/etc/fstab` állományhoz, amiről egy későbbi leckében lesz szó.

TIP

Mind az `mkswap`, mind a `swapon` panaszkodik, ha a swap fájlnak nem biztonságosak a jogosultságai. Az ajánlott fájl-engedély flag a `0600`. A tulajdonosnak és a csoportnak is `root`.-nak kell lennie.

Gyakorló feladatok

1. Milyen particionálási sémát kell használni egy 3 TB-os merevlemez három 1 GB-os partícióra történő particionálásához? Miért?

2. Hogyan tudjuk meg a `gdisk` segítségével, hogy mennyi szabad hely van a lemezen?

3. Mi lenne a parancs egy ext3 fájlrendszer létrehozására, amely előtte ellenőrzi a rossz blokkokat, `MyDisk` címkével és egy véletlenszerű UUID-vel, a `/dev/sdc1` eszközön?

4. A `parted` használatával milyen paranccsal hozhatunk létre egy 300 MB-os ext4 partíciót, 500 MB-ról indulva a lemezen?

5. Képzeljük el, hogy van 2 partíciónk, az egyik a `/dev/sda1`, a másik a `/dev/sda2` lemezen, mindkettő 20 GB méretű. Hogyan használhatjuk őket egyetlen Btrfs fájlrendszeren úgy, hogy az egyik partíció tartalma automatikusan tükrözve legyen a másikra, mint egy RAID1 beállításnál? Mekkora lesz a fájlrendszer?

Gondolkodtató feladatok

1. Tételezzük fel, hogy van egy 2 GB-os lemezünk MBR partíciós táblával és az alábbi layouttal:

```
Disk /dev/sdb: 1.9 GiB, 1998631936 bytes, 3903578 sectors
Disk model: DataTraveler 2.0
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x31a83a48

Device     Boot   Start     End Sectors  Size Id Type
/dev/sdb1          2048 1050623 1048576  512M 83 Linux
/dev/sdb3       2099200 3147775 1048576  512M 83 Linux
```

   Létre tudunk hozni egy 600 MB-os partíciót rajta? Miért?

2. A `/dev/sdc` lemezen van egy 1 GB-os első partíció, amely körülbelül 256 MB fájlt tartalmaz. A `parted` segítségével hogyan lehet ezt úgy összezsugorítani, hogy éppen elég hely legyen a fájloknak?

3. Képzeljük el, hogy van egy lemezünk a `/dev/sdb` címen, és szeretnénk létrehozni egy 1 GB-os swap partíciót a lemez elején. Tehát a `parted` segítségével létrehozzuk a partíciót az `mkpart primary linux-swap 0 1024M` paranccsal. Ezután a `swapon /dev/sdb1`-vel engedélyezzük a swapot ezen a partíción, de a következő hibaüzenetet kapjuk:

   `swapon: /dev/sdb1: read swap header failed`

   Mi volt a baj?

4. A lecke során kipróbáltunk néhány parancsot a `parted` programban, de véletlenül töröltük a 3. partíciót a merevlemezen. Tudjuk, hogy ez egy 250 MB-os UEFI partíció és egy 4 GB-os swap partíció után következett, és 10 GB-os volt. Melyik paranccsal tudjuk helyreállítani?

5. Képzeljük el, hogy van egy 4 GB-os kihasználatlan partíció a `/dev/sda3` lemezen. Az `fdisk` használatával milyen műveletsorozatot kellene végrehajtanunk, hogy aktív swap partícióvá

alakítsuk?

Összefoglalás

Ebben a leckében megtanultuk:

- Hogyan hozzunk létre egy MBR partíciós táblát egy lemezen az `fdisk` segítségével, és hogyan használjuk azt partíciók létrehozására és törlésére.
- Hogyan hozzunk létre MBR partíciós táblát egy lemezen az `gdisk` segítségével, és hogyan használjuk partíciók létrehozására és törlésére.
- Hogyan hozzunk létre ext2, ext3, ext4, XFS, VFAT és exFAT partíciókat.
- Hogyan használjuk a `parted`-et partíciók létrehozására, törlésére és helyreállítására MBR és GPT lemezeken.
- Hogyan használjuk az ext2, ext3, ext4 és Brts partíciók átméretezését.
- Hogyan hozzunk létre, állítsunk be és aktiváljunk swap partíciókat és swap fájlokat.

A leckében az alábbi parancsokról volt szó:

- `fdisk`
- `gdisk`
- `mkfs.ext2`, `mkfs.ext3`, `mkfs.ext4`, `mkfs.xfs`, `mkfs.vfat` és `mkfs.exfat`
- `parted`
- `btrfs`
- `mkswap`
- `swapon` és `swapoff`

Válaszok a gyakorló feladatokra

1. Milyen particionálási sémát kell használni egy 3 TB-os merevlemez három 1 GB-os partícióra történő particionálásához? Miért?

   GPT-t, mivel az MBR csak 2 TB méretű lemezeket támogat.

2. Hogyan tudjuk meg a `gdisk` segítségével, hogy mennyi szabad hely van a lemezen?

   Használjuk a `p`-t (print). A teljes szabad terület a partíciós tábla előtti utolsó információs sorban jelenik meg.

3. Mi lenne a parancs egy ext3 fájlrendszer létrehozására, amely előtte ellenőrzi a rossz blokkokat, `MyDisk` címkével és egy véletlenszerű UUID-vel, a `/dev/sdc1` eszközön?

   A parancs: `mkfs.ext3 -c -L MyDisk -U random /dev/sdc1`. Használhatjuk az `mke2fs -t ext3`-t alternatívaként az `mkfs.ext3` helyett.

4. A `parted` használatával milyen paranccsal hozhatunk létre egy 300 MB-os ext4 partíciót, 500 MB-ról indulva a lemezen?

   Használjuk az `mkpart primary ext4 500m 800m` parancsot. Ne feledjük, hogy nekünk kell létrehozni a fájlrendszert az `mkfs.ext4` segítségével, mivel a parted ezt nem teszi meg!

5. Képzeljük el, hogy van 2 partíciónk, az egyik a `/dev/sda1`, a másik a `/dev/sda2` lemezen, mindkettő 20 GB méretű. Hogyan használhatjuk őket egyetlen Btrfs fájlrendszeren úgy, hogy az egyik partíció tartalma automatikusan tükrözve legyen a másikra, mint egy RAID1 beállításnál? Mekkora lesz a fájlrendszer?

   Használjuk az `mkfs.btrfs /dev/sda1 /dev/sdb1 -m raid1` parancsot. A fájlrendszer 20 GB méretű lesz, mivel az egyik partíció a másik tükrözöttjeként fog viselkedni.

Válaszok a gondolkodtató feladatokra

1. Tételezzük fel, hogy van egy 2 GB-os lemezünk MBR partíciós táblával és az alábbi layouttal:

```
Disk /dev/sdb: 1.9 GiB, 1998631936 bytes, 3903578 sectors
Disk model: DataTraveler 2.0
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x31a83a48

Device     Boot   Start     End Sectors  Size Id Type
/dev/sdb1          2048 1050623 1048576  512M 83 Linux
/dev/sdb3       2099200 3147775 1048576  512M 83 Linux
```

   Létre tudunk hozni egy 600 MB-os partíciót rajta? Miért?

   Nem, mivel nincs elég összefüggő terület. Az első jel, hogy valami “nem működik”, az eszközök listája: van `/dev/sdb1` és `/dev/sdb3`, de nincs `/dev/sdb2`. Tehát valami hiányzik.

   Ezután meg kell néznünk, hogy hol végződik a partíció és hol kezdődik a másik. Az első partíció a `1050623` szektornál ér véget, a 2-es partíció pedig `2099200`-nál kezdődik. Ez egy 1048577 szektoros “gap”. Szektoronként 512 bájttal ez 536.871.424 bájt. Ha elosztjuk 1024-el, 524.288 kilobájtot kapunk. Újra osztjuk 1024-el és… 512 MB. Ez a “gap” mérete.

   Ha a lemez 2 GB, akkor a 3. partíció után legfeljebb további 512 MB-ot kapunk. Még akkor is, ha összesen kb. 1 GB van felosztatlanul, a legnagyobb összefüggő blokk 512 MB. Tehát nincs hely egy 600 MB-os partíciónak.

2. A `/dev/sdc` lemezen van egy 1 GB-os első partíció, amely körülbelül 256 MB fájlt tartalmaz. A `parted` segítségével hogyan lehet ezt úgy összezsugorítani, hogy éppen elég hely legyen a fájloknak?

   Ez egy több részből álló művelet. Először is a `resize2fs` segítségével zsugorítani kell a fájlrendszert. Az új méret közvetlen megadása helyett használhatjuk az `-M` paramétert, így az “elég nagy” lesz. Így: `resize2fs -M /dev/sdc1`.

   Ezután magát a partíciót méretezzük át a `resizepart` segítségével. Mivel ez az első partíció, feltételezhetjük, hogy nulláról indul és 241 MB-nál ér véget. A parancs tehát a következő: `resizepart 1 241M`.

3. Képzeljük el, hogy van egy lemezünk a `/dev/sdb` címen, és szeretnénk létrehozni egy 1 GB-os swap partíciót a lemez elején. Tehát a `parted` segítségével létrehozzuk a partíciót az `mkpart primary linux-swap 0 1024M` paranccsal. Ezután a `swapon /dev/sdb1`-vel engedélyezzük a swapot ezen a partíción, de a következő hibaüzenetet kapjuk:

   ```
   swapon: /dev/sdb1: read swap header failed
   ```

   Mi volt a baj?

   A jó típusú partíciót hoztuk létre (`linux-swap`), de ne feledjük, hogy az `mkpart` *nem hoz létre fájlrendszert*. Elfelejtettük beállítani a partíciót swap-tárhelyként az `mkswap` programmal, mielőtt használtuk volna.

4. A lecke során kipróbáltunk néhány parancsot a `parted` programban, de véletlenül töröltük a 3. partíciót a merevlemezen. Tudjuk, hogy ez egy 250 MB-os UEFI partíció és egy 4 GB-os swap partíció után következett, és 10 GB-os volt. Melyik paranccsal tudjuk helyreállítani?

   Nem kell pánikba esni, minden szükséges információ megvan a partíció helyreállításához, csak használjuk a `rescue` parancsot és számoljunk! Előtte 250 MB + 4.096 MB (4*1024) volt, tehát a kezdőpontnak 4346 MB körül kell lennie. Plusz 10,240 MB (10*1024) méret, a végére 14,586 MB-nak kellene lennie. Tehát a `rescue 4346m 14586m`-nek működnie kell. Lehet, hogy a lemez geometriájától függően egy kis “lazaságot” kell adnunk a mentéshez, egy kicsit korán kezdve és egy kicsit későn befejezve.

5. Képzeljük el, hogy van egy 4 GB-os kihasználatlan partíció a `/dev/sda3` lemezen. Az `fdisk` használatával milyen műveletsorozatot kellene végrehajtanunk, hogy aktív swap partícióvá alakítsuk?

   Először változtassuk a partíció típusát az alábbira: “Linux Swap” (82), mentsük a módosításokat és lépjünk ki. Ezután használjuk az `mkswap`-ot, hogy beállítsuk a partíciót swap területnek. Ezután a `swapon` segítségével engedélyezzük.

104.2 A fájlrendszerek integritásának karbantartása

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 101, Objective 104.2

Súlyozás

2

Kulcsfontosságú ismeretek

- A fájlrendszerek integritásának ellenőrzése.
- A szabad hely és az inode-ok figyelése.
- Egyszerű fájlrendszer-problémák javítása.

A használt fájlok, kifejezések és segédprogramok listája

- `du`
- `df`
- `fsck`
- `e2fsck`
- `mke2fs`
- `tune2fs`
- `xfs_repair`
- `xfs_fsr`
- `xfs_db`

104.2 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	104 Eszközök, Linux fájlrendszerek, Fájlrendszer-hierarchia szabvány
Fejezet:	104.2 A fájlrendszerek integritásának karbantartása
Lecke:	1/1

Bevezetés

A modern Linux fájlrendszerek naplóznak. Ez azt jelenti, hogy minden műveletet a végrehajtás előtt egy belső naplóban (a *journal*) regisztrálnak. Ha a művelet megszakad egy rendszerhiba miatt (például kernelpánik, áramkimaradás stb.), a napló ellenőrzésével rekonstruálható, így elkerülhető a fájlrendszer sérülése és az adatvesztés.

Ez nagymértékben csökkenti a kézi fájlrendszer-ellenőrzések szükségességét, de még mindig szükség lehet rájuk. Az ehhez használt eszközök (és a megfelelő paraméterek) ismerete jelentheti a különbséget az otthoni vacsora a családdal vagy az egész éjszakás munka a szerverszobában között.

Ebben a leckében a fájlrendszer használatának ellenőrzésére, működésének optimalizálására, valamint a sérülések ellenőrzésére és javítására rendelkezésre álló eszközökről lesz szó.

Lemezhasználat ellenőrzése

Két parancsot használhatunk annak ellenőrzésére, hogy mennyi helyet használunk, és mennyi van még a fájlrendszeren. Az első a `du`, ami a “disk usage” (lemezhasználat) rövidítése.

A `du` rekurzív jellegű. Legegyszerűbb formájában a parancs egyszerűen megmutatja, hogy hány 1 kilobájtos blokkot használ az aktuális mappa és annak összes almappája:

```
$ du
4816    .
```

Ez nem túl nagy segítség, így kérhetünk “ember által olvasható” kimenetet a `-h` paraméterrel:

```
$ du -h
4.8M    .
```

Alapértelmezés szerint a `du` csak a mappák használatának számát mutatja (figyelembe véve az összes fájlt és a benne lévő almappákat). Ha a mappában lévő összes fájlra vonatkozó egyedi számot szeretnénk megjeleníteni, használjuk az `-a` paramétert:

```
$ du -ah
432K    ./geminoid.jpg
508K    ./Linear_B_Hero.jpg
468K    ./LG-G8S-ThinQ-Mirror-White.jpg
656K    ./LG-G8S-ThinQ-Range.jpg
60K ./Stranger3_Titulo.png
108K    ./Baidu_Banho.jpg
324K    ./Xiaomi_Mimoji.png
284K    ./Mi_CC_9e.jpg
96K ./Mimoji_Comparativo.jpg
32K ./Xiaomi FCC.jpg
484K    ./geminoid2.jpg
108K    ./Mimoji_Abre.jpg
88K ./Mi8_Hero.jpg
832K    ./Tablet_Linear_B.jpg
332K    ./Mimoji_Comparativo.png
4.8M    .
```

Az alapértelmezett viselkedés az, hogy minden almappa használatát mutatja, majd az aktuális mappa teljes használatát, *az almappákkal együtt*:

```
$ du -h
4.8M    ./Temp
6.0M    .
```

A fenti példában láthatjuk, hogy a `Temp` almappa 4,8 MB-ot foglal el, az aktuális mappa pedig a `Temp` mappával együtt 6,0 MB-ot. De mennyi helyet foglalnak el az aktuális mappában lévő *fájlok*, az almappák nélkül? Erre ott van az `-S` paraméter:

```
$ du -Sh
4.8M    ./Temp
1.3M    .
```

TIP Ne feledjük, hogy a parancssori paraméterek a nagy- és kisbetűket megkülönböztetik: az `-s` nem azonos az `-S`-sel.

Ha meg akarjuk tartani ezt a különbséget az aktuális mappában lévő fájlok és az almappák által használt hely között, de a végén szeretnénk egy végösszeget, akkor hozzáadhatjuk a `-c` paramétert:

```
$ du -Shc
4.8M    ./Temp
1.3M    .
6.0M    total
```

A `du` kimenetének "mélységét" a `-d N` paraméterrel szabályozhatjuk, ahol az `N` a szinteket írja le. Ha például a `-d 1` paramétert használjuk, akkor az aktuális mappát és annak almappáit fogja megjeleníteni, de azok almappáit nem.

Nézzük meg a különbséget! `-d` nélkül:

```
$ du -h
216K    ./somedir/anotherdir
224K    ./somedir
232K    .
```

És a mélység egy szintre történő limitálásával (`-d 1`):

```
$ du -h -d1
224K    ./somedir
```

```
232K    .
```

Vegyük figyelembe, hogy még ha az `anotherdir` nem is jelenik meg, a méretét akkor is figyelembe vesszük.

Előfordulhat, hogy bizonyos típusú fájlokat ki akarunk zárni a számolásból, amit az `--exclude="PATTERN"` paranccsal tehetünk meg, ahol a `PATTERN` az a minta, amellyel szemben a keresést végezni akarjuk. Tekintsük ezt a mappát:

```
$ du -ah
124K    ./ASM68K.EXE
2.0M    ./Contra.bin
36K ./fixheadr.exe
4.0K    ./README.txt
2.1M    ./Contra_NEW.bin
4.0K    ./Built.bat
8.0K    ./Contra_Main.asm
4.2M    .
```

Most használjuk az `--exclude`-t, hogy minden `.bin` kiterjesztésű fájlt kiszűrjünk:

```
$ du -ah --exclude="*.bin"
124K    ./ASM68K.EXE
36K ./fixheadr.exe
4.0K    ./README.txt
4.0K    ./Built.bat
8.0K    ./Contra_Main.asm
180K    .
```

Vegyük figyelembe, hogy a végösszeg már nem tükrözi a kizárt fájlok méretét.

Szabad hely keresése

A `du` a fájlok szintjén működik. Van egy másik parancs is, amely a fájlrendszer szintjén mutatja meg a lemezhasználatot, és azt, hogy mennyi hely áll rendelkezésre. Ez a parancs a `df`.

Az `df` parancs egy listát ad a rendszeren lévő összes elérhető (már csatlakoztatott) fájlrendszerről, beleértve a teljes méretüket, a felhasznált hely nagyságát, a rendelkezésre álló hely nagyságát, a használat százalékos arányát és a csatlakoztatás helyét:

```
$ df
Filesystem     1K-blocks      Used Available Use% Mounted on
udev             2943068         0   2943068   0% /dev
tmpfs             595892      2496    593396   1% /run
/dev/sda1      110722904  25600600  79454800  25% /
tmpfs            2979440    951208   2028232  32% /dev/shm
tmpfs               5120         0      5120   0% /run/lock
tmpfs            2979440         0   2979440   0% /sys/fs/cgroup
tmpfs             595888        24    595864   1% /run/user/119
tmpfs             595888       116    595772   1% /run/user/1000
/dev/sdb1          89111      1550     80824   2% /media/carol/part1
/dev/sdb3          83187      1550     75330   3% /media/carol/part3
/dev/sdb2          90827      1921     82045   3% /media/carol/part2
/dev/sdc1      312570036 233740356  78829680  75% /media/carol/Samsung Externo
```

A méret 1 KB-os blokkokban való megjelenítése azonban nem túl felhasználóbarát. A `du`-hoz hasonlóan a `-h` paramétereket is hozzáadhatjuk, hogy egy “ember által olvashatóbb” kimenetet kapjunk:

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            2.9G     0  2.9G   0% /dev
tmpfs           582M  2.5M  580M   1% /run
/dev/sda1       106G   25G   76G  25% /
tmpfs           2.9G  930M  2.0G  32% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           2.9G     0  2.9G   0% /sys/fs/cgroup
tmpfs           582M   24K  582M   1% /run/user/119
tmpfs           582M  116K  582M   1% /run/user/1000
/dev/sdb1        88M  1.6M   79M   2% /media/carol/part1
/dev/sdb3        82M  1.6M   74M   3% /media/carol/part3
/dev/sdb2        89M  1.9M   81M   3% /media/carol/part2
/dev/sdc1       299G  223G   76G  75% /media/carol/Samsung Externo
```

Használhatjuk az `-i` paramétert is, ha blokkok helyett a használt/elérhető inode-okat szeretnénk megjeleníteni:

```
$ df -i
Filesystem      Inodes  IUsed   IFree IUse% Mounted on
udev            737142    547  736595    1% /dev
tmpfs           745218    908  744310    1% /run
```

```
/dev/sda6      6766592 307153 6459439    5% /
tmpfs           745218    215  745003    1% /dev/shm
tmpfs           745218      4  745214    1% /run/lock
tmpfs           745218     18  745200    1% /sys/fs/cgroup
/dev/sda1        62464    355   62109    1% /boot
tmpfs           745218     43  745175    1% /run/user/1000
```

Hasznos paraméter még a `-T`, amely az egyes fájlrendszerek típusát is kiírja:

```
$ df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
udev           devtmpfs  2.9G     0  2.9G   0% /dev
tmpfs          tmpfs     582M  2.5M  580M   1% /run
/dev/sda1      ext4      106G   25G   76G  25% /
tmpfs          tmpfs     2.9G  930M  2.0G  32% /dev/shm
tmpfs          tmpfs     5.0M     0  5.0M   0% /run/lock
tmpfs          tmpfs     2.9G     0  2.9G   0% /sys/fs/cgroup
tmpfs          tmpfs     582M   24K  582M   1% /run/user/119
tmpfs          tmpfs     582M  116K  582M   1% /run/user/1000
/dev/sdb1      ext4       88M  1.6M   79M   2% /media/carol/part1
/dev/sdb3      ext4       82M  1.6M   74M   3% /media/carol/part3
/dev/sdb2      ext4       89M  1.9M   81M   3% /media/carol/part2
/dev/sdc1      fuseblk   299G  223G   76G  75% /media/carol/Samsung Externo
```

A fájlrendszer típusának ismeretében szűrhetjük a kimenetet. Csak az adott típusú fájlrendszereket jeleníthetjük meg a `-t TYPE` paranccsal, vagy kizárhatjuk az adott típusú fájlrendszereket az `-x TYPE` paranccsal, mint az alábbi példákban.

A `tmpfs` fájlrendszerek kizárása:

```
$ df -hx tmpfs
Filesystem      Size  Used Avail Use% Mounted on
udev            2.9G     0  2.9G   0% /dev
/dev/sda1       106G   25G   76G  25% /
/dev/sdb1        88M  1.6M   79M   2% /media/carol/part1
/dev/sdb3        82M  1.6M   74M   3% /media/carol/part3
/dev/sdb2        89M  1.9M   81M   3% /media/carol/part2
/dev/sdc1       299G  223G   76G  75% /media/carol/Samsung Externo
```

Csak az `ext4` fájlrendszerek megjelenítése:

```
$ df -ht ext4
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       106G   25G   76G  25% /
/dev/sdb1        88M  1.6M   79M   2% /media/carol/part1
/dev/sdb3        82M  1.6M   74M   3% /media/carol/part3
/dev/sdb2        89M  1.9M   81M   3% /media/carol/part2
```

Az `df` kimenete is testreszabható, kiválasztva, hogy mi jelenjen meg és milyen sorrendben, a `--output=` paraméter használatával, amelyet a megjeleníteni kívánt mezők vesszővel elválasztott listája követ. Néhány a rendelkezésre álló mezők közül:

`source`
: A fájlrendszernek megfelelő eszköz.

`fstype`
: A fájlrendszer típusa.

`size`
: A fájlrendszer teljes mérete.

`used`
: Mennyi hely van használatban.

`avail`
: Mennyi hely elérhető.

`pcent`
: A használat százalékos aránya.

`target`
: Hová van a fájlrendszer felcsatolva (mount point).

Ha olyan kimenetet szeretnénk, amely megmutatja a célt, a forrást, a típust és a használatot, használhatjuk a következőt:

```
$ df -h --output=target,source,fstype,pcent
Mounted on                  Filesystem     Type     Use%
/dev                        udev           devtmpfs   0%
/run                        tmpfs          tmpfs      1%
/                           /dev/sda1      ext4      25%
/dev/shm                    tmpfs          tmpfs     32%
```

```
/run/lock                     tmpfs          tmpfs      0%
/sys/fs/cgroup                tmpfs          tmpfs      0%
/run/user/119                 tmpfs          tmpfs      1%
/run/user/1000                tmpfs          tmpfs      1%
/media/carol/part1            /dev/sdb1      ext4       2%
/media/carol/part3            /dev/sdb3      ext4       3%
/media/carol/part2            /dev/sdb2      ext4       3%
/media/carol/Samsung Externo  /dev/sdc1      fuseblk   75%
```

A `df` az inode információk ellenőrzésére is használható, ha a következő mezőket adjuk át az `--output=`-nak:

`itotal`

Az összes inode száma a fájlrendszerben.

`iused`

A használt inode-ok száma a fájlrendszerben.

`iavail`

Az elérhető inode-ok száma a fájlrendszerben.

`ipcent`

Az inode-ok százalékos aránya a fájlrendszerben.

Például:

```
$ df --output=source,fstype,itotal,iused,ipcent
Filesystem     Type      Inodes  IUsed IUse%
udev           devtmpfs  735764    593    1%
tmpfs          tmpfs     744858   1048    1%
/dev/sda1      ext4     7069696 318651    5%
tmpfs          tmpfs     744858    222    1%
tmpfs          tmpfs     744858      3    1%
tmpfs          tmpfs     744858     18    1%
tmpfs          tmpfs     744858     22    1%
tmpfs          tmpfs     744858     40    1%
```

ext2, ext3 és ext4 fájlrendszerek karbantartása

A fájlrendszer hibáinak ellenőrzésére (és remélhetőleg javítására) a Linux biztosítja az `fsck` segédprogramot (gondoljunk a “filesystem check” szóra, és soha nem fogjuk elfelejteni a nevét).

Legegyszerűbb formájában az `fsck` parancsot használva hívhatjuk meg, amelyet az ellenőrizni kívánt fájlrendszer helye követ:

```
# fsck /dev/sdb1
fsck from util-linux 2.33.1
e2fsck 1.44.6 (5-Mar-2019)
DT_2GB: clean, 20/121920 files, 369880/487680 blocks
```

WARNING Soha ne futtassuk az `fsck`-t (vagy a kapcsolódó segédprogramokat) felcsatolt fájlrendszeren. Ha ez mégis megtörténik, az adatok elveszhetnek.

Maga az `fsck` nem ellenőrzi a fájlrendszert, csupán meghívja a fájlrendszer típusának megfelelő segédprogramot. A fenti példában, mivel a fájlrendszer típusa nem volt megadva, az `fsck` alapértelmezés szerint ext2/3/4 fájlrendszert feltételezett, és az `e2fsck`-t hívta.

Egy fájlrendszer megadásához használjuk a `-t` opciót, amelyet a fájlrendszer neve követ, mint például az `fsck -t vfat /dev/sdc` esetben. Alternatívaként közvetlenül is meghívhatunk egy fájlrendszer-specifikus segédprogramot, például az `fsck.msdos`-t FAT fájlrendszerekhez.

TIP Írjuk be kétszer az `fsck`-t, majd utána a Tab billentyűt kétszer, hogy megjelenjen a rendszer összes kapcsolódó parancsának listája.

Az `fsck` tud néhány parancssori argumentumot is kezelni. A leggyakoribbak közül néhány:

`-A`

Ez ellenőrzi a fájlrendszereket az `/etc/fstab` listában.

`-C`

Megjelenít egy progress bart a fájlrendszer ellenőrzése során. Jelenleg csak ext2/3/4 fájlrendszereken működik.

`-N`

Ez kiírja, hogy mit tenne és kilép anélkül, hogy ténylegesen ellenőrizné a fájlrendszert.

`-R`

Az `-A`-val együtt használva kihagyja a root fájlrendszer ellenőrzését.

`-V`

Verbose mód, a szokásosnál több információt ír ki működés közben. Hasznos hibakeresési célokra.

Az ext2, ext3 és ext4 fájlrendszerek speciális segédprogramja az `e2fsck`, más néven `fsck.ext2`, `fsck.ext3` és `fsck.ext4` (ez a három csak link az `e2fsck`-re). Alapértelmezés szerint interaktív módban fut: ha fájlrendszer hibát talál, megáll, és megkérdezi a felhasználót, hogy mit tegyen. A felhasználónak be kell írnia az `y`-t a probléma kijavításához, az `n`-t a probléma kijavítatlanul hagyásához, vagy az `a`-t az aktuális és az összes további probléma kijavításához.

Természetesen a terminál előtt ülve várni, hogy az `e2fsck` megkérdezze, mit kell tennie, nem túl produktív időtöltés, különösen, ha egy nagy fájlrendszerrel van dolgunk. Vannak tehát olyan opciók, amelyek hatására az `e2fsck` nem interaktív módban fut:

`-p`
: Ez megkísérli automatikusan kijavítani a talált hibákat. Ha olyan hibát talál, amely a rendszergazda beavatkozását igényli, az `e2fsck` megadja a probléma leírását és kilép.

`-y`
: Ez minden kérdésre `y` (yes) választ ad.

`-n`
: A `-y` ellentettje. Amellett, hogy minden kérdésre `n` (no) választ ad, a fájlrendszer csak olvashatóan lesz csatolva, így nem módosítható.

`-f`
: Kényszeríti az `e2fsck`-t, hogy akkor is ellenőrizzen egy fájlrendszert, ha az “clean”-ként van megjelölve, azaz helyesen lett lecsatolva.

Egy ext fájlrendszer finomhangolása

Az ext2, ext3 és ext4 fájlrendszerek számos olyan paraméterrel rendelkeznek, amelyeket a rendszergazda beállíthat, vagy “hangolhat”, hogy jobban megfeleljenek a rendszer igényeinek. Az ezen paraméterek megjelenítésére vagy módosítására használt segédprogram neve `tune2fs`.

Ha egy adott fájlrendszer aktuális paramétereit szeretnénk látni, használjuk a `-l` paramétert, amelyet a partíciót reprezentáló eszköz követ. Az alábbi példa egy gép első lemezének (`/dev/sda1`) első partíciójára vonatkozó parancs kimenetét mutatja:

```
# tune2fs -l /dev/sda1
tune2fs 1.44.6 (5-Mar-2019)
Filesystem volume name:   <none>
Last mounted on:          /
Filesystem UUID:          6e2c12e3-472d-4bac-a257-c49ac07f3761
Filesystem magic number:  0xEF53
```

```
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype
needs_recovery extent 64bit flex_bg sparse_super large_file huge_file dir_nlink extra_isize
metadata_csum
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              7069696
Block count:              28255605
Reserved block count:     1412780
Free blocks:              23007462
Free inodes:              6801648
First block:              0
Block size:               4096
Fragment size:            4096
Group descriptor size:    64
Reserved GDT blocks:      1024
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         8192
Inode blocks per group:   512
Flex block group size:    16
Filesystem created:       Mon Jun 17 13:49:59 2019
Last mount time:          Fri Jun 28 21:14:38 2019
Last write time:          Mon Jun 17 13:53:39 2019
Mount count:              8
Maximum mount count:      -1
Last checked:             Mon Jun 17 13:49:59 2019
Check interval:           0 (<none>)
Lifetime writes:          20 GB
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:           256
Required extra isize:     32
Desired extra isize:      32
Journal inode:            8
First orphan inode:       5117383
Default directory hash:   half_md4
Directory Hash Seed:      fa95a22a-a119-4667-a73e-78f77af6172f
Journal backup:           inode blocks
Checksum type:            crc32c
```

```
Checksum:                 0xe084fe23
```

Az ext fájlrendszereknek van egy *mount counts* (csatolási szám) értéke. A számláló minden egyes alkalommal, amikor a fájlrendszert csatlakoztatjuk, 1-gyel nő, és amikor egy küszöbértéket (a *maximum mount count*) elérünk, a rendszer a következő indításkor automatikusan ellenőrzésre kerül az `e2fsck` programmal.

A maximális csatolási számot a `-c N` paraméterrel lehet beállítani, ahol `N` az a szám, ahányszor a fájlrendszert ellenőrzés nélkül lehet csatolni. A `-C N` paraméter a rendszer mountolásának számát az `N` értékre állítja be. Vegyük figyelembe, hogy a parancssori paraméterek nagy- és kisbetű-érzékenyek, így a `-c` nem azonos a `-C` paraméterrel.

Lehetőség van az ellenőrzések közötti időintervallum meghatározására is a `-i` paraméterrel, amelyet egy szám és a `d` betűk követnek a napok, `m` a hónapok és `y` az évek esetében. Például az `-i 10d` a fájlrendszert a következő újraindításkor 10 naponta ellenőrzi. A funkció kikapcsolásához használjuk a nulla értéket.

Az `-L` a fájlrendszer címkéjének beállítására használható. Ez a címke legfeljebb 16 karakter lehet. Az `-U` paraméter beállítja a fájlrendszer UUID-jét, amely egy 128 bites hexadecimális szám. A fenti példában az UUID a `6e2c12e3-472d-4bac-a257-c49ac07f3761`. A fájlrendszer csatlakoztatásánál mind a címke, mind az UUID használható az eszköz neve helyett (például `/dev/sda1`).

A `-e BEHAVIOUR` kapcsoló meghatározza a kernel viselkedését, ha fájlrendszer hibát talál. Három lehetséges viselkedési mód van:

`continue`
: Normálisan folytatja a végrehajtást.

`remount-ro`
: Csak olvasható módon csatolja a fájlrendszert.

`panic`
: Kernel pánikot okoz.

Az alapértelmezett viselkedés a `continue`. A `remount-ro` hasznos lehet adatérzékeny (data-sensitive) alkalmazásokban, mivel azonnal leállítja a lemezre történő írást, elkerülve ezzel a további potenciális hibákat.

Az ext3 fájlrendszerek alapvetően ext2 fájlrendszerek egy naplóval (journal). A `tune2fs` segítségével hozzáadhatunk egy naplót egy ext2 fájlrendszerhez és így átalakíthatjuk ext3-ra. Az

eljárás egyszerű, csak adjuk át a `tune2fs`-nek a `-j` paramétert, majd a fájlrendszert tartalmazó eszközt:

```
# tune2fs -j /dev/sda1
```

Ezután az átalakított fájlrendszer mountolásakor ne felejtsük el beállítani a típusát `ext3`-ra, hogy a napló használható legyen.

Amikor naplózott fájlrendszerekkel dolgozunk, a `-J` paraméter lehetővé teszi, hogy extra paramétereket használjunk néhány napló beállításához, mint például `-J size=` a napló méretének beállításához (megabájtban), `-J location=` a napló tárolási helyének megadásához (vagy egy adott blokk, vagy egy adott pozíció a lemezen olyan utótagokkal, mint `M` vagy `G`), vagy akár a napló külső eszközre helyezéséhez a `-J device=` paraméterrel.

Több paramétert is megadhatunk egyszerre, ha vesszővel választjuk el őket. Például: `-J size=10,location=100M,device=/dev/sdb1` egy 10 MB-os naplót hoz létre a 100 MB-os pozícióban a `/dev/sdb1` eszközön.

WARNING A `tune2fs` rendelkezik egy “brute force” opcióval, az `-f`-el, amely hiba esetén is kényszeríti a művelet befejezésére. Mondanom sem kell, hogy ezt csak rendkívül óvatosan szabad használni.

XFS fájlrendszerek karbantartása

XFS fájlrendszerek esetén az `fsck` megfelelője az `xfs_repair`. Ha azt gyanítjuk, hogy valami baj van a fájlrendszerrel, először is át kell vizsgálni, hogy nem sérült-e meg.

Ezt úgy lehet megtenni, hogy az `xfs_repair`-nek átadjuk az `-n` paramétert, majd a fájlrendszert tartalmazó eszközt. Az `-n` paraméter azt jelenti, hogy “no modify”: a fájlrendszer ellenőrizve lesz, a hibákat jelzi, de javítás nem történik:

```
# xfs_repair -n /dev/sdb1
Phase 1 - find and verify superblock...
Phase 2 - using internal log
        - zero log...
        - scan filesystem freespace and inode maps...
        - found root inode chunk
Phase 3 - for each AG...
        - scan (but do not clear) agi unlinked lists...
        - process known inodes and perform inode discovery...
        - agno = 0
```

```
        - agno = 1
        - agno = 2
        - agno = 3
        - process newly discovered inodes...
Phase 4 - check for duplicate blocks...
        - setting up duplicate extent list...
        - check for inodes claiming duplicate blocks...
        - agno = 1
        - agno = 3
        - agno = 0
        - agno = 2
No modify flag set, skipping phase 5
Phase 6 - check inode connectivity...
        - traversing filesystem ...
        - traversal finished ...
        - moving disconnected inodes to lost+found ...
Phase 7 - verify link counts...
No modify flag set, skipping filesystem flush and exiting.
```

Ha hibát találunk, akkor a `-n` paraméter nélkül hagyhatjuk jóvá a javítást, így: `xfs_repair /dev/sdb1`.

Az `xfs_repair` számos parancssori kapcsolót ismer, íme néhány:

`-l LOGDEV` és **`-r RTDEV`**

Ezekre akkor van szükség, ha a fájlrendszer külső log és valós idejű szekciókkal rendelkezik. Ebben az esetben a `LOGDEV` és `RTDEV` helyett a megfelelő eszközöket kell megadni.

`-m N`

Az `xfs_repair` memóriahasználatának `N` megabájtra való korlátozására szolgál, ami hasznos lehet a szerverek beállításainál. A man oldal alapján alapértelmezés szerint az `xfs_repair` a rendszer fizikai RAM memóriahasználatának 75%-áig skálázza a memóriahasználatot.

`-d`

A “dangerous” (veszélyes) mód lehetővé teszi a csak olvashatóként felcsatolt fájlrendszerek javítását.

`-v`

Talán ki is tudjuk találni: verbose mód. Minden egyes alkalommal, amikor ezt a paramétert használjuk, a “verbosity” növekszik (például a `-v -v` több információt ír ki, mint a `-v`).

Vegyük figyelembe, hogy az `xfs_repair` nem képes javítani a “dirty” naplóval rendelkező

fájlrendszereket. A hibás naplót a `-L` paraméterrel “nullázhatjuk”, de ne feledjük, hogy ez csak az utolsó lehetőség, mivel ez a fájlrendszer sérüléséhez és adatvesztéshez vezethet.

Egy XFS fájlrendszer hibakeresésére az `xfs_db` segédprogram használható, mint például az `xfs_db /dev/sdb1`. Ez leginkább a fájlrendszer különböző elemeinek és paramétereinek vizsgálatára szolgál.

Ennek a segédprogramnak van egy interaktív promptja, mint a `parted`, sok belső paranccsal. Egy súgórendszer is rendelkezésre áll: írjuk be a `help` parancsot az összes parancs listájának megtekintéséhez, majd a `help` parancsot a parancs neve után, hogy további információkat kapjunk a parancsról.

Egy másik hasznos segédprogram az `xfs_fsr`, amely egy XFS fájlrendszer átszervezésére (“defragmentálására” (töredezettségmentesítés)) használható. Ha minden további argumentum nélkül futtatjuk, két órán keresztül fut, és megpróbálja defragmentálni az összes csatlakoztatott, írható XFS fájlrendszert, amely szerepel az `/etc/mtab/` fájlban. Lehet, hogy ezt a segédprogramot a Linux-disztribúció csomagkezelőjével kell telepíteni, mivel előfordulhat, hogy nem része az alapértelmezett telepítésnek. További információt a megfelelő man oldalon találhatunk.

Gyakorló feladatok

1. A `du` használatával hogyan tudjuk ellenőrizni, hogy mennyi helyet használnak az aktuális mappában lévő fájlok?

2. A `df` használatával listázzuk ki az információkat minden ext4 fájlrendszerről, a kimenetek a következő mezőket tartalmazzák sorrendben: eszköz, csatolási pont, inode-ok száma, rendelkezésre álló inode-ok száma, szabad hely százalékos aránya.

3. Mi a parancs az `e2fsck` nem interaktív módban való futtatásához a `/dev/sdc1`, miközben automatikusan megpróbálja kijavítani a legtöbb hibát?

4. Tegyük fel, hogy a `/dev/sdb1` egy ext2 fájlrendszer. Hogyan lehet átalakítani ext3-ra, és ezzel egyidejűleg visszaállítani a mount countot, és a címkéjét `UserData`-ra változtatni?

5. Hogyan lehet ellenőrizni a hibákat egy XFS fájlrendszeren, anélkül, hogy javítanánk a talált sérüléseket?

Gondolkodtató feladatok

1. Tegyük fel, hogy van egy ext4 fájlrendszerünk a `/dev/sda1` lemezen a következő paraméterekkel, amelyeket a `tune2fs` programmal kaptunk:

   ```
   Mount count:              8
   Maximum mount count:      -1
   ```

   Mi fog történni a következő rendszerindításkor, ha a `tune2fs -c 9 /dev/sda1` parancsot adjuk ki?

2. Nézzük az alábbi `du -h` kimenetet:

   ```
   $ du -h
   216K    ./somedir/anotherdir
   224K    ./somedir
   232K    .
   ```

   Mennyi helyet foglalnak el csak az aktuális mappában lévő fájlok? Hogyan írhatnánk át a parancsot, hogy egyértelműbben mutassa ezt az információt?

3. Mi történik az ext2 fájlrendszerrel (`/dev/sdb1`), ha az alábbi parancsot adjuk ki?

   ```
   # tune2fs -j /dev/sdb1 -J device=/dev/sdc1 -i 30d
   ```

4. Hogyan ellenőrizhetjük a hibákat egy XFS fájlrendszeren a `/dev/sda1`-en, amelynek van egy naplórészlege a `/dev/sdc1`-en, anélkül, hogy bármilyen javítást végeznénk?

5. Mi a különbség a `-T` és a `-t` paraméterek között az `df` esetében?

Összefoglalás

Ebben a leckében megtanultuk:

- Hogyan ellenőrizhetjük a használt és szabad helyet egy fájlrendszerben.
- Hogyan alakítsuk az `df` kimenetét az igényeinknek megfelelően.
- Hogyan ellenőrizhetjük és javíthatjuk a fájlrendszer integritását az `fsck` és az `e2fsck` segítségével.
- Hogyan finomhangolhatunk egy ext fájlrendszert a `tune2fs` segítségével.
- Hogyan ellenőrizhetjük és javíthatjuk az XFS fájlrendszereket az `xfs_repair` segítségével.

A következő parancsokról volt szó a leckében:

`du`
: A fájlrendszerben használt lemezterület nagyságának megtekintése.

`df`
: A fájlrendszeren rendelkezésre álló (szabad) lemezterület megtekintése.

`fsck`
: A fájlrendszert ellenőrző-javító segédprogram.

`e2fsck`
: A kiterjesztett (ext2/3/4) fájlrendszerekre jellemző fájlrendszert ellenőrző-javító segédprogram.

`tune2fs`
: A kiterjesztett (ext2/3/4) fájlrendszer paramétereinek módosítása.

`xfs_repair`
: Az `fsck` megfelelője XFS fájlrendszerekhez.

`xfs_db`
: Ez a segédprogram az XFS fájlrendszer különböző paramétereinek megtekintésére szolgál.

Válaszok a gyakorló feladatokra

1. A `du` használatával hogyan tudjuk ellenőrizni, hogy mennyi helyet használnak az aktuális mappában lévő fájlok?

   Először is, használjuk az `-S` paramétert az aktuális mappa kimenetének almappáktól való szétválasztására. Ezután használjuk a `-d 0` paramétert, hogy a kimeneti mélységet nullára korlátozzuk, ami azt jelenti, hogy "nincs almappa". Ne feledkezzünk meg a `-h` paraméterről, hogy a kimenetet "ember által olvasható" formátumban kapjuk meg:

   ```
   $ du -S -h -d 0
   ```

   vagy

   ```
   $ du -Shd 0
   ```

2. A `df` használatával listázzuk ki az információkat minden ext4 fájlrendszerről, a kimenetek a következő mezőket tartalmazzák sorrendben: eszköz, csatolási pont, inode-ok száma, rendelkezésre álló inode-ok száma, szabad hely százalékos aránya.

   A fájlrendszereket a `-t` opcióval szűrhetjük, amelyet a fájlrendszer neve követ. A szükséges kimenethez használjuk az `--output=source,target,itotal,iavail,pcent` parancsot. A válasz tehát a következő:

   ```
   $ df -t ext4 --output=source,target,itotal,iavail,pcent
   ```

3. Mi a parancs az `e2fsck` futtatásához a `/dev/sdc1` lemezen nem interaktív módban, miközben automatikusan megpróbálja kijavítani a legtöbb hibát?

   A paraméter, amivel automatikusan megpróbálja kijavítani a legtöbb hibát a `-p`. Tehát a válasz:

   ```
   # e2fsck -p /dev/sdc1
   ```

4. Tegyük fel, hogy a `/dev/sdb1` egy ext2 fájlrendszer. Hogyan lehet átalakítani ext3-ra, és ezzel egyidejűleg visszaállítani a mount countot, és a címkéjét `UserData`-ra változtatni?

   Ne feledjük, hogy egy ext2 fájlrendszer ext3-ra való átalakítása csak egy napló hozzáadásának kérdése, amit a `-j` paraméterrel tehetünk meg. A mount count visszaállításához használjuk a `-C 0` paramétert. A címke megváltoztatásához használjuk a `-L UserData` paramétert. A helyes

válasz a következő:

```
# tune2fs -j -C 0 -L UserData /dev/sdb1
```

5. Hogyan lehet ellenőrizni a hibákat egy XFS fájlrendszeren, anélkül, hogy javítanánk a talált sérüléseket?

   Használjuk az `-n` paramétert, mint például `xfs -n`, majd utána a megfelelő eszközt.

Válaszok a gondolkodtató feladatokra

1. Tegyük fel, hogy van egy ext4 fájlrendszerünk a `/dev/sda1` lemezen a következő paraméterekkel, amelyeket a `tune2fs` programmal kaptunk:

   ```
   Mount count:              8
   Maximum mount count:      -1
   ```

   Mi fog történni a következő rendszerindításkor, ha a `tune2fs -c 9 /dev/sda1` parancsot adjuk ki?

   A parancs a fájlrendszer maximális mount count-ját 9-re állítja be. Mivel ez jelenleg 8, a következő rendszerindításakor a fájlrendszer ellenőrzése megtörténik.

2. Nézzük az alábbi `du -h` kimenetet:

   ```
   $ du -h
   216K    ./somedir/anotherdir
   224K    ./somedir
   232K    .
   ```

   Mennyi helyet foglalnak el csak az aktuális mappában lévő fájlok? Hogyan írhatnánk át a parancsot, hogy egyértelműbben mutassa ezt az információt?

   Az összesen 232 K-ból 224 K-t a `somedir` almappa és annak almappái használnak. Ezeket leszámítva tehát 8 K-t foglalnak el az aktuális mappában lévő fájlok. Ez az információ még világosabban megjeleníthető az `-S` paraméter használatával, amely a mappákat szétválasztja a számolásnál.

3. Mi történik az ext2 fájlrendszerrel (`/dev/sdb1`), ha az alábbi parancsot adjuk ki?

   ```
   # tune2fs -j /dev/sdb1 -J device=/dev/sdc1 -i 30d
   ```

   Egy napló kerül hozzáadásra a `/dev/sdb1`-hoz, ezzel ext3-ra konvertálja. A napló a `/dev/sdc1` eszközön lesz tárolva és a fájlrendszer minden 30. napon ellenőrizve lesz.

4. Hogyan ellenőrizhetjük a hibákat egy XFS fájlrendszeren a `/dev/sda1`-en, amelynek van egy naplórészlege a `/dev/sdc1`-en, anélkül, hogy bármilyen javítást végeznénk?

   Használjuk az `xfs_repair`-t a `-l /dev/sdc1`-vel a naplórészletet tartalmazó eszköz

megjelölésére, és `-n`-t a változtatások elkerülése érdekében.

```
# xfs_repair -l /dev/sdc1 -n
```

5. Mi a különbség a `-T` és a `-t` paraméterek között az `df` esetében?

   A `-T` paraméter az egyes fájlrendszerek típusát is tartalmazza a `df` kimenetén. A `-t` egy szűrő, és csak a megadott típusú fájlrendszereket jeleníti meg a kimeneten, az összes többit kizárva.

104.3 Fájlrendszerek csatolása és leválasztása

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 101, Objective 104.3

Súlyozás

3

Kulcsfontosságú ismeretek

- Fájlrendszerek manuális csatlakoztatása és leválasztása.
- A fájlrendszerek mountolásának beállítása indításkor.
- Felhasználó által csatlakoztatható cserélhető fájlrendszerek konfigurálása.
- Címkék és UUID-k használata a fájlrendszerek azonosításához és csatolásához.
- A systemd mount egységek ismerete.

A használt fájlok, kifejezések és segédprogramok listája

- `/etc/fstab`
- `/media/`
- `mount`
- `umount`
- `blkid`
- `lsblk`

104.3 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	104 Eszközök, Linux fájlrendszerek, Fájlrendszer-hierarchia szabvány
Fejezet:	104.3 Fájlrendszerek csatolása és leválasztása
Lecke:	1/1

Bevezetés

Eddigre megtanultuk, hogyan kell lemezeket particionálni és hogyan kell azokon fájlrendszereket létrehozni és karbantartani. Mielőtt azonban egy fájlrendszert elérhetnénk Linuxon, *csatolni kell* (mount).

Ez azt jelenti, hogy a fájlrendszert a rendszer mappastruktúrájának egy adott pontjához, az úgynevezett *csatolási ponthoz* (mount point) csatolja. A fájlrendszereket manuálisan vagy automatikusan lehet csatolni, és ennek számos módja van. Ezek közül néhányat meg fogunk ismerni ebben a leckében.

Fájlrendszerek csatolása és leválasztása

Egy fájlrendszer manuális felcsatolásához a parancs a `mount` és a szintaxisa az alábbi:

```
mount -t TYPE DEVICE MOUNTPOINT
```

Ahol:

`TYPE`
: A csatolandó fájlrendszer típusa (pl. ext4, btrfs, exfat, stb.).

`DEVICE`
: A fájlrendszert tartalmazó partíció neve (pl. `/dev/sdb1`)

`MOUNTPOINT`
: Ahová a fájlrendszer csatolásra kerül. A mounted-on mappának nem kell üresnek lennie, de muszáj léteznie. A benne lévő fájlok azonban név szerint elérhetetlenek lesznek, amíg a fájlrendszer fel van csatolva.

Például egy exFAT fájlrendszert tartalmazó USB flash meghajtó csatolásához a `/dev/sdb1` lemezen található `flash` nevű mappába a home mappánk alatt, használhatjuk a következőt:

```
# mount -t exfat /dev/sdb1 ~/flash/
```

TIP Sok Linux rendszer a Bash shell-t használja, és ezeken a rendszereken a csatolási pont elérési útvonalában lévő tilde (~) a jelenlegi felhasználó home mappájának rövidítése. Ha például az aktuális felhasználó neve `john`, akkor a helyére `/home/john` kerül.

A csatolás után a fájlrendszer tartalma a `~/flash` mappában lesz elérhető:

```
$  ls -lh ~/flash/
total 469M
-rwxrwxrwx 1 root root 454M jul 19 09:49 lineage-16.0-20190711-MOD-quark.zip
-rwxrwxrwx 1 root root  16M jul 19 09:44 twrp-3.2.3-mod_4-quark.img
```

Felcsatolt fájlrendszerek listázása

Ha csak a `mount` parancsot írjuk be, akkor a rendszerre jelenleg csatlakoztatott összes fájlrendszer listáját kapjuk meg. Ez a lista elég nagy lehet, mert a rendszerhez csatlakoztatott lemezek mellett számos futásidejű fájlrendszer is található a memóriában, amelyek különböző célokat szolgálnak. A kimenet szűréséhez használhatjuk a `-t` paramétert, hogy csak a megfelelő típusú fájlrendszereket listázza ki, mint az alábbiakban:

```
# mount -t ext4
/dev/sda1 on / type ext4 (rw,noatime,errors=remount-ro)
```

Több fájlrendszert is megadhatunk egyszerre, ha vesszővel választjuk el őket:

```
# mount -t ext4,fuseblk
/dev/sda1 on / type ext4 (rw,noatime,errors=remount-ro)
/dev/sdb1 on /home/carol/flash type fuseblk
(rw,nosuid,nodev,relatime,user_id=0,group_id=0,default_permissions,allow_other,blksize=4096)
[DT_8GB]
```

A fenti példák kimenete a következő formátumban írható le:

```
SOURCE on TARGET type TYPE OPTIONS
```

Ahol a `SOURCE` az a partíció, ami tartalmazza a fájlrendszert, a `TARGET` a mappa, ahová csatolva van, a `TYPE` a fájlrendszer típusa és az `OPTIONS` azon opciók, amelyeket a csatoláskor átadunk a `mount` parancsnak.

További parancssori paraméterek

Számos parancssori paraméter használható a `mount` paranccsal. Néhány a leggyakrabban használtak közül:

`-a`

Ez a `/etc/fstab` fájlban felsorolt összes fájlrendszert csatolja (erről bővebben a következő részben).

`-o` vagy `--options`

Ez egy vesszővel elválasztott *mount options* (csatolási opciók) listát ad át a mount parancsnak, amely megváltoztathatja a fájlrendszer csatolásának módját. Ezekről is szó lesz az `/etc/fstab` mellett.

`-r` vagy `-ro`

A fájlrendszert csak olvashatóként csatolja.

`-w` vagy `-rw`

A fájlrendszerként írhatóként csatolja.

Egy fájlrendszer leválasztásához használjuk az `umount` parancsot, amelyet az eszköz neve vagy a csatolási pont követ. A fenti példát figyelembe véve az alábbi parancsok felcserélhetők:

```
# umount /dev/sdb1
```

```
# umount ~/flash
```

A `umount` parancssori paraméterei közül néhány:

`-a`

Az `/etc/fstab`-ban listázott összes fájlrendszert leválasztja.

`-f`

Kikényszeríti a fájlrendszer leválasztását. Ez hasznos lehet, ha egy távoli fájlrendszert csatlakoztattunk, amely elérhetetlenné vált.

`-r`

Ha a fájlrendszert nem lehet leválasztani, ez megpróbálja csak olvashatóvá tenni.

Megnyitott fájlok kezelése

Egy fájlrendszer leválasztásakor előfordulhat, hogy hibaüzenetet kapunk, amely szerint a `target is busy`. Ez akkor fordul elő, ha a fájlrendszeren bármilyen fájl meg van nyitva. Azonban nem biztos, hogy azonnal nyilvánvaló, hogy hol található egy megnyitott fájl, vagy hogy mi fér hozzá a fájlrendszerhez.

Ilyen esetekben használhatjuk az `lsof` parancsot, amelyet a fájlrendszert tartalmazó eszköz neve követ, hogy megnézzük az azt elérő folyamatok listáját és a megnyitott fájlokat. Például:

```
# umount /dev/sdb1
umount: /media/carol/External_Drive: target is busy.

# lsof /dev/sdb1
COMMAND  PID   USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
evince  3135 carol   16r   REG   8,17 21881768 5195 /media/carol/External_Drive/Documents/E-
Books/MagPi40.pdf
```

A `COMMAND` a fájlt megnyitó futtatható program neve, a `PID` pedig a folyamat száma. A `NAME` a megnyitott fájl neve. A fenti példában a `MagPi40.pdf` fájlt az `evince` program (egy PDF-megjelenítő) nyitotta meg. Ha bezárjuk a programot, akkor lecsatolhatjuk a fájlrendszert.

NOTE

Mielőtt az `lsof` kimenet megjelenik, a `GNOME` felhasználók egy figyelmeztető üzenetet láthatnak a terminál ablakban.

```
lsof: WARNING: can't stat() fuse.gvfsd-fuse file system /run/user/1000/gvfs
```

```
Output information may be incomplete.
```

Az `lsof` megpróbálja feldolgozni az összes csatlakoztatott fájlrendszert. Ez a figyelmeztető üzenet azért jelenik meg, mert az `lsof` egy `GNOME` virtuális fájlrendszerrel (GVFS) találkozott. Ez egy speciális esete a felhasználói térben lévő fájlrendszernek (`FUSE`). Hídként működik a `GNOME`, annak API-jai és a kernel között. Senki - még a root sem - nem férhet hozzá egy ilyen fájlrendszerhez, kivéve a tulajdonosát, aki csatolta (ebben az esetben a `GNOME`). Ezt a figyelmeztetést figyelmen kívül hagyhatjuk.

Hová csatoljunk?

A fájlrendszert bárhová csatolhatjuk, ahová csak szeretnénk. Van azonban néhány jó gyakorlat, amelyet érdemes betartani a rendszer adminisztrációjának megkönnyítése érdekében.

Hagyományosan az `/mnt` volt az a mappa, amely alá minden külső eszközt csatlakoztattak, és itt számos előre konfigurált “rögzítési pont” (anchor point) létezett a gyakori eszközökhöz, mint például a CD-ROM meghajtók (`/mnt/cdrom`) és a floppy lemezek (`/mnt/floppy`).

Ezt felváltotta a `/media`, amely mostantól a felhasználó által eltávolítható adathordozók (pl. külső lemezek, USB flash meghajtók, memóriakártya-olvasók stb.) alapértelmezett csatolási pontja.

A legtöbb modern Linux disztribúcióban és asztali környezetben a cserélhető eszközök automatikusan a `/media/USER/LABEL` címszó alá lesznek csatlakoztatva, amikor a rendszerhez csatlakoznak, ahol a `USER` a felhasználónév, a `LABEL` pedig az eszköz címkéje. Például a `john` felhasználó által csatlakoztatott `FlashDrive` feliratú USB flash meghajtó a `/media/john/FlashDrive/` alá lesz csatlakoztatva. Ennek kezelése az asztali környezettől függően eltérő.

Ennek ellenére, amikor *manuálisan* kell csatolni egy fájlrendszert, jó gyakorlat, hogy az `/mnt` alá csatoljuk.

Fájlrendszerek csatolása indításkor

Az `/etc/fstab` fájl tartalmazza a csatolható fájlrendszerek leírását. Ez egy szöveges fájl, ahol minden sor egy-egy csatolandó fájlrendszert ír le, soronként hat mezővel, a következő sorrendben:

```
FILESYSTEM MOUNTPOINT TYPE OPTIONS DUMP PASS
```

Ahol:

`FILESYSTEM`

A csatolandó fájlrendszert tartalmazó eszköz. Az eszköz helyett megadhatjuk a partíció UUID-jét vagy címkéjét is, amiről később még lesz szó.

`MOUNTPOINT`

Ahová a fájlrendszer csatolásra kerül.

`TYPE`

A fájlrendszer típusa.

`OPTIONS`

A `mount`-nak átadandó csatolási opciók.

`DUMP`

Megjelöli, hogy az ext2, ext3 vagy ext4 fájlrendszereket a `dump` parancs figyelembe vegye-e a mentés során. Általában nulla, ami azt jelenti, hogy ezeket figyelmen kívül kell hagyni.

`PASS`

Ha nem nulla, meghatározza a sorrendet, amelyben a fájlrendszereket a rendszerindításkor ellenőrizni fogja. Általában nulla.

Például egy gép első lemezének első partíciója a következőképpen írható le:

```
/dev/sda1  /  ext4  noatime,errors
```

Az `OPTIONS` kapcsolási opciók egy vesszővel elválasztott paraméterlista, amely lehet általános vagy fájlrendszer-specifikus. Az általánosak között vannak:

`atime` és **`noatime`**

Alapértelmezés szerint minden egyes fájl olvasásakor frissül a hozzáférési idő információja. Ennek kikapcsolása (a `noatime` kapcsolóval) felgyorsíthatja a lemez I/O-t. Ne tévesszük össze ezt a módosítási idővel, amely minden alkalommal frissül, amikor egy fájlba írunk.

`auto` és **`noauto`**

A fájlrendszer automatikusan csatlakoztatható-e (vagy sem) a `mount -a` paranccsal.

`defaults`

Ez az alábbi opciókat adja át a `mount`-nak: `rw`, `suid`, `dev`, `exec`, `auto`, `nouser` és `async`.

`dev` és `nodev`

A csatlakoztatott fájlrendszerben lévő karakteres vagy blokkos eszközöket kell-e értelmezni.

`exec` és `noexec`

Engedélyezi vagy megtagadja a bináris programok végrehajtását a fájlrendszerben.

`user` és `nouser`

Lehetővé teszi (vagy nem teszi) egy közönséges felhasználó számára a fájlrendszer csatolását.

`group`

Lehetővé teszi egy felhasználó számára a fájlrendszer csatolását, ha a felhasználó ugyanahhoz a csoporthoz tartozik, amelyhez az azt tartalmazó eszköz tulajdonosa.

`owner`

Lehetővé teszi egy felhasználó számára egy fájlrendszer csatolását, ha a felhasználó birtokolja az azt tartalmazó eszközt.

`suid` és `nosuid`

Engedélyezi vagy nem engedélyezi a SETUID és SETGID bitek hatását.

`ro` és `rw`

Csak olvasható vagy írható fájlrendszer csatolása.

`remount`

Ez megkísérli egy már csatolt fájlrendszer újbóli csatolását. Ezt nem az `/etc/fstab` állományban használjuk, hanem a `mount -o` paramétereként. Például a már csatolt `/dev/sdb1` partíció újracsatolásához a `mount -o remount,ro /dev/sdb1` parancsot használhatjuk. Újracsatoláskor nem kell megadni a fájlrendszer típusát, csak az eszköz nevét *vagy* a csatolási pontot.

`sync` és `async`

A fájlrendszerrel végzett összes I/O művelet szinkron vagy aszinkron legyen-e. Az `async` általában az alapértelmezett. A `mount` man oldala arra figyelmeztet, hogy a `sync` használata korlátozott számú írási ciklusú adathordozókon (mint a flash meghajtók vagy memóriakártyák) lerövidítheti az eszköz élettartamát.

UUID-k és címkék használata

A csatolandó fájlrendszert tartalmazó eszköz nevének megadása problémákat okozhat. Előfordulhat, hogy ugyanazt az eszköznevet más eszközhöz rendelik, attól függően, hogy mikor

vagy hol csatlakoztatták a rendszerhez. Például egy `/dev/sdb1` USB flash meghajtó a `/dev/sdc1` nevet kaphatja, ha egy másik portra csatlakoztatjuk, vagy egy másik flash meghajtó után.

Ezt elkerülhetjük, ha megadjuk a kötet címkéjét vagy UUID-jét (*Universally Unique Identifier*). Mindkettő a fájlrendszer létrehozásakor kerül megadásra, és nem változik, kivéve, ha a fájlrendszert megsemmisítik, vagy manuálisan új címkét vagy UUID-t rendelnek hozzá.

Az `lsblk` paranccsal lekérdezhetünk információkat egy fájlrendszerről, és megtudhatjuk a hozzá tartozó címkét és UUID-t. Ehhez használjuk az `-f` paramétert, amelyet az eszköz neve követ:

```
$ lsblk -f /dev/sda1
NAME FSTYPE LABEL UUID                                 FSAVAIL FSUSE% MOUNTPOINT
sda1 ext4         6e2c12e3-472d-4bac-a257-c49ac07f3761   64,9G    33% /
```

Az egyes oszlopok jelentése a következő:

`NAME`
: A fájlrendszert tartalmazó eszköz neve.

`FSTYPE`
: A fájlrendszer típusa.

`LABEL`
: A fájlrendszer címkéje.

`UUID`
: A fájlrendszerhez tartozó Universally Unique Identifier (UUID).

`FSAVAIL`
: Mennyi szabad hely van a fájlrendszerben.

`FSUSE%`
: A fájlrendszer kihasználtságának százalékos aránya.

`MOUNTPOINT`
: Hová van csatolva a fájlrendszer.

Az `/etc/fstab'-ban egy eszköz megadható az UUID azonosítójával az `UUID=` opcióval, amelyet az UUID követ, vagy a `LABEL=` opcióval, amelyet a címke követ. Tehát a következő helyett:

```
/dev/sda1  /  ext4  noatime,errors
```

Használhatjuk:

```
UUID=6e2c12e3-472d-4bac-a257-c49ac07f3761  /  ext4  noatime,errors
```

Vagy, ha a lemez címkéje `homedisk`:

```
LABEL=homedisk  /home ext4  defaults
```

Ugyanez a szintaxis használható a `mount` paranccsal is. Az eszköz neve helyett az UUID-t vagy a címkét kell megadni. Például egy külső NTFS lemez csatlakoztatásához, amelynek UUID-je `56C11DCC5D2E1334` az `/mnt/external` lemezre, a parancs a következő:

```
$ mount -t ntfs UUID=56C11DCC5D2E1334 /mnt/external
```

Lemezek csatolása a Systemd-vel

A *systemd* az init folyamat, az első folyamat, amely sok Linux disztribúcióban fut. Felelős más folyamatok indításáért, a szolgáltatások elindításáért és a rendszer indításáért. Sok más feladat mellett a systemd a fájlrendszerek csatolásának (és automount-olásának) kezelésére is használható.

A systemd ezen funkciójának használatához létre kell hozni egy *mount unit* nevű konfigurációs fájlt. Minden csatolandó kötet saját mount unitot kap, és ezeket a `/etc/systemd/system/` állományban kell elhelyezni.

A mount unitok egyszerű szöveges fájlok, amelyeknek a kiterjesztése `.mount`. Az alapvető formátum az alábbiakban látható:

```
[Unit]
Description=

[Mount]
What=
Where=
Type=
Options=
```

```
[Install]
WantedBy=
```

`Description=`

A mount unit rövid leírása, például `Mounts the backup disk`.

`What=`

Mit kell csatolni. A kötetet a `/dev/disk/by-uuid/VOL_UUID` formában kell megadni, ahol a `VOL_UUID` a kötet UUID-je.

`Where=`

Az a teljes elérési útvonal, ahová a kötetet csatlakoztatni kell.

`Type=`

A fájlrendszer típusa.

`Options=`

A csatolási opciók, amelyeket át szeretnénk adni, ezek ugyanazok, amelyeket a `mount` paranccsal vagy az `/etc/fstab`-ban használunk.

`WantedBy=`

Függőségkezelésre szolgál. Ebben az esetben a `multi-user.target`-et fogjuk használni, ami azt jelenti, hogy amikor a rendszer többfelhasználós környezetbe bootol (normál boot), az egység fel lesz csatolva.

A külső lemezre vonatkozó korábbi példánkat a következőképpen írhatnánk le:

```
[Unit]
Description=External data disk

[Mount]
What=/dev/disk/by-uuid/56C11DCC5D2E1334
Where=/mnt/external
Type=ntfs
Options=defaults

[Install]
WantedBy=multi-user.target
```

De még nem végeztünk. A helyes működéshez a csatolási egységnek *meg kell* egyeznie a csatolási

pont nevével. Ebben az esetben a csatolási pont `/mnt/external`, így a fájl neve `mnt-external.mount` kell, hogy legyen.

Ezután újra kell indítani a systemd daemont a `systemctl` paranccsal, és el kell indítani az egységet:

```
# systemctl daemon-reload
# systemctl start mnt-external.mount
```

A külső lemez tartalmának most már elérhetőnek kell lennie az `/mnt/external` helyen. A csatolás állapotát az alábbi `systemctl status mnt-external.mount` paranccsal ellenőrizhetjük:

```
# systemctl status mnt-external.mount
● mnt-external.mount - External data disk
   Loaded: loaded (/etc/systemd/system/mnt-external.mount; disabled; vendor pres
   Active: active (mounted) since Mon 2019-08-19 22:27:02 -03; 14s ago
    Where: /mnt/external
     What: /dev/sdb1
    Tasks: 0 (limit: 4915)
   Memory: 128.0K
   CGroup: /system.slice/mnt-external.mount

ago 19 22:27:02 pop-os systemd[1]: Mounting External data disk...
ago 19 22:27:02 pop-os systemd[1]: Mounted External data disk.
```

A `systemctl start mnt-external.mount` parancs csak az aktuális munkamenethez engedélyezi az egységet. Ha minden indításkor engedélyezni akarjuk, akkor a `start` parancsot helyettesítsük az `enable` parancsra:

```
# systemctl enable mnt-external.mount
```

Mount Unit automatikus csatolása

A csatolási egységek automatikusan csatolása, amikor a csatolási ponthoz hozzáférnek. Ehhez szükség van egy `.automount` fájlra az egységet leíró `.mount` fájl mellett. Az alapvető formátum a következő:

```
[Unit]
Description=
```

```
[Automount]
Where=

[Install]
WantedBy=multi-user.target
```

A korábbiakhoz hasonlóan a `Description=` a fájl rövid leírása, a `Where=` pedig a csatolási pont. Például egy `.automount` fájl az előző példánkhoz a következő lenne:

```
[Unit]
Description=Automount for the external data disk

[Automount]
Where=/mnt/external

[Install]
WantedBy=multi-user.target
```

Mentsük el a fájlt a csatolási ponttal azonos névvel (ebben az esetben `mnt-external.automount`), töltsük újra a systemd-t és indítsuk el az egységet:

```
# systemctl daemon-reload
# systemctl start mnt-external.automount
```

Mostantól minden alkalommal, amikor az `/mnt/external` mappához hozzáférünk, a lemez fel lesz csatolva. Mint korábban, az automount engedélyezéséhez minden indításkor a következőt kell használni:

```
# systemctl enable mnt-external.automount
```

Gyakorló feladatok

1. A `mount` használatával hogyan lehet egy `/dev/sdc1` fájlrendszert `/dev/sdc1`-re `/mnt/external` csak olvashatóan csatlakoztatni, a `noatime` és az `async` opciókkal?

2. A `/dev/sdd2` fájlrendszer leválasztásakor a `target is busy` hibaüzenetet kapjuk. Hogyan lehet megtudni, hogy a fájlrendszeren mely fájlok vannak megnyitva, és milyen folyamatok nyitották meg őket?

3. Tekintsük a következő bejegyzést az `/etc/fstab'-ban: `/dev/sdb1 /data ext4 noatime,noauto,async`. Ez a fájlrendszer fel lesz csatolva, ha a `mount -a` parancsot adjuk ki? Miért?

4. Hogyan lehet megtudni a `/dev/sdb1` alatti fájlrendszer UUID-jét?

5. Hogyan használhatjuk a `mount` parancsot a `6e2c12e3-472d-4bac-a257-c49ac07f3761` UUID-vel rendelkező, a `/mnt/data` címre csatolt exFAT fájlrendszer olvashatóvá tételéhez?

6. Hogyan kaphatunk egy listát az összes `ext3` és `ntfs` fájlrendszerről, amely jelenleg fel van csatolva a rendszerre?

Gondolkodtató feladatok

1. Tekintsük a következő bejegyzést az `/etc/fstab'-ban: `/dev/sdc1 /backup ext4 noatime,nouser,async`. Tudja-e egy felhasználó csatlakoztatni ezt a fájlrendszert a `mount /backup` paranccsal? Miért?

2. Tekintsünk egy távoli fájlrendszert, amely az `/mnt/server` címre van felcsatolva, és a hálózati kapcsolat megszűnése miatt elérhetetlenné vált. Hogyan lehetne kikényszeríteni, hogy leválasztásra kerüljön, vagy, ha ez nem lehetséges, csak olvashatóként csatolja?

3. Írjunk egy `/etc/fstab` bejegyzést, amely egy btrfs kötetet csatlakoztatna a `Backup` címkével az `/mnt/backup` lemezre, alapértelmezett beállításokkal és anélkül, hogy engedélyezné a bináris programok végrehajtását!

4. Tekintsük a következő systemd mount egységet:

```
[Unit]
Description=External data disk

[Mount]
What=/dev/disk/by-uuid/56C11DCC5D2E1334
Where=/mnt/external
Type=ntfs
Options=defaults

[Install]
WantedBy=multi-user.target
```

   - Mi lenne egyenértékű `/etc/fstab` bejegyzés ehhez a fájlrendszerhez?

5. Mi legyen a fenti egység fájlneve, hogy a systemd használni tudja? Hol kell elhelyezni?

Összefoglalás

Ebben a leckében megtanultuk, hogyan lehet manuálisan vagy automatikusan csatolni és leválasztani a fájlrendszereket. Néhány parancsot és fogalmat megismertünk:

- `mount` (csatol egy eszközt egy helyhez)
- `umount` (leválaszt egy eszközt)
- `lsof` (kilistázza a fájlrendszerhez hozzáférő processzeket)
- `/mnt` és `/media` mappák
- `/etc/fstab`
- `lsblk` (kilistázza egy fájlrendszer típusát és UUID-jét)
- Hogyan csatoljunk egy fájlrendszert az UUID vagy a címke segítségével.
- Hogyan csatoljunk egy fájlrendszert a systemd mount egységek használatával.
- Hogyan lehet egy fájlrendszert automatikusan csatolni a systemd mount egységek használatával.

Válaszok a gyakorló feladatokra

1. A `mount` használatával hogyan lehet egy `/dev/sdc1` fájlrendszert `/dev/sdc1`-re `/mnt/external` csak olvashatóan csatlakoztatni, a `noatime` és az `async` opciókkal?

   ```
   # mount -t ext4 -o noatime,async,ro /dev/sdc1 /mnt/external
   ```

2. A `/dev/sdd2` fájlrendszer leválasztásakor a `target is busy` hibaüzenetet kapjuk. Hogyan lehet megtudni, hogy a fájlrendszeren mely fájlok vannak megnyitva, és milyen folyamatok nyitották meg őket?

   Használjuk az `lsof`-ot az eszköz nevével:

   ```
   $ lsof /dev/sdd2
   ```

3. Tekintsük a következő bejegyzést az `/etc/fstab'-ban: `/dev/sdb1 /data ext4 noatime,noauto,async`. Ez a fájlrendszer fel lesz csatolva, ha a `mount -a` parancsot adjuk ki? Miért?

   Nem lesz csatolva. A kulcs a `noauto` paraméter, ami azt jelenti, hogy ez a bejegyzés figyelmen kívül lesz hagyva a `mount -a` által.

4. Hogyan lehet megtudni a `/dev/sdb1` alatti fájlrendszer UUID-jét?

   Használjuk az `lsblk -f`-et a fájlrendszer nevével:

   ```
   $ lsblk -f /dev/sdb1
   ```

5. Hogyan használhatjuk a `mount` parancsot a `6e2c12e3-472d-4bac-a257-c49ac07f3761` UUID-vel rendelkező, a `/mnt/data` címre csatolt exFAT fájlrendszer olvashatóvá tételéhez?

   Mivel a fájlrendszer csatolva van, nem kell aggódni a fájlrendszer típusa vagy azonosítója miatt, csak használjuk a `remount` opciót a `ro` (csak olvasható) paraméterrel és a csatolási ponttal:

   ```
   # mount -o remount,ro /mnt/data
   ```

6. Hogyan kaphatunk egy listát az összes `ext3` és `ntfs` fájlrendszerről, amely jelenleg fel van csatolva a rendszerre?

Használjuk a `mount -t` parancsot, a fájlrendszerek vesszővel elválasztott listájával:

```
# mount -t ext3,ntfs
```

Válaszok a gondolkodtató feladatokra

1. Tekintsük a következő bejegyzést az `/etc/fstab'-ban: `/dev/sdc1 /backup ext4 noatime,nouser,async`. Tudja-e egy felhasználó csatlakoztatni ezt a fájlrendszert a `mount /backup` paranccsal? Miért?

   Nem, a `nouser` paraméter nem teszi lehetővé, hogy hétköznapi felhasználók csatolják ezt a fájlrendszert.

2. Tekintsünk egy távoli fájlrendszert, amely az `/mnt/server` címre van felcsatolva, és a hálózati kapcsolat megszűnése miatt elérhetetlenné vált. Hogyan lehetne kikényszeríteni, hogy leválasztásra kerüljön, vagy, ha ez nem lehetséges, csak olvashatóként csatolja?

   Adjuk át a `-f` és `-r` paramétereket az unmount-nak. A parancs az alábbi lesz: `umount -f -r /mnt/server`. Ne feledjük, hogy a paramétereket csoportosíthatjuk is, így a `umount -fr /mnt/server` is működni fog.

3. Írjunk egy `/etc/fstab` bejegyzést, amely egy btrfs kötetet csatlakoztatna a `Backup` címkével az `/mnt/backup` lemezre, alapértelmezett beállításokkal és anélkül, hogy engedélyezné a bináris programok végrehajtását

   A sor az alábbi lesz: `LABEL=Backup /mnt/backup btrfs defaults,noexec`

4. Tekintsük a következő systemd mount egységet:

   ```
   [Unit]
   Description=External data disk

   [Mount]
   What=/dev/disk/by-uuid/56C11DCC5D2E1334
   Where=/mnt/external
   Type=ntfs
   Options=defaults

   [Install]
   WantedBy=multi-user.target
   ```

   - Mi lenne egyenértékű `/etc/fstab` bejegyzés ehhez a fájlrendszerhez?

     A bejegyzés az alábbi lesz: `UUID=56C11DCC5D2E1334 /mnt/external ntfs defaults`

5. Mi legyen a fenti egység fájlneve, hogy a systemd használni tudja? Hol kell elhelyezni?

A fájlnévnek ugyanannak kell lenni, mint a csatolási pontnak, így `mnt-external.mount` a `/etc/systemd/system` helyen.

104.5 Fájl jogosultságok és tulajdonjogok kezelése

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 101, Objective 104.5

Súlyozás

3

Kulcsfontosságú ismeretek

- A normál és speciális fájlok, valamint mappák hozzáférési jogosultságainak kezelése.
- A biztonság fenntartásához olyan hozzáférési módok használata, mint a suid, sgid és a sticky bit.
- A fájlkészítési mask megváltoztatásának ismerete.
- A csoportmező használata a csoporttagok fájlhozzáférésének megadásához.

A használt fájlok, kifejezések és segédprogramok listája

- `chmod`
- `umask`
- `chown`
- `chgrp`

104.5 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	104 Eszközök, Linux fájlrendszerek, Fájlrendszer-hierarchia szabvány
Fejezet:	104.5 Fájl jogosultságok és tulajdonjogok kezelése
Lecke:	1/1

Bevezetés

Többfelhasználós rendszer lévén a Linuxnak valamilyen módon nyomon kell követnie, hogy kié az egyes fájlok tulajdonjoga, és hogy egy felhasználó jogosult-e műveleteket végrehajtani egy fájlon. Ennek célja, hogy biztosítsa azon felhasználók magánéletét, akik esetleg bizalmasan szeretnék kezelni a fájljaik tartalmát, valamint hogy biztosítsa az együttműködést azáltal, hogy bizonyos fájlokat több felhasználó számára is elérhetővé tesz.

Ez egy háromszintű jogosultsági rendszeren keresztül történik. Minden lemezen lévő fájl egy felhasználó és egy felhasználói csoport tulajdonában van, és háromféle jogosultsággal rendelkezik: egy a tulajdonosnak, egy a fájlt birtokló csoportnak és egy mindenki másnak. Ebben a leckében megtanuljuk, hogyan kérdezhetjük le egy fájl jogosultságait, mit jelentenek ezek a jogosultságok, és hogyan manipulálhatjuk őket.

Fájlokra és mappákra vonatkozó információk lekérdezése

Az `ls` parancs bármely mappa tartalmának listázására használható. Ebben az alapformában csak

a fájlneveket kapjuk meg:

```
$ ls
Another_Directory  picture.jpg  text.txt
```

De minden egyes fájlról sokkal több információ áll rendelkezésre, beleértve a fájltípust, a méretet, a tulajdonjogot és egyebeket. Ahhoz, hogy ezeket az információkat láthassuk, az `ls`-től kell kérni egy “hosszú formátumú” listázást az `-l` paraméter használatával:

```
$ ls -l
total 536
drwxrwxr-x 2 carol carol   4096 Dec 10 15:57 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
```

A fenti kimenet minden egyes oszlopának jelentése van. Nézzük meg a lecke szempontjából fontos oszlopokat.

- A lista *első* oszlopa a fájl típusát és jogosultságait mutatja. Például az `drwxrwxr-x` esetén:
  - Az első karakter (d), jelzi a fájl típuást.
  - A következő három karakter (`rwx`), jelzi a fájl tulajdonosának (más néven *user* vagy `u`) jogosultságait.
  - A következő három karakter (`rwx`), jelzi a fájlt tulajdonló csoport (más néven *group* vagy `g`) jogosultságait.
  - Az utolsó három karakter (`r-x`), jelzi mindenki más (más néven *others* vagy `o`) jogosultságait.

TIP Gyakran hallani azt is, hogy az *others* jogosultságkészletet *world* (világi) jogosultságokként emlegetik, mint “Mindenki másnak a világon ezek a jogosultságai vannak”.

- A *harmadik* és *negyedik* oszlopok a tulajdonosi információkat mutatják: a fájlt birtokló felhasználót és csoportot.
- A *hetedik* és utolsó oszlop a fájl nevét mutatja.

A *második* oszlop az adott fájlra mutató hardlinkek számát jelzi. Az *ötödik* oszlop a fájlméretet mutatja. A *hatodik* oszlop a fájl utolsó módosításának dátumát és időpontját mutatja. Ezek az oszlopok azonban nem relevánsak a jelenlegi téma szempontjából.

Mi lesz a mappákkal?

Ha megpróbálunk információt lekérdezni egy mappáról az `ls -l` használatával, akkor a mappa tartalmának listája jelenik meg:

```
$ ls -l Another_Directory/
total 0
-rw-r--r-- 1 carol carol 0 Dec 10 17:59 another_file.txt
```

Ha ezt el szeretnénk kerülni, és magáról a mappáról szeretnénk információt lekérdezni, adjuk hozzá a `ls`-hez a `-d` paramétert:

```
$ ls -l -d Another_Directory/
drwxrwxr-x 2 carol carol 4096 Dec 10 17:59 Another_Directory/
```

Rejtett fájlok megnézése

A korábban az `ls -l` segítségével lekérdezett lista nem teljes:

```
$ ls -l
total 544
drwxrwxr-x 2 carol carol   4096 Dec 10 17:59 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
```

Van még három másik fájl is ebben a mappában, de ezek el vannak rejtve. Linuxon azok a fájlok, amelyek neve ponttal (`.`) kezdődik, automatikusan el vannak rejtve. Ahhoz, hogy lássuk őket, az `ls`-hez hozzá kell adnunk az `-a` paramétert:

```
$ ls -l -a
total 544
drwxrwxr-x 3 carol carol   4096 Dec 10 16:01 .
drwxrwxr-x 4 carol carol   4096 Dec 10 15:56 ..
drwxrwxr-x 2 carol carol   4096 Dec 10 17:59 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
-rw-r--r-- 1 carol carol      0 Dec 10 16:01 .thisIsHidden
```

A `.thisIsHidden` fájl egyszerűen rejtett, mivel a neve `.`-al kezdődik.

A `.` és a `..` mappák azonban különlegesek. A "`." az aktuális mappára mutató mutató. A `..` pedig a szülő mappára mutat, arra, amelyikben az aktuális mappa található. A Linuxban minden mappa legalább ezt a két mappát tartalmazza.

TIP Az `ls` (és sok más Linux-parancs) több paramétert is kombinálhat. Az `ls -l -a` például `ls -la`-ként is írható.

Fájltípusok megértése

Azt már korábban említettük, hogy az `ls -l` mindenkori kimenetének első betűje a fájl típusát írja le. A három leggyakoribb fájltípus a következő:

`-` (normál fájl)

Egy fájl bármilyen típusú adatot tartalmazhat, és segíthet az adatok kezelésében. A fájlok módosíthatók, áthelyezhetők, másolhatók és törölhetők.

`d` (mappa)

Egy mappa más fájlokat vagy mappákat tartalmaz, és segít a fájlrendszer rendszerezésében. Technikailag a mappa a fájl egy speciális fajtája.

`l` (szimbolikus link)

Ez a “fájl” egy mutató (pointer) egy másik fájlra vagy mappára, bárhol a fájlrendszerben.

Ezeken kívül van még három másik fájltípus, amelyeket legalább ismerni kellene, de ennek a leckének a keretein kívül esnek:

`b` (blokkeszköz)

Ez a fájl egy virtuális vagy fizikai eszközt jelöl, általában lemezeket vagy más típusú tárolóeszközöket, például az első merevlemezt, amelyet a `/dev/sda` reprezentálhat.

`c` (karakteres eszköz)

Ez a fájl egy virtuális vagy fizikai eszközt jelöl. A terminálok (mint például a fő terminál a `/dev/ttyS0`-n) és a soros portok gyakori példái a karakteres eszközöknek.

`s` (socket)

A socketek “csatornaként” szolgálnak, amelyek információt továbbítanak két program között.

WARNING Ne módosítsuk a blokkeszközök, karakteres eszközök vagy socketek engedélyeit, hacsak nem tudjuk biztosan, mit csinálunk. Ez megakadályozhatja a rendszer működését!

Jogosultságok megértése

Az `ls -l` kimenetén a fájljogosultságok közvetlenül a fájltípus után jelennek meg, három, egyenként három karakterből álló csoportként, az `r`, `w` és `x` sorrendben. Hamarosan megtudjuk, ezek mit jelentenek. Ne feledjük, hogy a `-` (kötőjel) az engedélyek hiányát jelenti.

Fájlok jogosultságai

`r`

A *read* rövidítése, oktális értéke `4` (hamarosan lesz szó az oktális értékekről). Ez egy fájl megnyitásának és tartalmának olvasására vonatkozó engedélyt jelent.

`w`

A *write* rövidítése, oktális értéke `2`. Ez a fájl szerkesztésének vagy törlésének engedélyezését jelenti.

`x`

Az *execute* rövidítése, oktális értéke `1`. Ez azt jelenti, hogy a fájl futtatható futtatható programként vagy szkriptként.

Így például egy `rw-` jogosultságú fájl olvasható és írható, de nem futtatható.

Mappák jogosultságai

`r`

A *read* rövidítése, oktális értéke `4`. Ez a mappa tartalmának, például a fájlneveknek az olvasására való jogosultságot jelenti. De ez *nem* jelenti maguknak a fájloknak az olvasási engedélyét.

`w`

A *write* rövidítése, oktális értéke `2`. Ez a mappában lévő fájlok létrehozásának vagy törlésének engedélyezését jelenti.

Ne feledjük, hogy ezeket a változtatásokat nem végezhetjük el csak *write* jogosultsággal, hanem *execute* jogosultságra (x) is szükség van a mappa megváltoztatásához.

`x`

Az *execute* rövidítése, és oktális értéke `1`. Ez a mappába való belépés engedélyét jelenti, de a benne lévő fájlok listázását nem (ehhez az `r` szükséges).

A mappákról szóló utolsó rész kissé zavarosnak tűnhet. Képzeljük el például, hogy van egy

`Another_Directory` nevű mappánk, a következő jogosultságokkal:

```
$ ls -ld Another_Directory/
d--x--x--x 2 carol carol 4,0K Dec 20 18:46 Another_Directory
```

Képzeljük el azt is, hogy ebben a mappában van egy `hello.sh` nevű shell szkript:

```
-rwxr-xr-x 1 carol carol 33 Dec 20 18:46 hello.sh
```

Ha mi vagyunk a `carol` felhasználó, és megpróbáljuk listázni az `Another_Directory` mappa tartalmát, hibaüzenetet kapunk, mivel a felhasználónak nincs olvasási joga a mappához:

```
$ ls -l Another_Directory/
ls: cannot open directory 'Another_Directory/': Permission denied
```

A `carol` felhasználónak azonban van futtatási jogosultsága, ami azt jelenti, hogy beléphet a mappába. Ezért `carol` felhasználó hozzáférhet a mappában lévő fájlokhoz, amennyiben rendelkezik a megfelelő jogosultságokkal *az adott fájlhoz*. Tegyük fel, hogy a felhasználónak teljes jogosultsága (`rwx`) van a `hello.sh` szkripthez. Ekkor *lehet* futtatni a szkriptet, még akkor is, ha *nem* tudja elolvasni az azt tartalmazó mappa tartalmát, ha ismerjük a teljes fájlnevet:

```
$ sh Another_Directory/hello.sh
Helló LPI Világ!
```

Mint már említettük, a jogosultságokat sorrendben adjuk meg: először a fájl tulajdonosának, majd a tulajdonos csoportnak, végül a többi felhasználónak. Amikor valaki megpróbál valamilyen műveletet végrehajtani a fájlon, az engedélyek ellenőrzése ugyanígy történik.

Először a rendszer ellenőrzi, hogy az aktuális felhasználóé-e a fájl, és ha ez igaz, akkor csak az első jogosultságok első csoportját alkalmazza. Ellenkező esetben a rendszer ellenőrzi, hogy az aktuális felhasználó tagja-e annak a csoportnak, amelyik a fájl tulajdonosa. Ebben az esetben csak a második jogosultságkészletet alkalmazza. Minden más esetben a rendszer a harmadik jogosultságkészletet alkalmazza.

Ez azt jelenti, hogy ha az aktuális felhasználó a fájl tulajdonosa, akkor csak a tulajdonosi jogosultságok érvényesek, még akkor is, ha a csoport vagy más engedélyek megengedőbbek, mint a tulajdonos jogosultságai.

Fájl engedélyek módosítása

A `chmod` parancsot egy fájl engedélyeinek módosítására használjuk, és legalább két paramétert igényel: az első leírja, hogy milyen engedélyeket kell megváltoztatni, a második pedig arra a fájlra vagy mappára mutat, ahol a változtatás megtörténik. Ne feledjük, hogy csak a fájl tulajdonosa vagy a rendszergazda (root) módosíthatja a fájl jogosultságait.

A megváltoztatható jogosultságok kétféleképpen, vagy “módban” írhatók le.

Az első, az úgynevezett *szimbolikus mód* (symbolic mod) finomabb ellenőrzést biztosít, lehetővé téve egyetlen jogosultság hozzáadását vagy visszavonását anélkül, hogy a jogosultságkészlet többi elemét módosítaná. A másik mód, az úgynevezett *octal mode* (oktális mód), könnyebben megjegyezhető és gyorsabban használható, ha egyszerre szeretnénk beállítani az összes jogosultság értékét.

Mindkét mód ugyanahhoz a végeredményhez vezet. Így például az alábbi parancsok:

```
$ chmod ug+rw-x,o-rwx text.txt
```

és

```
$ chmod 660 text.txt
```

pontosan ugyanazt a kimenetet fogják eredményezni, egy fájlt az alábbi beállított jogosultságokkal:

```
-rw-rw---- 1 carol carol  765 Dec 20 21:25 text.txt
```

Lássuk, hogyan működnek az egyes módok.

Szimbolikus mód

Amikor a *szimbolikus módban* leírjuk, hogy mely jogosultságokat szeretnénk megváltoztatni, az első karakter(ek) jelzi(k), hogy kinek a jogosultságai lesznek megváltoztatva: a felhasználóé (`u`), a csoporté (`g`), másoké (`o`) és/vagy mindenkié (`a`).

Ezután meg kell adnunk a parancsnak, hogy mit tegyen: megadhatunk egy jogosultságot (`+`), visszavonhatunk egy engedélyt (`-`), vagy beállíthatjuk egy adott értékre (`=`).

Végül meg kell adni, hogy milyen jogosultságon akarunk cselekedni: olvasás (`r`), írás (`w`) vagy

futtatás (`x`).

Képzeljük el például, hogy van egy `text.txt` nevű fájlunk a következő jogosultságkészlettel:

```
$ ls -l text.txt
-rw-r--r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

Ha a fájlt birtokló csoport tagjainak szeretnénk írási jogosultságot adni, akkor a `g+w` paramétert használjuk. Egyszerűbb, ha így gondolkodunk: “A csoport (`g`) számára adjunk (`+`) írási engedélyt (`w`)”. A parancs tehát a következő lenne:

```
$ chmod g+w text.txt
```

Nézzük meg az eredményt az `ls` segítségével:

```
$ ls -l text.txt
-rw-rw-r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

Szeretnénk eltávolítani ugyanazon fájl tulajdonosának az olvasási jogosultságait? Gondoljunk erre a következőképpen: “A felhasználótól (`u`) vonjuk vissza (`-`) az olvasási engedélyeket (`r`)”. Tehát a paraméter `u-r`, így:

```
$ chmod u-r text.txt
$ ls -l text.txt
--w-rw-r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

Mi van akkor, ha a jogosultságokat pontosan `rw-'-ként akarjuk beállítani mindenki számára? Akkor gondoljunk erre a következőképpen: “Mindenki számára (`a`), állítsuk be pontosan (`=`) az olvasást (`r`), az írást (`w`), és ne legyen futtatás (`-`)”. Tehát:

```
$ chmod a=rw- text.txt
$ ls -l text.txt
-rw-rw-rw- 1 carol carol 765 Dec 20 21:25 text.txt
```

Természetesen egyszerre több jogosultságot is lehet módosítani. Ebben az esetben vesszővel (`,`) kell őket elválasztani:

```
$ chmod u+rwx,g-x text.txt
```

```
$ ls -lh text.txt
-rwxrw-rw- 1 carol carol 765 Dec 20 21:25 text.txt
```

A fenti példa a következőképpen olvasható: “A felhasználó (u) számára adjunk (+) olvasási, írási és végrehajtási (rwx) jogosultságokat, a csoporttól (g) pedig vonjuk vissza (-) a végrehajtási jogosultságot (x).”.

A chmod egy mappán futtatva csak a mappa jogosultságait módosítja. A chmod rendelkezik rekurzív móddal is, ami akkor hasznos, ha a “egy mappában és annak almappáiban lévő összes fájl engedélyeit” akarjuk megváltoztatni. Ennek használatához a -R paramétert a parancs neve után, a módosítandó jogosultságok előtt kell megadni:

```
$ chmod -R u+rwx Another_Directory/
```

Ez a parancs a következőképpen olvasható: “Rekurzívan (-R), a felhasználó (u) számára olvasási, írási és végrehajtási (rwx) engedélyeket ad (+)”.

WARNING

Legyünk óvatosak és gondoljuk meg kétszer is, mielőtt a -R kapcsolót használjuk, mivel könnyen megváltoztathatja olyan fájlok és mappák jogosultságait, amelyeket nem akarunk megváltoztatni, különösen olyan mappák esetében, amelyekben sok fájl és almappa található.

Oktális mód

Oktális módban a jogosultságokat más úton adjuk meg: egy három számjegyű oktális értékként, 8-as számrendszerben.

Minden jogosultsághoz tartozik egy érték, az alábbi sorrendben: először az olvasás (r), ami a 4, majd az írás (w), ami a 2 és végül a futtatás (x), ami pedig az 1. Ha nincs jogosultság, használjuk a nulla (0) értéket. Tehát az rwx jogosultság 7 (4+2+1) lenne, az r-x pedig 5 (4+0+1).

A három számjegyből az első a felhasználó (u) jogosultságait reprezentálja, a második a csoportét (g) és a harmadik mindenki másét (o). Ha egy fájl jogosultságait az alábbira szeretnénk beállítani: rw-rw----, az oktális érték 660 lesz:

```
$ chmod 660 text.txt
$ ls -l text.txt
-rw-rw---- 1 carol carol 765 Dec 20 21:25 text.txt
```

Ettől függetlenül az *oktális mód* szintaxisa ugyanaz, mint a *szimbolikus mód* esetén, az első

paraméter reprezentálja a megváltoztatni kívánt jogosultságokat, a második paraméter pedig arra a fájlra vagy mappára mutat, ahol a változtatásokat végre szeretnénk hajtani.

TIP Ha egy jogosultsági érték *páratlan*, a fájl biztosan futtatható!

Melyik szintaxist használjuk? Az *oktális mód* akkor ajánlott, ha a jogosultságokat egy adott értékre, például `640`-re (`rw- r-- ---`) akarjuk állítani.

A *szimbolikus mód* hasznosabb, ha csak egy adott értéket akarunk megfordítani, függetlenül a fájl aktuális jogosultságaitól. Például a felhasználó számára a `chmod u+x script.sh` segítségével hozzáadhatjuk a végrehajtási engedélyeket anélkül, hogy figyelembe vennénk, vagy akár csak megérintenénk a csoport és mások aktuális engedélyeit.

Fájl tulajdonjogának megváltoztatása

A `chown` parancs arra való, hogy megváltoztassuk egy fájl vagy mappa tulajdonjogait. A szintaxis nagyon egyszerű:

```
chown USERNAME:GROUPNAME FILENAME
```

Vegyük például a `text.txt` nevű fájlt:

```
$ ls -l text.txt
-rw-rw---- 1 carol carol 1881 Dec 10 15:57 text.txt
```

A fájl tulajdonosa a `carol` felhasználó, és a csoport is `carol`. Most megváltoztatjuk a fájlt birtokló csoportot egy másik csoportra, például a `students` csoportra:

```
$ chown carol:students text.txt
$ ls -l text.txt
-rw-rw---- 1 carol students 1881 Dec 10 15:57 text.txt
```

Ne feledjük, hogy a fájlokat birtokló felhasználónak nem kell a fájlokat birtokló csoporthoz tartoznia. A fenti példában a `carol` felhasználónak nem kell a `students` csoport tagjainak lennie.

A felhasználói vagy csoportos engedélyek elhagyhatók, ha nem szeretnénk megváltoztatni őket. Tehát, ha csak a fájlt birtokló csoportot szeretnénk megváltoztatni, akkor a `chown :students text.txt` parancsot használjuk. Ha csak a felhasználót szeretnénk megváltoztatni, a parancs a következő lenne: `chown carol: text.txt` vagy csak `chown carol text.txt`. Alternatívaként használhatjuk a `chgrp students text.txt` parancsot is.

Hacsak nem mi vagyunk a rendszergazda (root), nem változtathatjuk meg egy fájl tulajdonjogát egy másik felhasználóra vagy csoportra, amelyhez nem tartozik. Ha megpróbáljuk, az `Operation not permitted` (Művelet nem engedélyezett) hibaüzenetet kapjuk.

Csoportok lekérdezése

Mielőtt megváltoztatnánk egy fájl tulajdonjogát, hasznos lehet tudni, hogy milyen csoportok léteznek a rendszerben, mely felhasználók tagjai egy csoportnak, és mely csoportokhoz tartozik egy felhasználó.

Ha meg szeretnénk nézni, hogy milyen csoportok vannak a rendszerben, használjuk a `getent group` parancsot. A kimenet ehhez hasonló lesz (a kimenet rövidítve van):

```
$ getent group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,rigues
tty:x:5:rigues
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:rigues
```

Ha tudni szeretnénk, hogy egy felhasználó melyik csoporthoz tartozik, adjuk meg a felhasználónevet a `groups` paramétereként:

```
$ groups carol
carol : carol students cdrom sudo dip plugdev lpadmin sambashare
```

Ha a fordítottját szeretnénk (azaz mely felhasználók tartoznak egy csoporthoz), használjuk a `groupmems` parancsot. A `-g` paraméter megadja a csoportot, az `-l` pedig felsorolja a csoport összes tagját:

```
# groupmems -g cdrom -l
carol
```

TIP A `groupmems` csak rootként, rendszergazdaként futtatható. Ha nem rootként vagyunk

> bejelentkezve, használjuk a `sudo`-t a parancs előtt.

Alapértelmezett jogosultságok

Végezzünk el egy kísérletet! Nyissunk meg egy terminálablakot, és hozzunk létre egy üres fájlt a következő paranccsal:

```
$ touch testfile
```

Most nézzük meg a fájl jogosultságait. Ezek eltérőek *lehetnek* a saját rendszerünk esetén, de tegyük fel, hogy így néznek ki:

```
$ ls -lh testfile
-rw-r--r-- 1 carol carol 0 jul 13 21:55 testfile
```

A jogosultságok `rw-r—r--`: *olvasás* és *írás* a felhasználónak, és *olvasás* a csoportnak és mindenki másnak, vagyis oktális módban `644`. Most próbáljunk meg létrehozni egy mappát:

```
$ mkdir testdir
$ ls -lhd testdir
drwxr-xr-x 2 carol carol 4,0K jul 13 22:01 testdir
```

A jogosultságok most `rwxr-xr-x` : *olvasás*, *írás* és *futtatás* a felhasználónak, *olvasás* és *futtatás* a csoportnak és mindenki másnak, vagy `755` oktális módban.

Nem számít, hogy a fájlrendszerben hol van, minden létrehozott fájl vagy mappa ugyanazokat a jogosultságokat kapja. Gondolkodjunk el azon, hogy honnan származnak!

Ezek a *user mask* vagy `umask`-ból származnak, amely minden létrehozott fájl alapértelmezett jogosultságát beállítja. Az aktuális értékeket a `umask` paranccsal ellenőrizhetjük:

```
$ umask
0022
```

De ez nem úgy néz ki, mint az `rw-r—r--`, vagy akár a `644`. Talán a `-S` paraméterrel kellene próbálkoznunk, hogy szimbolikus módban kapjunk kimenetet:

```
$ umask -S
```

```
u=rwx,g=rx,o=rx
```

Ezek ugyanazok a jogosultságok, amelyeket a tesztmappánk kapott a fenti példák egyikében. De miért van az, hogy amikor létrehoztunk egy fájlt, a jogosultságok másak voltak?

Nos, nincs értelme alapértelmezés szerint globális végrehajtási engedélyeket beállítani mindenkinek bármilyen fájlhoz, nem igaz? A mappáknak szükségük van a végrehajtási engedélyekre (különben nem lehet megnyitni őket), de a fájloknak nem, tehát nem kapnak ilyeneket. Ezért `rw-r—r--`.

Az alapértelmezett jogosultságok megjelenítése mellett a `umask` az aktuális shell munkamenetre vonatkozó jogosultságok megváltoztatására is használható. Például, ha az alábbi parancsot használjuk:

```
$ umask u=rwx,g=rwx,o=
```

Minden új mappa az `rwxrwx---` és minden fájl a `rw-rw----` jogosultságot fogja örökölni (mivel nem kapnak végrehajtási jogot). Ha megismételjük a fenti példákat (a `testfile` és a `testdir` létrehozását), majd megnézzük a jogosultságokat, az alábbit kapjuk:

```
$ ls -lhd test*
drwxrwx--- 2 carol carol 4,0K jul 13 22:25 testdir
-rw-rw---- 1 carol carol    0 jul 13 22:25 testfile
```

Ha pedig a `umask`-ot az `-S` (szimbolikus mód) paraméter nélkül ellenőrizzük, akkor a következőt kapjuk:

```
$ umask
0007
```

Az eredmény nem tűnik ismerősnek, mert a használt értékek eltérőek. Íme egy táblázat az egyes értékekkel és azok jelentésével:

Érték	Fájl-jogosultságok	Mappa-jogosultságok
`0`	`rw-`	`rwx`
1	`rw-`	`rw-`
2	`r--`	`r-x`

Érték	Fájl-jogosultságok	Mappa-jogosultságok
3	`r--`	`r--`
4	`-w-`	`-wx`
5	`-w-`	`-w-`
6	`---`	`--x`
7	`---`	`---`

Látható, hogy a `007` megfelel az `rwxrwx---`-nek, pontosan úgy, ahogy kértük. Az első nulla figyelmen kívül hagyható.

Speciális jogosultságok

A felhasználói, csoportos és egyéb olvasási, írási és végrehajtási engedélyeken kívül minden fájlnak három további *speciális jogosultsága* lehet, amelyek megváltoztathatják egy mappa működését vagy egy program futását. Ezek szimbolikus vagy oktális módban adhatók meg, és a következők:

Sticky Bit

A sticky (ragadós) bit, más néven *restricted deletion flag* oktális értéke `1`, és szimbolikus módban egy `t` jelöli a mindenki más jogosultságain belül. Ez csak a mappákra vonatkozik, a normál fájlokra nincs hatása. Linuxon megakadályozza, hogy a felhasználók eltávolítsanak vagy átnevezzenek egy fájlt egy mappában, kivéve, ha az adott fájl vagy mappa a tulajdonukban van.

Azon mappák esetében, amelyeknél a sticky bit be van állítva, egy `t` jelenik meg az `x` helyén az *others* jogosultságai között az `ls -l` kimenetén:

```
$ ls -ld Sample_Directory/
drwxr-xr-t 2 carol carol 4096 Dec 20 18:46 Sample_Directory/
```

Oktális módban a speciális engedélyek 4 számjegyű jelöléssel kerülnek megadásra, ahol az első számjegy jelöli az adott speciális engedélyt. Például az `Another_Directory` mappához tartozó sticky bit (1 érték) beállításához oktális módban, `755` jogosultsággal, a parancs a következő lenne:

```
$ chmod 1755 Another_Directory
$ ls -ld Another_Directory
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
```

Set GID

A Set GID, más néven SGID vagy Set Group ID bit, oktális értéke `2`, és szimbolikus módban a *group* engedélyeknél egy `s` jelöli. Ez futtatható fájlokra vagy mappákra alkalmazható. Fájlok esetén a folyamatot a fájl tulajdonában lévő csoport jogosultságaival futtatja. Ha mappákra alkalmazzuk, akkor minden alatta létrehozott fájl vagy mappa a szülő mappa csoportját örökli.

Az SGID bitet tartalmazó fájlok és mappák esetén az `ls -l` kimenetén a *group* engedélyeknél az `x` helyett egy `s` jelenik meg:

```
$ ls -l test.sh
-rwxr-sr-x 1 carol root 33 Dec 11 10:36 test.sh
```

Ha szimbolikus módban szeretnénk SGID-engedélyeket adni egy fájlhoz, a parancs a következő:

```
$ chmod g+s test.sh
$ ls -l test.sh
-rwxr-sr-x 1 carol root     33 Dec 11 10:36 test.sh
```

A következő példa segít jobban megérteni az SGID hatását egy mappára. Tegyük fel, hogy van egy `Sample_Directory` nevű mappánk, amely a `carol` felhasználó és a `users` csoport tulajdonában van, a következő jogosultsági struktúrával:

```
$ ls -ldh Sample_Directory/
drwxr-xr-x 2 carol users 4,0K Jan 18 17:06 Sample_Directory/
```

Most lépjünk át ebbe a mappába, és a `touch` paranccsal hozzunk létre egy üres fájlt benne. Az eredmény a következő lesz:

```
$ cd Sample_Directory/
$ touch newfile
$ ls -lh newfile
-rw-r--r-- 1 carol carol 0 Jan 18 17:11 newfile
```

Amint látjuk, a fájl a `carol` felhasználó és a `carol` csoport tulajdonában van. De ha a mappa SGID jogosultsággal lenne beállítva, az eredmény más lenne. Először is adjuk hozzá az SGID bitet a `Sample_Directory` mappához, és ellenőrizzük az eredményeket:

```
$ sudo chmod g+s Sample_Directory/
```

```
$ ls -ldh Sample_Directory/
drwxr-sr-x 2 carol users 4,0K Jan 18 17:17 Sample_Directory/
```

A csoportjogosultságoknál az `s` jelzi, hogy az SGID bit be van állítva. Most lépjünk át ebbe a mappába, és hozzunk létre egy újabb üres fájlt a `touch` paranccsal:

```
$ cd Sample_Directory/
$ touch emptyfile
$ ls -lh emptyfile
 -rw-r--r-- 1 carol users 0 Jan 18 17:20 emptyfile
```

A fájl tulajdonosa a `users` csoport. Ez azért van, mert az SGID bit miatt a fájl a szülő mappa csoporttulajdonosát örökölte, ami a `users`.

Set UID

A SUID, más néven Set User ID oktális értéke `4`, és szimbolikus módban a *user* engedélyeknél egy `s` jelöli. Csak fájlokra vonatkozik, mappákra nincs hatása. A viselkedése hasonló az SGID bithez, de a folyamat a fájl tulajdonában lévő *user* jogosultságaival fog futni. A SUID bitet tartalmazó fájlok esetén az `ls -l` kimenetén az `x` helyett egy `s` jelzi a felhasználó jogosultságait:

```
$ ls -ld test.sh
-rwsr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

Egy paraméterhez több speciális engedélyt is kombinálhatunk. Tehát, ha az SGID (`2` érték) és SUID (`4` érték) értéket akarjuk beállítani oktális üzemmódban a `test.sh` szkripthez `755` jogosultsággal, akkor a következőket kell beírnunk:

```
$ chmod 6755 test.sh
```

És az eredmény az alábbi lesz:

```
$ ls -lh test.sh
-rwsr-sr-x 1 carol carol 66 Jan 18 17:29 test.sh
```

TIP

Ha a terminálunk támogatja a színeket, márpedig manapság a legtöbb támogatja, akkor az `ls -l` kimenetére pillantva gyorsan láthatjuk, hogy ezek a speciális engedélyek be vannak-e állítva. A sticky bit esetében a mappa neve fekete

betűtípussal, kék háttérrel jelenhet meg. Ugyanez vonatkozik az SGID (sárga háttérrel) és SUID (piros háttérrel) bitekkel rendelkező fájlokra is. A színek eltérőek lehetnek attól függően, hogy milyen Linux-disztribúciót és terminálbeállításokat használunk.

Gyakorló feladatok

1. Hozzunk létre egy `emptydir` nevű mapát az `mkdir emptydir` parancs segítségével! Majd az `ls` használatával listázzuk az `emptydir` mappa jogosultságait!

2. Hozzunk létre egy `emptyfile` nevű üres fájlt a `touch emptyfile` paranccsal! A `chmod` szimbolikus módban történő használatával, adjunk futtatási jogokat az `emptyfile` fájl tulajdonosának és vegyük el az írási és futtatási jogosultságokat mindenki mástól! Csak a `chmod` parancsot használjuk!

3. Mik lennének egy fájl alapértelmezett jogosultságai, ha az `umask` érték `027`-re van állítva?

4. Képzeljük el a `test.sh` nevű shell szkriptet az alábbi jogosultságokkal és tulajdonjogokkal:

```
-rwxr-sr-x 1 carol root     33 Dec 11 10:36 test.sh
```

   - Mik a tulajdonos jogosultságai a fájlon?

   - Az oktális jelölést használva, mi lenne a `chmod` szintaxisa a fájlnak adott speciális jogosultság “visszavonásához”?

5. Vegyük az alábbi fájlt:

```
$ ls -l /dev/sdb1
brw-rw---- 1 root disk 8, 17 Dec 21 18:51 /dev/sdb1
```

   Milyen típusú fájl az `sdb1`? Ki tud bele írni?

6. Vegyük az alábbi 4 fájlt:

```
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
----r--r-- 1 carol carol    0 Dec 11 10:55 foo.bar
```

```
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
drwxr-sr-x 2 carol users  4,0K Jan 18 17:26 Sample_Directory
```

Írjuk le a megfelelő jogosultságokat minden fájlhoz és mappához, az oktális mód 4 számjegyű jelölésével!

`Another_Directory`	
`foo.bar`	
`HugeFile.zip`	
`Sample_Directory`	

Gondolkodtató feladatok

1. Próbáljuk ki ezt egy terminálban: hozzounk létre egy üres fájlt `emptyfile` néven a `touch emptyfile` paranccsal! Most "nullázzuk" a fájl jogosultságait a `chmod 000 emptyfile` paranccsal! Mi történik, ha az `emptyfile` jogosultságait úgy változtatjuk meg, hogy csak *egy* értéket adunk meg a `chmod`-nak oktális módban, például `chmod 4 emptyfile`? És ha kettőt használunk, mint a `chmod 44 emptyfile`? Mit tudhatunk meg arról, hogy a `chmod` hogyan olvassa a numerikus értéket?

2. Tekintsük meg az ideiglenes mappa (`/tmp`) engedélyeit egy Linux rendszerben:

```
$ ls -l /tmp
drwxrwxrwt  19 root root  16K Dec 21 18:58 tmp
```

   A felhasználónak, csoportnak és mindenki másnak teljes jogosultságai vannak. De törölhet-e egy sima felhasználó *bármilyen* fájlt ebből a mappából? Miért van ez így?

3. A `test.sh` nevű fájl a következő jogosultságokkal rendelkezik: `-rwsr-xr-x`. Ez azt jelenti, hogy a SUID bit be van állítva. Most futtassuk a következő parancsokat:

```
$ chmod u-x test.sh
$ ls -l test.sh
-rwSr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

   Mit csináltunk? Mit jelent a nagybetűs `S`?

4. Hogyan hozhatnánk létre egy `Box` nevű mappát, ahol az összes fájl automatikusan a `users` csoport tulajdonában van, és csak az a felhasználó törölheti, aki létrehozta őket?

Összefoglalás

Ebben a leckében megtanultuk, hogyan használhatjuk az `ls`-t a fájljogosultságokról szóló információk lekérdezésére (és dekódolására), hogyan szabályozhatjuk vagy változtathatjuk meg, hogy ki hozhat létre, törölhet vagy módosíthat egy fájlt a `chmod` segítségével, mind *oktális*, mind *szimbolikus* módban, hogyan módosíthatjuk a fájlok tulajdonjogát a `chown` és `chgrp` segítségével, és hogyan kérdezhetjük le és változtathatjuk meg a fájlok és mappák alapértelmezett jogosultsági maszkját a `umask` segítségével.

A leckében az alábbi parancsokról volt szó:

`ls`

Fájlok listázása, opcionálisan olyan részletekkel, mint például a jogosultságok.

`chmod`

Egy fájl vagy mappa jogosultságainak módosítása.

`chown`

Egy fájl vagy mappa tulajdonos felhasználójának és/vagy csoportjának módosítása.

`chgrp`

Egy fájl vagy mappa tulajdonosi csoportjának módosítása.

`umask`

A fájlok és mappák alapértelmezett jogosultsági maszkjának lekérdezése vagy beállítása.

Válaszok a gyakorló feladatokra

1. Hozzunk létre egy `emptydir` nevű mapát az `mkdir emptydir` parancs segítségével! Majd az `ls` használatával listázzuk az `emptydir` mappa jogosultságait!

   Adjuk a `-d` paramétert az `ls`-hez, hogy a mappa tartalmának listázása helyett a mappa fájlattribútumait láthassuk. A válasz tehát a következő:

   ```
   ls -l -d emptydir
   ```

   Bónuszpont, ha a két paramétert egybeolvasztottuk: `ls -ld emptydir`.

2. Hozzunk létre egy `emptyfile` nevű üres fájlt a `touch emptyfile` paranccsal! A `chmod` szimbolikus módban történő használatával, adjunk futtatási jogokat az `emptyfile` fájl tulajdonosának és vegyük el az írási és futtatási jogosultságokat mindenki mástól! Csak a `chmod` parancsot használjuk!

   Gondolkodjunk így:

   - "A fájlt tulajdonló felhasználónak (`u`) adjunk (`+`) futtatási (`x`) jogokat", tehát `u+x`.
   - "A csoporttól (`g`) és mindenki mástól (`o`), vegyük el (`-`) az írási (`w`) és futtatási (`x`) jogokat", tehát `go-wx`.

     A két jogosultságkészlet összekapcsolásához vesszőt teszünk közéjük. A végeredmény tehát a következő:

     ```
     chmod u+x,go-wx emptyfile
     ```

3. Mik lennének egy fájl alapértelmezett jogosultságai, ha az `umask` érték `027`-re van állítva?

   A jogosultságok: `rw-r-----`

4. Képzeljük el a `test.sh` nevű shell szkriptet az alábbi jogosultságokkal és tulajdonjogokkal:

   ```
   -rwxr-sr-x 1 carol root     33 Dec 11 10:36 test.sh
   ```

   - Mik a tulajdonos jogosultságai a fájlon?

     A tulajdonos jogosultsága (az `ls -l` kimenetén a 2-tól a 4. karakterig) az `rwx`, tehát a válasz: "olvasni, írni és futtatni a fájlt".

- Az oktális jelölést használva, mi lenne a `chmod` szintaxisa a fájlnak adott speciális engedély "visszavonásához"?

  "Visszavonhatjuk" a speciális jogosultságokat azzal, ha egy 4. számjegyet, a `0`-t átadjuk a `chmod`-nak. Az aktuális jogosultság `755`, tehát a parancs `chmod 0755`.

5. Vegyük az alábbi fájlt:

```
$ ls -l /dev/sdb1
brw-rw---- 1 root disk 8, 17 Dec 21 18:51 /dev/sdb1
```

   Milyen típusú fájl az `sdb1`? Ki tud bele írni?

   Az `ls -l` kimenetének első karaktere a fájl típusát mutatja. A `b` egy *blokkeszköz*, általában egy (belső vagy külső) lemez, amely a géphez van csatlakoztatva. A tulajdonos (`root`) és a `disk` csoport bármely felhasználója írhat rá.

6. Vegyük az alábbi 4 fájlt:

```
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
----r--r-- 1 carol carol    0 Dec 11 10:55 foo.bar
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
drwxr-sr-x 2 carol users 4,0K Jan 18 17:26 Sample_Directory
```

   Írjuk le a megfelelő jogosultságokat minden fájlhoz és mappához, az oktális mód 4 számjegyű jelölésével!

   A megfelelő jogosultságok, oktális módban az alábbiak:

`Another_Directory`	`1755`. `1` a sticky bit, `755` a szokásos jogosultságok (`rwx` a felhasználónak, `r-x` a csoportnak és mindenki másnak).
`foo.bar`	`0044`. Nincs speciális jogosultság (így az első számjegy `0`), nincs jogosultság a felhasználónak (`---`) és csak olvasás (`r—r--`) a csoportnak és mindenki másnak.
`HugeFile.zip`	`0664`. Nincs speciális jogosultság, tehát az első számjegy `0`. `6` (`rw-`) a felhasználónak és a csoportnak, `4` (`r--`) mindenki másnak.

`Sample_Directory`	`2755`. `2` az SGID bit, `7` (`rwx`) a felhasználónak, `5` (`r-x`) a csoportnak és mindenki másnak.

n <<< == Válaszok a gondolkodtató feladatokra

1. Próbáljuk ki ezt egy terminálban: hozzunk létre egy üres fájlt `emptyfile` néven a `touch emptyfile` paranccsal! Most "nullázzuk" a fájl jogosultságait a `chmod 000 emptyfile` paranccsal! Mi történik, ha az `emptyfile` jogosultságait úgy változtatjuk meg, hogy csak *egy* értéket adunk meg a `chmod`-nak oktális módban, például `chmod 4 emptyfile`? És ha kettőt használunk, mint a `chmod 44 emptyfile`? Mit tudhatunk meg arról, hogy a `chmod` hogyan olvassa a numerikus értéket?

   Ne feledjük, hogy "nulláztuk" az `emptyfile` jogosultságait. Tehát a kezdeti állapota a következő lesz:

   ```
   ---------- 1 carol carol    0 Dec 11 10:55 emptyfile
   ```

   Most próbáljuk ki az első parancsot, `chmod 4 emptyfile`:

   ```
   $ chmod 4 emptyfile
   $ ls -l emptyfile
   -------r-- 1 carol carol 0 Dec 11 10:55 emptyfile
   ```

   Látjuk? Az *others* jogosultságai megváltoztak. És mi történik, ha 2 számjeggyel próbálkozunk, mint pl. `chmod 44 emptyfile`?

   ```
   $ chmod 44 emptyfile
   $ ls -l emptyfile
   ----r--r-- 1 carol carol 0 Dec 11 10:55 emptyfile
   ```

   Most a *group* és *others* jogosultságok voltak érintettek. Ebből arra következtethetünk, hogy oktális módban a `chmod` "visszafelé" olvassa az értéket, a legkevésbé jelentős számjegytől (*others*) a legjelentősebbig (*user*). Ha egy számjegyet adunk át, akkor az *others* jogosultságait módosítjuk. Két számjeggyel a *group* és *others*, három számjeggyel a *user*, *group* és *others*, négy számjeggyel pedig a *user*, *group*, *others* és a speciális jogosultságokat módosítjuk.

2. Tekintsük meg az ideiglenes mappa (`/tmp`) engedélyeit egy Linux rendszerben:

   ```
   $ ls -l /tmp
   ```

```
drwxrwxrwt  19 root root  16K Dec 21 18:58 tmp
```

A felhasználónak, csoportnak és mindenki másnak teljes jogosultságai vannak. De törölhet-e egy sima felhasználó *bármilyen* fájlt ebből a mappából? Miért van ez így?

A `/tmp` egy úgynevezett *world writeable* mappa, ami azt jelenti, hogy bármelyik felhasználó írhat bele. De nem akarjuk, hogy egy felhasználó mások által létrehozott fájlokat piszkáljon, ezért a *sticky bit* be van állítva (amit a `t` jelez az *others* jogosultságoknál). Ez azt jelenti, hogy egy felhasználó törölheti a `/tmp` állományokat, de csak a saját maga által létrehozottakat.

3. A `test.sh` nevű fájl a következő jogosultságokkal rendelkezik: `-rwsr-xr-x`. Ez azt jelenti, hogy a SUID bit be van állítva. Most futtassuk a következő parancsokat:

```
$ chmod u-x test.sh
$ ls -l test.sh
-rwSr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

Mit csináltunk? Mit jelent a nagybetűs `S`?

Eltávolítottuk a fájl tulajdonosának végrehajtási jogosultságait. Az `ls -l` kimenetén az `s` (vagy `t`) veszi át az `x` helyét, így a rendszernek meg kell mutatnia, hogy a felhasználónak van-e végrehajtási jogosultsága vagy sem. Ezt úgy teszi, hogy megváltoztatja a különleges karakterek esetét.

A kisbetűs `s` az első jogosultsági csoportban azt jelenti, hogy a fájl tulajdonosa rendelkezik végrehajtási jogosultsággal, és a SUID bit be van állítva. A nagybetűs `S` azt jelenti, hogy a fájl tulajdonosa nem rendelkezik (`-`) végrehajtási jogokkal, és a SUID bit be van állítva.

Ugyanez mondható el az SGID-ről is, a kisbetűs `s` a második jogosultsági csoportban azt jelenti, hogy a fájl tulajdonosának a csoportja rendelkezik végrehajtási jogosultsággal, és az SGID bit be van állítva. A nagybetűs `S` azt jelenti, hogy a fájl tulajdonosának csoportja nem rendelkezik (`-`) végrehajtási jogokkal, és az SGID bit be van állítva.

Ez igaz a sticky bitre is, amelyet a harmadik jogosultsági csoportban az `t` jelöl. A kisbetűs `t` azt jelenti, hogy a sticky bit be van állítva, és mindenki másnak végrehajtási joga van. A nagybetűs `T` azt jelenti, hogy a sticky bit be van állítva, és mindenki másnak nincs futtatási jogosultságuk.

4. Hogyan hozhatnánk létre egy `Box` nevű mappát, ahol az összes fájl automatikusan a `users` csoport tulajdonában van, és csak az a felhasználó törölheti, aki létrehozta őket?

Ez egy többlépcsős folyamat. Az első lépés a mappa létrehozása:

```
$ mkdir Box
```

Azt akarjuk, hogy minden fájl, amelyet ebben a mappában hozunk létre, automatikusan a `users` csoporthoz legyen rendelve. Ezt úgy érhetjük el, hogy ezt a csoportot állítjuk be a mappa tulajdonosának, majd beállítjuk az SGID bitet. Biztosítanunk kell azt is, hogy a csoport bármely tagja írhasson ebbe a mappába.

Mivel nem érdekel minket, hogy milyen a többi jogosultság, és csak a speciális biteket akarjuk "átfordítani", érdemes a szimbolikus módot használni:

```
$ chown :users Box/
$ chmod g+wxs Box/
```

Vegyük figyelembe, hogy ha az aktuális felhasználó nem tartozik a `users` csoporthoz, akkor a fenti parancsok előtt a `sudo` parancsot kell használni, hogy a változtatást root-ként hajtsuk végre.

Most az utolsó rész következik, annak biztosítása, hogy csak az a felhasználó törölhesse a fájlt, aki létrehozta azt. Ezt úgy érhetjük el, hogy a mappában beállítjuk a sticky bitet (amit egy `t` jelez). Ne feledjük, hogy ez a beállítás a mindenki más engedélyeinél (o) van beállítva.

```
$ chmod o+t Box/
```

A `Box` mappa jogosultságai a következők kell, hogy legyenek:

```
drwxrwsr-t 2 carol users  4,0K Jan 18 19:09 Box
```

Természetesen az SGID-et és a sticky bitet megadhatjuk csak egy `chmod` paranccsal:

```
$ chmod g+wxs,o+t Box/
```

Bónusz, ha erre is gondoltunk.

104.6 Szimbolikus és hard linkek létrehozása és megváltoztatása

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 101, Objective 104.6

Súlyozás

2

Kulcsfontosságú ismeretek

- Linkek létrehozása.
- Hard és/vagy soft linkek azonosítása.
- Fájlok másolása vs linkelés.
- Linkek használata a rendszergazdai feladatok támogatására.

A használt fájlok, kifejezések és segédprogramok listája

- `ln`
- `ls`

Linux Professional Institute

104.6 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	104 Eszközök, Linux fájlrendszerek, Fájlrendszer-hierarchia szabvány
Fejezet:	104.6 Szimbolikus és hard linkek létrehozása és megváltoztatása
Lecke:	1/1

Bevezetés

A Linuxon néhány fájl különleges bánásmódban részesül, vagy a tárolási helyük miatt, mint például az ideiglenes fájlok, vagy a fájlrendszerrel való kölcsönhatásuk miatt, mint például a hivatkozások. Ebben a leckében megtudhatjuk, hogy mik azok a linkek és hogyan kezelhetjük őket.

A linkek megértése

Mint már említettük, Linuxon mindent fájlként kezelünk. De van egy speciális fájlfajta, amit *linknek* hívnak, és a Linux rendszerben kétféle link létezik:

Szimbolikus linkek

Más néven *soft linkek*, egy másik fájl elérési útvonalára mutatnak. Ha töröljük a fájlt, amire a link mutat (az úgynevezett *target* (cél)), a link továbbra is létezni fog, de “nem működik”, mivel most már a “semmire” mutat.

Hard linkek

Gondoljunk úgy a hard linkre, mint az eredeti fájl második nevére. Ez *nem* duplikátum, hanem egy további bejegyzés a fájlrendszerben, amely ugyanarra a helyre (inode) mutat a lemezen.

TIP

Az *inode* egy olyan adatszerkezet, amely egy fájlrendszerben lévő objektum (például egy fájl vagy mappa) attribútumait tárolja. Ezen attribútumok közé tartoznak a jogosultságok, a tulajdonjog és az, hogy a lemez mely blokkjain tárolják az objektum adatait. Gondoljunk rá úgy, mint egy index bejegyzésére, innen ered a neve is, amely az “index node”-ból származik.

Munka a hard linkekkel

Hard linkek létrehozása

Linuxon a hard link létrehozásának parancsa az `ln`. Az alapvető szintaxis a következő:

```
$ ln TARGET LINK_NAME
```

A `TARGET`-nek már léteznie kell (ez az a fájl, amelyre a hivatkozás mutat), és ha a cél nem az aktuális mappában van, vagy ha máshol akarjuk létrehozni a hivatkozást, akkor *meg kell* adni a teljes elérési utat. Például a parancs:

```
$ ln target.txt /home/carol/Documents/hardlink
```

létre fog hozni egy `hardlink` nevű fájlt a `/home/carol/Documents/` mappában, ami a `target.txt` fájlra hivatkozik az aktuális mappában.

Ha az utolsó paramétert (`LINK_NAME`) elhagyjuk, akkor egy, a célponttal azonos nevű link jön létre az aktuális mappában.

Hard linkek menedzselése

A hard linkek olyan bejegyzések a fájlrendszerben, amelyek különböző nevűek, de ugyanarra az adatra mutatnak a lemezen. Minden ilyen név azonos, és használható egy fájlra való hivatkozáshoz. Ha megváltoztatjuk az egyik név tartalmát, akkor az összes többi, erre a fájlra mutató név tartalma is megváltozik, mivel ezek a nevek mind ugyanarra az adatra mutatnak. Ha az egyik nevet töröjük, a többi név továbbra is működik.

Ez azért történik, mert amikor “törlünk” egy fájlt, az adatok valójában nem törlődnek a lemezről. A rendszer egyszerűen törli a fájlrendszer táblájában a lemezen lévő adatoknak megfelelő inode-

ra mutató bejegyzést. De ha van egy második bejegyzés, amely ugyanarra az inode-ra mutat, akkor még mindig hozzáférhetünk az adatokhoz. Gondoljunk erre úgy, mintha két út találkozna ugyanabban a pontban. Még ha az egyik utat blokkoljuk vagy átirányítjuk is, akkor is elérhetjük a célt a másik úton.

Ezt az `ls` paraméter `-i` paraméterének használatával ellenőrizhetjük. Tekintsük a következő mappa tartalmát:

```
$ ls -li
total 224
3806696 -r--r--r-- 2 carol carol 111702 Jun  7 10:13 hardlink
3806696 -r--r--r-- 2 carol carol 111702 Jun  7 10:13 target.txt
```

Az engedélyek előtti szám az inode száma. Látjuk, hogy a `hardlink` és a `target.txt` fájlnak ugyanaz a száma (`3806696`)? Ez azért van, mert az egyik a másik hard linkje.

De melyik az eredeti és melyik a link? Nem igazán lehet megmondani, mivel minden gyakorlati szempontból ugyanazok.

Vegyük figyelembe, hogy minden egyes hard link, amely egy fájlra mutat, növeli a fájl *linkszámát* (link count). Ez a szám közvetlenül az `ls -l` kimenetén a jogosultságok után található. Alapértelmezés szerint minden fájl linkszáma `1` (a mappáké `2`), és minden rá mutató hard link eggyel növeli a számot. Ez az oka tehát annak, hogy a fenti listában szereplő fájlok linkszáma `2`.

A szimbolikus linkekkel ellentétben itt csak fájlokra lehet hard linkeket létrehozni, és mind a linknek, mind a célnak ugyanabban a fájlrendszerben kell lennie.

Hard linkek mozgatása és eltávolítása

Mivel a hardlinkeket hagyományos fájlokként kezeljük, az `rm` paranccsal törölhetők, az `mv` paranccsal pedig átnevezhetők vagy mozgathatók a fájlrendszerben. És mivel a hard link a céllal azonos inode-ra mutat, szabadon mozgatható, nem kell attól tartani, hogy a link “eltörik”.

Szimbolikus linkek

Szimbolikus linkek létrehozása

A szimbolikus link létrehozására használt parancs szintén az `ln`, de az `-s` paraméterrel kiegészítve. Például így:

```
$ ln -s target.txt /home/carol/Documents/softlink
```

Ez létrehoz egy `softlink` nevű fájlt a `/home/carol/Documents/` mappában, amely az aktuális mappában lévő `target.txt` fájlra mutat.

A hard linekkhez hasonlóan elhagyhatjuk a hivatkozás nevét, hogy az aktuális mappában a céllal azonos nevű hivatkozást hozzunk létre.

Szimbolikus linkek menedzselése

A szimbolikus hivatkozások a fájlrendszer egy másik elérési útvonalára mutatnak. Soft linkeket hozhatunk létre fájlokra *és* mappákra, akár különböző partíciókon is. Egy szimbolikus linket elég könnyű észrevenni az `ls` parancs kimenete alapján:

```
$ ls -lh
total 112K
-rw-r--r-- 1 carol carol 110K Jun  7 10:13 target.txt
lrwxrwxrwx 1 carol carol   12 Jun  7 10:14 softlink -> target.txt
```

A fenti példában a `softlink` fájl jogosultságainak első karaktere `l`, ami szimbolikus linket jelez. Továbbá, közvetlenül a fájlnév után látható a link céljának neve, a `target.txt` fájl.

Vegyük figyelembe, hogy a fájl- és mappalistákon a soft linkek maguk mindig az `rwx` engedélyeket mutatják a felhasználó, a csoport és mások számára, de a gyakorlatban a hozzáférési engedélyek ugyanazok, mint a célponté.

Szimbolikus linkek mozgatása és eltávolítása

A hard linkekhez hasonlóan a szimbolikus linkek is eltávolíthatók az `rm` segítségével, és áthelyezhetők vagy átnevezhetők az `mv` segítségével. Létrehozásukkor azonban különös gondossággal kell eljárni, hogy a linket ne “törjük el”, ha elmozdítjuk az eredeti helyéről.

Szimbolikus hivatkozások létrehozásakor tisztában kell lennünk azzal, hogy hacsak nincs teljes mértékben megadva az elérési útvonal, a célpont helyét a hivatkozás helyéhez képest *relatívnak* értelmezi a rendszer. Ez problémákat okozhat, ha a hivatkozás vagy a fájl, amelyre mutat, áthelyezésre kerül.

Ezt egy példával könnyebb megérteni. Tegyük fel, hogy van egy `original.txt` nevű fájl az aktuális mappában, és egy `softlink` nevű szimbolikus linket szeretnénk létrehozni hozzá. Használhatjuk a következőt:

```
$ ln -s original.txt softlink
```

És úgy tűnik, minden rendben lesz. Ellenőrizzük le az `ls`-el:

```
$ ls -lh
total 112K
-r--r--r-- 1 carol carol 110K Jun  7 10:13 original.txt
lrwxrwxrwx 1 carol carol   12 Jun  7 19:23 softlink -> original.txt
```

Nézzük meg, hogyan épül fel a hivatkozás: a `softlink` az (→) `original.txt` fájlra mutat. Nézzük azonban, mi történik, ha a linket az előző mappába helyezzük át, és megpróbáljuk megjeleníteni a tartalmát a `less` paranccsal:

```
$ mv softlink ../
$ less ../softlink
../softlink: No such file or directory
```

Mivel az `original.txt` elérési útvonalát nem adtuk meg, a rendszer azt feltételezi, hogy az ugyanabban a mappában van, mint a hivatkozás. Ha ez már nem igaz, a hivatkozás nem működik.

Ezt úgy lehet megelőzni, ha a hivatkozás létrehozásakor mindig megadjuk a cél teljes elérési útvonalát:

```
$ ln -s /home/carol/Documents/original.txt softlink
```

Így nem számít, hogy hová helyezzük át a linket, az továbbra is működni fog, mivel a cél abszolút helyére mutat. Ellenőrizzük az `ls`-el:

```
$ ls -lh
total 112K
lrwxrwxrwx 1 carol carol   40 Jun  7 19:34 softlink -> /home/carol/Documents/original.txt
```

Gyakorló feladatok

1. Mi a `chmod` paramétere *szimbolikus* módban, hogy engedélyezzük a sticky bitet egy mappán?

2. Képzeljük el, hogy van egy `document.txt` nevű fájl a `/home/carol/Documents` mappában. Milyen paranccsal hozhatunk létre egy `text.txt` nevű szimbolikus linket az aktuális mappán?

3. Magyarázzuk el a különbséget egy fájlhoz vezető hard link és a fájl másolata között!

Gondolkodtató feladatok

1. Képzeljük el, hogy egy mappában létrehozunk egy `recipes.txt` nevű fájlt. Ebben a mappában létrehozunk egy hard linket is erre a fájlra, a `receitas.txt`-t, és egy szimbolikus (vagy *soft*) linket erre a fájlra, a `rezepte.txt`-t.

   ```
   $ touch recipes.txt
   $ ln recipes.txt receitas.txt
   $ ln -s recipes.txt rezepte.txt
   ```

   A mappa tartalma a következő lesz:

   ```
   $ ls -lhi
   total 160K
   5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 receitas.txt
   5388833 -rw-r--r-- 4 carol carol  0K jun 17 17:25 recipes.txt
   5388837 lrwxrwxrwx 1 carol carol  12 jun 17 17:25 rezepte.txt -> receitas.txt
   ```

   Ne feledjük, hogy a `receitas.txt` hard linkként ugyanarra az inode-ra mutat, amelyhez a `recipes.txt` van rendelve. Mi történne a `rezepte.txt` soft linkkel, ha a `receitas.txt` fájlt törölnénk? Miért?

2. Képzeljük el, hogy egy pendrive van csatlakoztatva a rendszerhez és mountolva van a `/media/youruser/FlashA` címre. Létre akarunk hozni egy `schematics.pdf` nevű linket a home mappában, amely a pendrive gyökerén lévő `esquema.pdf` fájlra mutat. Írjuk be tehát a következő parancsot:

   ```
   $ ln /media/youruser/FlashA/esquema.pdf ~/schematics.pdf
   ```

   Mi fog történni? Miért?

3. Vegyük a következő `ls -lah` kimenetet:

   ```
   $ ls -lah
   total 3,1M
   drwxr-xr-x 2 carol carol 4,0K jun 17 17:27 .
   drwxr-xr-x 5 carol carol 4,0K jun 17 17:29 ..
   -rw-rw-r-- 1 carol carol 2,8M jun 17 15:45 compressed.zip
   -rw-r--r-- 4 carol carol  77K jun 17 17:25 document.txt
   -rw-rw-r-- 1 carol carol 216K jun 17 17:25 image.png
   ```

```
-rw-r--r-- 4 carol carol  77K jun 17 17:25 text.txt
```

- Hány link mutat a `document.txt` fájlra?

- Soft vagy hard linkek?

- Melyik paramétert kell átadni az `ls`-nek, hogy lássuk, melyik inode-ot foglalják el az egyes állományok?

4. Képzeljük el, hogy a `~/Documents` mappában van egy `clients.txt` nevű fájl, amely néhány ügyfél nevét tartalmazza, és egy `somedir` nevű mappa. Ezen belül van egy *másik* fájl, szintén `clients.txt` néven, más nevekkel. Ennek a struktúrának a lemásolásához használjuk a következő parancsokat:

```
$ cd ~/Documents
$ echo "John, Michael, Bob" > clients.txt
$ mkdir somedir
$ echo "Bill, Luke, Karl" > somedir/clients.txt
```

Ezután hozzunk létre egy linket a `somedir`-ben `partners.txt` néven, amely erre a fájlra mutat, a következő parancsokkal:

```
$ cd somedir/
$ ln -s clients.txt partners.txt
```

Tehát a mappastruktúra a következő:

```
Documents
|-- clients.txt
`-- somedir
    |-- clients.txt
    `-- partners.txt -> clients.txt
```

Most helyezzük át a `partners.txt`-t a `somedir`-ből a `~/Documents`-be és listázzuk a tartalmát!

```
$ cd ~/Documents/
```

```
$ mv somedir/partners.txt .
$ less partners.txt
```

Működni fog a link? Ha igen, akkor melyik fájl tartalma lesz felsorolva? Miért?

5. Vegyük a következő fájlokat:

```
-rw-r--r-- 1 carol carol 19 Jun 24 11:12 clients.txt
lrwxrwxrwx 1 carol carol 11 Jun 24 11:13 partners.txt -> clients.txt
```

Milyen hozzáférési jogosultságok vannak a `partners.txt` fájlhoz? Miért?

Összefoglalás

Ebben a leckében megtanultuk:

- Mik azok a linkek.
- A különbségeket a *szimbolikus* és *hard* linkek között.
- Hogyan hozzunk létre linkeket.
- Hogyan mozgassuk, nevezzük át vagy töröljük ezeket a linkeket.

A leckében az alábbi parancsokról volt szó:

- `ln`: A “link” parancs. Önmagában ez a parancs egy hard linket hoz létre. Az `-s` kapcsolóval *szimbolikus* vagy *soft* link hozható létre. Ne feledjük, hogy a hard linkek csak ugyanazon a partíción és fájlrendszeren belül lehetnek, a szimbolikus linkek pedig áthaladhatnak a partíciók és fájlrendszerek között (még a hálózathoz csatolt tárolókon is).
- Az `ls -i` paramétere, amely lehetővé teszi egy fájl inode számának megtekintését.

Válaszok a gyakorló feladatokra

1. Mi a `chmod` paramétere *szimbolikus* módban, hogy engedélyezzük a sticky bitet egy mappán?

   Szimbolikus módban a sticky bit szimbóluma a `t`. Mivel ezt a jogosultságot szeretnénk engedélyezni (hozzáadni) a mappához, a paraméternek `+t`-nek kell lennie.

2. Képzeljük el, hogy van egy `document.txt` nevű fájl a `/home/carol/Documents` mappában. Milyen paranccsal hozhatunk létre egy `text.txt` nevű szimbolikus linket az aktuális mappán?

   `ln -s` a parancs a szimbolikus létrehozásához. Mivel meg kell adni a linkelni kívánt fájl teljes elérési útvonalát, a parancs a következőképpen néz ki:

   ```
   $ ln -s /home/carol/Documents/document.txt text.txt
   ```

3. Magyarázzuk el a különbséget egy fájlhoz vezető hard link és a fájl másolata között!

   A hard link csak egy másik neve egy fájlnak. Bár úgy néz ki, mint az eredeti fájl másolata, mind a link és az eredeti fájl ugyanaz, mivel ugyanarra az adatra mutatnak a lemezen. A link tartalmán végrehajtott változtatások az eredetin is tükröződnek, és vica-versa. A másolat egy teljesen független egység, amely más helyet foglal el a lemezen. A másolaton végrehajtott változtatások nem jelennek meg az eredetin, és vica-versa.

Válaszok a gondolkodtató feladatokra

1. Képzeljük el, hogy egy mappában létrehozunk egy `recipes.txt` nevű fájlt. Ebben a mappában létrehozunk egy hard linket is erre a fájlra, a `receitas.txt`-t, és egy szimbolikus (vagy *soft*) linket erre a fájlra, a `rezepte.txt`-t.

   ```
   $ touch recipes.txt
   $ ln recipes.txt receitas.txt
   $ ln -s receitas.txt rezepte.txt
   ```

   A mappa tartalma a következő lesz:

   ```
   $ ls -lhi
   total 160K
   5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 receitas.txt
   5388833 -rw-r--r-- 4 carol carol  0K jun 17 17:25 recipes.txt
   5388837 lrwxrwxrwx 1 carol carol  12 jun 17 17:25 rezepte.txt -> receitas.txt
   ```

   Ne feledjük, hogy a `receitas.txt` hard linkként ugyanarra az inode-ra mutat, amelyhez a `recipes.txt` van rendelve. Mi történne a `rezepte.txt` soft linkkel, ha a `receitas.txt` fájlt törölnénk? Miért?

   A `rezepte.txt` soft link nem működik. Ennek oka, hogy a soft linkek nevekre mutatnak, nem pedig inodeokra és a `receitas.txt` név már nem létezik, még akkor sem, ha az adatok még mindig a lemezen vannak `recipes.txt` néven.

2. Képzeljük el, hogy egy pendrive van csatlakoztatva és a `/media/youruser/FlashA` címre van mountolva. Létre akarunk hozni egy `schematics.pdf` nevű linket a home mappában, amely a pendrive gyökerén lévő `esquema.pdf` fájlra mutat. Írjuk be tehát a következő parancsot:

   ```
   $ ln /media/youruser/FlashA/esquema.pdf ~/schematics.pdf
   ```

   Mi fog történni? Miért?

   A parancs sikertelen lesz. A hibaüzenet `Invalid cross-device link` lesz és ez a magyarázatot is világossá teszi: a hard linkek nem mutathatnak egy másik partíción vagy eszközön lévő célpontra. Az egyetlen módja egy ilyen hivatkozás létrehozásának, ha *szimbolikus* vagy *soft* linket használunk, az `ln` parancshoz hozzáadva az `-s` paramétert.

3. Vegyük a következő `ls -lah` kimenetet:

```
$ ls -lah
total 3,1M
drwxr-xr-x 2 carol carol 4,0K jun 17 17:27 .
drwxr-xr-x 5 carol carol 4,0K jun 17 17:29 ..
-rw-rw-r-- 1 carol carol 2,8M jun 17 15:45 compressed.zip
-rw-r--r-- 4 carol carol  77K jun 17 17:25 document.txt
-rw-rw-r-- 1 carol carol 216K jun 17 17:25 image.png
-rw-r--r-- 4 carol carol  77K jun 17 17:25 text.txt
```

- Hány link mutat a `document.txt` fájlra?

  Minden fájl `1` linkszámmal kezdődik. Mivel a fájl linkszáma `4`, három link mutat erre a fájlra.

- Soft vagy hard linkek?

  Hard linkek, mivel a soft linkek nem növelik a fájl linkszámát.

- Melyik paramétert kell átadni az `ls`-nek, hogy lássuk, melyik inode-ot foglalják el az egyes állományok?

  A paraméter az `-i`. Az inode az `ls` kimenetének első oszlopa lesz, mint például:

  ```
  $ ls -lahi
  total 3,1M
  5388773 drwxr-xr-x 2 carol carol 4,0K jun 17 17:27 .
  5245554 drwxr-xr-x 5 carol carol 4,0K jun 17 17:29 ..
  5388840 -rw-rw-r-- 1 carol carol 2,8M jun 17 15:45 compressed.zip
  5388833 -rw-r--r-- 4 carol carol  77K jun 17 17:25 document.txt
  5388837 -rw-rw-r-- 1 carol carol 216K jun 17 17:25 image.png
  5388833 -rw-r--r-- 4 carol carol  77K jun 17 17:25 text.txt
  ```

4. Képzeljük el, hogy a `~/Documents` mappában van egy `clients.txt` nevű fájl, amely néhány ügyfél nevét tartalmazza, és egy `somedir` nevű mappa. Ezen belül van egy *másik* fájl, szintén `clients.txt` néven, más nevekkel. Ennek a struktúrának a lemásolásához használjuk a következő parancsokat:

   ```
   $ cd ~/Documents
   $ echo "John, Michael, Bob" > clients.txt
   $ mkdir somedir
   $ echo "Bill, Luke, Karl" > somedir/clients.txt
   ```

Ezután hozzunk létre egy linket a `somedir`-ben `partners.txt` néven, amely erre a fájlra mutat, a következő parancsokkal:

```
$ cd somedir/
$ ln -s clients.txt partners.txt
```

Tehát a mappastruktúra a következő:

```
Documents
|-- clients.txt
`-- somedir
    |-- clients.txt
    `-- partners.txt -> clients.txt
```

Most helyezzük át a `partners.txt`-t a `somedir`-ből a `~/Documents`-be és listázzuk a tartalmát!

```
$ cd ~/Documents/
$ mv somedir/partners.txt .
$ less partners.txt
```

Működni fog a link? Ha igen, akkor melyik fájl tartalma lesz felsorolva? Miért?

Ez “trükkös” , de a link működni fog és a `~/Documents`-ben lévő fájl tartalma lesz listázva, amely a `John`, `Michael`, `Bob` neveket tartalmazza.

Ne feledjük, hogy mivel a `partners.txt` soft link létrehozásakor nem adtuk meg a `clients.txt` cél teljes elérési útját, a cél helyét a link helyéhez képest relatívnak fogja értelmezni, ami ebben az esetben az aktuális mappa.

Amikor a hivatkozás a `~/Documents/somedir` mappából a `~/Documents` mappába került, a linknek nem kellett volna működnie, mivel a cél már nem volt ugyanabban a mappában, mint a hivatkozás. Történetesen azonban a `~/Documents` mappában van egy `clients.txt` nevű fájl, így a link erre a fájlra mutat a `~/somedir` mappában lévő eredeti célpont helyett.

Ennek elkerülése érdekében szimbolikus link létrehozásakor mindig adjuk meg a cél teljes elérési útját.

5. Vegyük a következő fájlokat:

```
-rw-r--r-- 1 carol carol 19 Jun 24 11:12 clients.txt
```

```
lrwxrwxrwx 1 carol carol 11 Jun 24 11:13 partners.txt -> clients.txt
```

Milyen hozzáférési jogosultságok vannak a `partners.txt` fájlhoz? Miért?

A `partners.txt` hozzáférési engedélyei `rw-r—r—r--`, mivel a linkek mindig ugyanazokat a hozzáférési engedélyeket öröklik, mint a célpont.

104.7 Rendszerfájlok keresése és a fájlok megfelelő helyre helyezése

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 101, Objective 104.7

Súlyozás

2

Kulcsfontosságú ismeretek

- A fájlok helyes helyének megértése az FHS alatt.
- Fájlok és parancsok keresése Linux rendszerben.
- Az FHS-ben meghatározott fontos fájlok és könyvtárak helyének és céljának ismerete.

A használt fájlok, kifejezések és segédprogramok listája

- `find`
- `locate`
- `updatedb`
- `whereis`
- `which`
- `type`
- `/etc/updatedb.conf`

104.7 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	104 Eszközök, Linux fájlrendszerek, Fájlrendszer-hierarchia szabvány
Fejezet:	104.7 Rendszerfájlok keresése és a fájlok megfelelő helyre helyezése
Lecke:	1/1

Bevezetés

A Linux disztribúciók mindenféle formában és méretben léteznek, de egy dolog, ami szinte mindegyikben közös, hogy követik a *Filesystem Hierarchy Standard* (FHS) szabványt, amely egy “szabványos layout”-ot határoz meg a fájlrendszer számára, ami megkönnyíti az együttműködést és a rendszeradminisztrációt. Ebben a leckében többet fogunk megtudni erről a szabványról, és arról, hogyan találhatunk meg fájlokat egy Linux rendszerben.

A Fájlrendszer-hierarchia szabvány

A Fájrendszer-hierarchia szabvány (Filesystem Hierarchy Standard (FHS)) a Linux Foundation törekvése a Linux rendszerek mappaszerkezetének és a mappák tartalmának szabványosítására. A szabványnak való megfelelés nem kötelező, de a legtöbb disztribúció követi.

NOTE Akit érdekelnek a fájlrendszer-szervezés részletei, elolvashatja az FHS 3.0 specifikációt, amely több formátumban is elérhető a http://refspecs.linuxfoundation.org/fhs.shtml oldalon

A szabvány szerint az alapvető mappaszerkezet a következő:

`/`

Ez a gyökérkönyvtár, a hierarchia legfelső mappája. Minden más mappa ezen belül található. Egy fájlrendszert gyakran hasonlítanak egy “fához”, így ez lenne a “törzs”, amelyhez minden ág kapcsolódik.

`/bin`

Alapvető binárisok, minden felhasználó számára elérhető.

`/boot`

A rendszerindítási folyamathoz szükséges fájlok, beleértve az initrd-t (Initial RAM Disk) és magát a Linux kernelt.

`/dev`

Eszközfájlok. Ezek lehetnek a rendszerhez csatlakoztatott fizikai eszközök (például a `/dev/sda` az első SCSI vagy SATA lemez) vagy a kernel által biztosított virtuális eszközök.

`/etc`

Hoszt-specifikus konfigurációs fájlok. A programok szükség esetén létrehozhatnak almappákat az `/etc` alatt több konfigurációs fájl tárolására.

`/home`

A rendszerben minden felhasználónak van egy “home” mappája a személyes fájlok és beállítások tárolsához és a legtöbb a `/home` alatt található. Általában a home mappa neve ugyanaz, mint a felhasználónév, szóval a John nevű felhasználónak a mappája a `/home/john` alatt lesz. A kivételek a szuperfelhasználó (root), akinek külön mappája van (`/root`) és néhány rendszerfelhasználó.

`/lib`

Az operációs rendszer indításához és a a `/bin` és `/sbin` alatt található bináris programok futtatásához szükséges megosztott mappák.

`/media`

A felhasználó által csatolható cserélhető adathordozók, például flash meghajtók, CD- és DVD-ROM-olvasók, floppy diszkek, memóriakártyák és külső lemezek ide vannak mountolva.

`/mnt`

Csatolási pont az ideiglenesen csatolt fájlrendszerekhez.

`/opt`

Alkalmazások szoftvercsomagjai.

`/root`

A szuperfelhasználó (root) home mappája.

`/run`

Futásidejű változó adatok.

`/sbin`

Rendszer binárisok.

`/srv`

A rendszer által kiszolgált adatok. Például a webszerver által kiszolgált oldalakat a következő cím alatt lehet tárolni: `/srv/www`.

`/tmp`

Ideiglenes fájlok.

`/usr`

Csak olvasható felhasználói adatok, beleértve egyes másodlagos segédprogramok és alkalmazások által igényelt adatokat is.

`/proc`

Virtuális fájlrendszer, amely a futó folyamatokhoz kapcsolódó adatokat tartalmazza.

`/var`

A rendszer működése során írt változó adatok, beleértve a nyomtatási várólistát, a naplóadatokat, a postafiókokat, az ideiglenes fájlokat, a böngésző gyorsítótárát stb.

Ne feledjük, hogy néhány ilyen mappa, mint például az `/etc`, `/usr` és `/var`, almappák egész hierarchiáját tartalmazza.

Ideiglenes fájlok

Az ideiglenes fájlok olyan fájlok, amelyeket a programok olyan adatok tárolására használnak, amelyekre csak rövid ideig van szükség. Ilyenek lehetnek a futó folyamatok adatai, az összeomlási naplók, az automatikus mentésből származó scratch fájlok, a fájlkonvertálás során használt köztes fájlok, a gyorsítótár-fájlok és hasonlók.

Az ideiglenes fájlok helye

A *Filesystem Hierarchy Standard* (FHS) 3.0 verziója meghatározza az ideiglenes fájlok szabványos helyeit a Linux rendszereken. Minden egyes helynek más a célja és a viselkedése, és ajánlott, hogy a fejlesztők kövessék az FHS által meghatározott konvenciókat, amikor ideiglenes adatokat írnak a lemezre.

`/tmp`

Az FHS szerint a programoknak nem szabad feltételezniük, hogy az ide írt fájlok megmaradnak a program invokálásai között. Az ajánlás szerint ezt a mappát a rendszer indításakor törölni kell (minden fájlt eltávolítani), de ez nem kötelező.

`/var/tmp`

Egy másik hely az ideiglenes fájloknak, de ezt a helyet *nem szabad törölni* a rendszer indításakor. Az itt tárolt fájlok általában az újraindítások után is megmaradnak.

`/run`

Ez a mappa tartalmazza a futó folyamatok által használt futásidejű változó adatokat, például a folyamatazonosító fájlokat (`.pid`). Azok a programok, amelyeknek egynél több futásidejű fájlra van szükségük, létrehozhatnak itt almappákat. Ezt a helyet a rendszer indításakor ki kell üríteni. Ennek a mappának a korábbi megfelelője a `/var/run` mappa volt, és egyes rendszereken a `/var/run` mappa szimbolikus hivatkozás lehet a `/run` mappára.

Megjegyzendő, hogy semmi sem akadályozza meg, hogy egy program a rendszerben máshol is létrehozzon ideiglenes fájlokat, de jó gyakorlat, ha tiszteletben tartjuk az FHS által meghatározott konvenciókat.

Fájlok keresése

A Linux rendszerben lévő fájlok kereséséhez használhatjuk a `find` parancsot. Ez egy nagyon hatékony eszköz, sok paraméterrel, amelyekkel a viselkedését és a kimenetet pontosan a saját igényeink szerint módosíthatjuk.

A `find`-nek két argumentumra van szüksége: egy kiindulási pontra és arra, hogy mit keressen. Például, ha az aktuális mappában (és az almappákban) minden olyan fájlt meg akarunk keresni, amelynek a neve `.jpg`-re végződik, akkor a következőt használhatjuk:

```
$ find . -name '*.jpg'
./pixel_3a_seethrough_1.jpg
./Mate3.jpg
./Expert.jpg
```

```
./Pentaro.jpg
./Mate1.jpg
./Mate2.jpg
./Sala.jpg
./Hotbit.jpg
```

Ez minden olyan fájlra illik, amelyiknek a nevének utolsó négy karaktere `.jpg`, függetlenül attól, hogy mi áll előtte, mivel a `*` a "bármi" joker. Nézzük meg azonban, mi történik, ha a minta *végére* egy másik `*` kerül:

```
$ find . -name '*.jpg*'
./pixel_3a_seethrough_1.jpg
./Pentaro.jpg.zip
./Mate3.jpg
./Expert.jpg
./Pentaro.jpg
./Mate1.jpg
./Mate2.jpg
./Sala.jpg
./Hotbit.jpg
```

A (fentebb kiemelt) `Pentaro.jpg.zip` fájl nem szerepelt az előző listában, mert bár a nevében szerepel a `.jpg`, nem felelt meg a mintának, mivel utána további karakterek következtek. Az új minta jelentése "bármi `.jpg` bármi", tehát ennek már megfelel.

TIP Ne feledjük, hogy a `-name` paraméter esetében a nagy- és kisbetűket kell figyelembe venni! Ha nagy- és kisbetűkre nem érzékeny keresést szeretnénk végezni, használjuk az `-iname` paramétert.

A `*.jpg` kifejezést aposztrófok közé kell tenni, hogy a shell ne értelmezze magát a mintát. Próbáljuk ki nélkülük, és nézzük meg, mi történik.

Alapértelmezés szerint a `find` a kezdőpontnál kezdődik, és végigmegy a talált almappákon (és azok almappáin). Ezt a viselkedést korlátozhatjuk a `-maxdepth N` paraméterekkel, ahol `N` a szintek maximális száma.

Ha csak az aktuális mappában szeretnénk keresni, használjuk a `-maxdepth 1` parancsot. Tegyük fel, hogy a következő mappaszerkezetünk van:

```
directory
├── clients.txt
```

```
├── partners.txt -> clients.txt
└── somedir
    ├── anotherdir
    └── clients.txt
```

Ha a `somedir` mappában szeretnénk keresni, akkor a `-maxdepth 2`-t kell használnia (az aktuális mappa +1 szinttel lejjebb). Az `anotherdir` mappán belüli kereséshez a `-maxdepth 3`-ra lenne szükség (az aktuális mappa +2 szinttel lejjebb). A `-mindepth N` paraméter éppen ellenkezőleg működik, és csak *legalább* `N` szinttel lejjebb lévő mappákban keres.

A `-mount` paraméter használható annak elkerülésére, hogy a `find` a csatlakoztatott fájlrendszereken belülre lemenjen. A `-fstype` paraméterrel a keresést bizonyos fájlrendszer-típusokra is korlátozhatjuk. Így a `find /mnt -fstype exfat -iname "*report*"` csak a `/mnt` alá mountolt exFAT fájlrendszerekben keresne.

Attribútumok keresése

Az alábbi paraméterek segítségével kereshetünk bizonyos tulajdonságokkal rendelkező fájlokat, például olyanokat, amelyeket a felhasználó írhat, amelyek meghatározott jogosultságokkal rendelkeznek, vagy amelyek megadott méretűek:

`-user USERNAME`
: `USERNAME` felhasználó tulajdonában lévő fájlok.

`-group GROUPNAME`
: `GROUPNAME` csoport tulajdonában lévő fájlok.

`-readable`
: Az aktuális felhasználó által olvasható fájlok.

`-writable`
: Az aktuális felhasználó által írható fájlok.

`-executable`
: Az aktuális felhasználó által futtatható fájlok. Mappák esetében minden olyan mappára vonatkozik, amelybe a felhasználó beléphet (`x` jogosultság).

`-perm NNNN`
: Minden olyan fájl, amely pontosan az `NNNN` jogosultsággal rendelkezik. Például a `-perm 0664` minden olyan fájlra illik, amelyet a felhasználó és a csoport olvashat és írhat, és amelyet mindenki más is olvashat (vagy `rw-rw-r--`).

Ha az `NNNN` elé egy `-`-t is beírunk, akkor olyan fájlokat kereshetünk, amelyeknek *legalább* a megadott jogosultsága van. Például a `-perm -644` olyan fájlokat keres, amelyek legalább `644` (`rw-r—r--`) jogosultsággal rendelkeznek. Ez magában foglalja a `664` (`rw-rw-r--`) vagy akár a `775` (`rwxrwx-r-x`) jogosultságú fájlokat is.

`-empty`

Üres fájlok és mappák.

`-size N`

Bármilyen `N` méretű fájl, ahol az `N` alapértelmezés szerint 512 bájtos blokkok száma. Az `N` más egységeket jelző utótagokat is fogad: az `Nc` bájtban, az `Nk` kibibájtban (KiB, 1024 bájt szorzata), az `NM` mebibájtban (MiB, 1024 * 1024 szorzata), az `NG` pedig gibibájtban (GiB, 1024 * 1024 * 1024 szorzata) számolja a méretet.

A relatív méretek kereséséhez ismét hozzáadhatjuk a `+` vagy `-` előtagokat (itt a *nagyobb, mint* és *kisebb, mint*). Például a `-size -10M` minden 10 MiB-nál kisebb méretű fájlt megtalál.

Ha például olyan fájlokat szeretnénk keresni a home mappánkban, amelyek a név bármelyik részében a nagy- és kisbetűket figyelmen kívül hagyó `report` mintát tartalmazzák, `0644` jogosultsággal rendelkeznek, 10 nappal ezelőtt érte el őket valaki, és legalább 1 Mib méretűek, akkor használhatjuk a következőt:

```
$ find ~ -iname "*report*" -perm 0644 -atime 10 -size +1M
```

Keresés idő alapján

Az attribútumokra történő keresés mellett idő szerinti keresést is végezhetünk, így megtalálhatjuk azokat a fájlokat, amelyekhez egy adott időszakban hozzáfértek, amelyek attribútumai megváltoztak vagy amelyeket módosítottak. A paraméterek a következők:

`-amin N`, **`-cmin N`**, **`-mmin N`**

Ez olyan fájlokat fog keresni, amelyekhez hozzáfértek, amelyek attribútumai megváltoztak, illetve amelyeket `N` perccel ezelőtt módosítottak.

`-atime N`, **`-ctime N`**, **`-mtime N`**

Ez olyan fájlokat fog keresni, amelyekhez hozzáfértek, amelyek attribútumai megváltoztak, illetve amelyeket `N*24` perccel ezelőtt módosítottak..

A `-cmin N` és `-ctime N` esetében bármilyen attribútumváltozás egyezést eredményez, beleértve a jogosultságok változását, a fájl olvasását vagy írását. Ez különösen hatékonnyá teszi ezeket a

paramétereket, mivel gyakorlatilag bármilyen művelet, amely a fájlt érinti, egyezést vált ki.

A következő példa az aktuális mappában lévő minden olyan fájlnak megfelel, amelyet 24 óránál nem régebben módosítottak, és amely nagyobb, mint 100 MB:

```
$ find . -mtime -1 -size +100M
```

A `locate` és az `updatedb` használata

A `locate` és az `updatedb` olyan parancsok, amelyekkel gyorsan megkereshetünk egy adott mintának megfelelő fájlt egy Linux rendszeren. De a `find`-al ellentétben a `locate` nem a fájlrendszerben keresi meg a mintát: ehelyett az `updatedb` parancs futtatásával létrehozott adatbázisban keresi meg. Ez nagyon gyors eredményt ad, de pontatlan lehet attól függően, hogy mikor frissítették utoljára az adatbázist.

A legegyszerűbb módja a `locate` használatának, ha megadunk neki egy keresési mintát. Például, ha minden JPEG képet meg akarunk találni, akkor használjuk a `locate jpg` parancsot. A találatok listája elég hosszú lehet, de így kell kinéznie:

```
$ locate jpg
/home/carol/Downloads/Expert.jpg
/home/carol/Downloads/Hotbit.jpg
/home/carol/Downloads/Mate1.jpg
/home/carol/Downloads/Mate2.jpg
/home/carol/Downloads/Mate3.jpg
/home/carol/Downloads/Pentaro.jpg
/home/carol/Downloads/Sala.jpg
/home/carol/Downloads/pixel_3a_seethrough_1.jpg
/home/carol/Downloads/jpg_specs.doc
```

Ha a minta `jpg`, a `locate` mindent kilistáz, ami tartalmazza ezt a mintát, függetlenül attól, hogy mi van előtte vagy utána. Erre példa a fenti felsorolásban szereplő `jpg_specs.doc` fájl: tartalmazza a mintát, de a kiterjesztése nem `jpg`.

TIP Ne feledjük, hogy a `locate` mintákra illeszkedik, nem fájlkiterjesztésekre!

Alapértelmezés szerint a minta nagy- és kisbetű érzékeny. Ez azt jelenti, hogy a `.JPG` fájlokat tartalmazó fájlok nem jelennek meg, mivel a minta kisbetűs. Ennek elkerülése érdekében adjuk meg az `-i` paramétert a `locate`-nek. Megismételjük az előző példánkat:


```
$ locate -i .jpg
/home/carol/Downloads/Expert.jpg
/home/carol/Downloads/Hotbit.jpg
/home/carol/Downloads/Mate1.jpg
/home/carol/Downloads/Mate1_old.JPG
/home/carol/Downloads/Mate2.jpg
/home/carol/Downloads/Mate3.jpg
/home/carol/Downloads/Pentaro.jpg
/home/carol/Downloads/Sala.jpg
/home/carol/Downloads/pixel_3a_seethrough_1.jpg
```

Vegyük észre, hogy a fenti félkövérrel szedett `Mate1_old.JPG` fájl nem volt jelen az előző listában!

Több mintát is átadhatunk a `locate`-nek, csak szóközzel kell elválasztani őket. Az alábbi példa a `zip` és `jpg` mintáknak megfelelő fájlokat keresné meg a nagy- és kisbetűket figyelmen kívül hagyva:

```
$ locate -i zip jpg
/home/carol/Downloads/Expert.jpg
/home/carol/Downloads/Hotbit.jpg
/home/carol/Downloads/Mate1.jpg
/home/carol/Downloads/Mate1_old.JPG
/home/carol/Downloads/Mate2.jpg
/home/carol/Downloads/Mate3.jpg
/home/carol/Downloads/OPENMSXPIHAT.zip
/home/carol/Downloads/Pentaro.jpg
/home/carol/Downloads/Sala.jpg
/home/carol/Downloads/gbs-control-master.zip
/home/carol/Downloads/lineage-16.0-20190711-MOD-quark.zip
/home/carol/Downloads/pixel_3a_seethrough_1.jpg
/home/carol/Downloads/jpg_specs.doc
```

Ha több mintát használunk, beállíthatjuk, hogy a locate csak azokat a fájlokat listázza, amelyek *mindegyiknek* megfelelnek. Ezt az `-A` kapcsolóval tehetjük meg. A következő példa minden olyan fájlt megmutat, amely megfelel a `.jpg` *és* a `.zip` mintának:

```
$ locate -A .jpg .zip
/home/carol/Downloads/Pentaro.jpg.zip
```

Ha egy adott mintának megfelelő fájlok számát szeretnénk megszámolni ahelyett, hogy a teljes

elérési útjukat listáztatnánk, használhatjuk a `-c` kapcsolót. Például a rendszerben lévő `.jpg` fájlok megszámlálásához:

```
$ locate -c .jpg
1174
```

A `locate` egyik problémája, hogy csak az `updatedb` által generált adatbázisban (a `/var/lib/mlocate.db` állományban) található bejegyzéseket mutatja meg. Ha az adatbázis elavult, a kimenet olyan fájlokat mutathat, amelyeket az utolsó frissítés óta töröltek. Ennek elkerülésére az `-e` paraméter hozzáadása az egyik megoldás, amely ellenőrzi, hogy a fájl még létezik-e, mielőtt megjelenítené a kimeneten.

Természetesen ez nem oldja meg azt a problémát, hogy az utolsó adatbázis-frissítés után létrehozott fájlok nem jelennek meg. Ehhez frissíteni kell az adatbázist az `updatedb` paranccsal. Hogy ez mennyi ideig fog tartani, az a lemezen lévő fájlok mennyiségétől függ.

Az `updatedb` működésének kontrollálása

Az `updatedb` viselkedését az `/etc/updatedb.conf` fájlban lehet szabályozni. Ez egy szöveges fájl, ahol minden sor egy-egy változót vezérel. Az üres tetszőket figyelmen kívül hagyjuk, és a # karakterrel kezdődő sorokat kommentárként kezeljük.

`PRUNEFS=`

Az `updatedb` nem vizsgálja az e paraméter után megadott fájlrendszertípusokat. A típusok listáját szóközzel kell elválasztani, és maguk a típusok nem érzékenyek a nagy- és kisbetűkre, tehát az `NFS` és az `nfs` ugyanaz.

`PRUNENAMES=`

Ez egy szóközzel elválasztott lista azokról a mappákról, amelyeket az `updatedb` nem vizsgálhat.

`PRUNEPATHS=`

Azoknak az elérési utaknak a listája, amelyeket az `updatedb` figyelmen kívül hagy. Az elérési utak neveit szóközzel kell elválasztani, és ugyanúgy kell megadni, ahogyan az `updatedb` megjelenítené őket (például `/var/spool /media`)

`PRUNE_BIND_MOUNTS=`

Ez egy egyszerű `yes` vagy `no` változó. Ha `yes`, a bind mounts (azok a mappák, amelyek a `mount --bind` paranccsal máshová vannak csatolva) figyelmen kívül maradnak.

Binárisok, man oldalak és forráskód keresése

A `which` egy nagyon hasznos parancs, amely megmutatja egy futtatható fájl teljes elérési útvonalát. Ha például a `bash` futtatható állományát akarjuk megtalálni, használhatjuk a következőt:

```
$ which bash
/usr/bin/bash
```

Az `-a` kapcsoló hozzáadásával a parancs az összes olyan elérési utat megmutatja, amely megfelel a végrehajtható fájlnak. Figyeljük meg a különbséget:

```
$ which mkfs.ext3
/usr/sbin/mkfs.ext3

$ which -a mkfs.ext3
/usr/sbin/mkfs.ext3
/sbin/mkfs.ext3
```

TIP A `PATH`-ban megtalálható mappák megkereséséhez használjuk az `echo $PATH` parancsot. Ez kiírja (`echo`) a `PATH` változó (`$PATH`) tartalmát a terminálra.

A `type` egy hasonló parancs, amely információkat mutat egy bináris állományról, beleértve annak helyét és típusát. Csak használjuk a `type` parancsot a parancs neve után:

```
$ type locate
locate is /usr/bin/locate
```

Az `-a` paraméter ugyanúgy működik, mint a `which` paraméter, az összes olyan elérési utat mutatja, amely megfelel a futtatható fájlnak. Például így:

```
$ type -a locate
locate is /usr/bin/locate
locate is /bin/locate
```

A `-t` paraméter pedig megmutatja a parancs fájltípusát, amely lehet `alias`, `keyword`, `function`, `builtin` vagy `file`. Például:

```
$ type -t locate
```

```
file

$ type -t ll
alias

$ type -t type
type is a built-in shell command
```

A `whereis` parancs sokoldalúbb, és a binárisok mellett arra is használható, hogy megmutassa a man oldalak vagy akár a program forráskódjának helyét (ha a rendszerben elérhető). Csak írjuk be a `whereis` parancsot, majd a bináris nevét:

```
$ whereis locate
locate: /usr/bin/locate /usr/share/man/man1/locate.1.gz
```

A fenti eredmények tartalmazzák a bináris állományokat (`/usr/bin/locate`) és a tömörített man oldalakt (`/usr/share/man/man1/locate.1.gz`).

Az eredményeket gyorsan szűrhetjük az olyan parancssori kapcsolókkal, mint a `-b`, amely csak a bináris állományokra, az `-m`, amely csak a man oldalakra, vagy az `-s`, amely csak a forráskódra korlátozza a keresést. A fenti példát megismételve a következőket kapjuk:

```
$ whereis -b locate
locate: /usr/bin/locate

$ whereis -m locate
locate: /usr/share/man/man1/locate.1.gz
```

Gyakorló feladatok

1. Képzeljük el, hogy egy programnak egyszer használatos ideiglenes fájlt kell létrehoznia, amelyre a program bezárása után soha többé nem lesz szükség. Mi lenne az a mappa, ahol ezt a fájlt létrehozhatnánk?

2. Melyik az az ideiglenes mappa, amelyet törölni *kell* a rendszerindítás során?

3. A `find` használatával keressünk csak az aktuális mappában a felhasználó által írható, az elmúlt 10 napban módosított és 4 GiB-nál nagyobb méretű fájlokat!

4. A `locate` segítségével keresseünk meg minden olyan fájlt, amelynek a nevében szerepel a `report` és az `updated`, `update` vagy `updating` mintázat!

5. Hogyan lehet megtalálni, hogy hol van az `ifconfig` man oldala tárolva?

6. Milyen változót kell hozzáadni az `/etc/updatedb.conf'-hoz, hogy az `updatedb` figyelmen kívül hagyja az `ntfs` fájlrendszereket?

7. A rendszergazda egy belső lemezt (`/dev/sdc1`) szeretne csatolni. Az FHS szerint melyik mappa alá kell ezt a lemezt felcsatolni?

Gondolkodtató feladatok

1. A `locate` használatakor az eredmények az `updatedb` által generált adatbázisból származnak. Ez az adatbázis azonban elavult lehet, ami miatt a `locate` olyan fájlokat mutat, amelyek már nem léteznek. Hogyan lehet elérni, hogy a `locate` csak létező fájlokat jelenítsen meg?

2. Keressünk meg minden olyan fájlt az aktuális mappában vagy legfeljebb 2 szinttel lejjebb lévő almappákban, kivéve a csatlakoztatott fájlrendszereket, amelyek nevében szerepel a `Status` vagy `statute` minta!

3. A keresést az `ext4` fájlrendszerekre korlátozva keressünk meg minden olyan fájlt az `/mnt` alatt, amely legalább a csoport számára futtatási jogosultsággal rendelkezik, az aktuális felhasználó számára olvasható, és amelynek bármely attribútuma megváltozott az elmúlt 2 órában!

4. Keressük meg azokat a fájlokat, amelyek 30 napnál korábban lettek létrehozva, üresek és amelyek legalább két szinttel lejjebb vannak az aktuális mappától!

5. Tekintsük úgy, hogy a `carol` és `john` felhasználók az `mkt` csoport tagjai! Keressük meg `john` home könyvtárában azokat a fájlokat, amelyek `carol` számára is olvashatók!

Összefoglalás

Ebben a leckében megismerkedtünk a Linuxon lévő fájlrendszer alapvető szervezésével az FHS szerint, valamint azzal, hogyan lehet bináris állományokat és fájlokat találni, akár név, akár attribútumok alapján. A következő parancsokat tárgyaltuk ebben a leckében:

`find`

Sokoldalú parancs fájlok és mappák keresésére, különböző keresési kritériumok alapján.

`locate`

Egy segédprogram, amely egy helyi adatbázist használ, amely a helyileg tárolt fájlok helyét tartalmazza.

`updatedb`

A `locate` parancs által használt helyi adatbázis frissítése.

`which`

Megjeleníti egy futtatható fájl teljes elérési útvonalát.

`whereis`

Megjeleníti a man oldalak, bináris programok és forráskódok helyét a rendszerben.

`type`

Megjeleníti egy bináris állomány helyét és az alkalmazás típusát (például egy telepített program, egy beépített Bash program és így tovább).

Válaszok a gyakorló feladatokra

1. Képzeljük el, hogy egy programnak egyszer használatos ideiglenes fájlt kell létrehoznia, amelyre a program bezárása után soha többé nem lesz szükség. Mi lenne az a mappa, ahol ezt a fájlt létrehozhatnánk?

   Mivel a program futása után nem érdekel minket a fájl, a helyes mappa a `/tmp`.

2. Melyik az az ideiglenes mappa, amelyet törölni *kell* a rendszerindítás során?

   Ez a mappa a `/run`, vagy néhány rendszer esetén `/var/run`.

3. A `find` használatával keressünk csak az aktuális mappában a felhasználó által írható, az elmúlt 10 napban módosított és 4 GiB-nál nagyobb méretű fájlokat!

   Ehhez a `-writable`, `-mtime` és `-size` paraméterek kellenek:

   ```
   find . -writable -mtime -10 -size +4G
   ```

4. A `locate` segítségével keresseünk meg minden olyan fájlt, amelynek a nevében szerepel a `report` és az `updated`, `update` vagy `updating` mintázat!

   Mivel a `locate` esetén minden mintának egyeznie kell, használjuk az `-A` kapcsolót:

   ```
   locate -A "report" "updat"
   ```

5. Hogyan lehet megtalálni, hogy hol van az `ifconfig` man oldala tárolva?

   Használjuk a `whereis -m` paraméterét:

   ```
   whereis -m ifconfig
   ```

6. Milyen változót kell hozzáadni az `/etc/updatedb.conf'-hoz, hogy az `updatedb` figyelmen kívül hagyja az `ntfs` fájlrendszereket?

   A változó a `PRUNEFS=`, majd utána a fájlrendszer típusa: `PRUNEFS=ntfs`

7. A rendszergazda egy belső lemezt (`/dev/sdc1`) szeretne csatolni. Az FHS szerint melyik mappa alá kell ezt a lemezt felcsatolni?

   A gyakorlatban a lemez bárhová felcsatolható. Az FHS azonban azt ajánlja, hogy az ideiglenes

csatolásokat az `/mnt` alatt végezzük.

Válaszok a gondolkodtató feladatokra

1. A `locate` használatakor az eredmények az `updatedb` által generált adatbázisból származnak. Ez az adatbázis azonban elavult lehet, ami miatt a `locate` olyan fájlokat mutat, amelyek már nem léteznek. Hogyan lehet elérni, hogy a `locate` csak létező fájlokat jelenítsen meg?

   Adjuk hozzá az `-e` paramétert a locate-hez, mint: `locate -e PATTERN`.

2. Keressünk meg minden olyan fájlt az aktuális mappában vagy legfeljebb 2 szinttel lejjebb lévő almappákban, kivéve a csatlakoztatott fájlrendszereket, amelyek nevében szerepel a `Status` vagy `statute` minta!

   Ne feledjük, hogy a `-maxdepth` esetében figyelembe kell venni az aktuális mappát is, tehát három szintet akarunk (az aktuális, plusz 2 szinttel lejjebb):

   ```
   find . -maxdepth 3  -mount -iname "*statu*"
   ```

3. A keresést az `ext4` fájlrendszerekre korlátozva keressünk meg minden olyan fájlt az `/mnt` alatt, amely legalább a csoport számára futtatási jogosultsággal rendelkezik, az aktuális felhasználó számára olvasható, és amelynek bármely attribútuma megváltozott az elmúlt 2 órában!

   A `mount` paraméter `-fstype` paraméterének használatával a keresést bizonyos fájlrendszer-típusokra korlátozhatjuk. Az aktuális felhasználó által olvasható fájlnak legalább `4`-nek kell lennie a jogosultságok első számjegyében, a csoport által futtathatónak pedig legalább `1`-nek a második számjegyében. Mivel mások jogosultságai nem érdekelnek minket, a harmadik számjegyre `0`-t használhatunk. A `-cmin N` használatával szűrhetjük a legutóbbi attribútumváltozásokat, ne feledjük, hogy az N értéket percekben adjuk meg. Tehát:

   ```
   find /mnt -fstype ext4 -perm -410 -cmin -120
   ```

4. Keressük meg azokat a fájlokat, amelyek 30 napnál korábban lettek létrehozva, üresek és amelyek legalább két szinttel lejjebb vannak az aktuális mappától!

   A `-mindepth N` paraméterrel a keresést legalább `N` szinttel lejjebb korlátozhatjuk, de ne feledjük, hogy az aktuális mappát is bele kell számolnunk a számolásba. Az `-empty` az üres fájlok, az `-mtime N` pedig a módosítási idő kereséséhez használható. Tehát:

   ```
   find . -empty -mtime +30 -mindepth 3
   ```

5. Tekintsük úgy, hogy a `carol` és `john` felhasználók az `mkt` csoport tagjai! Keressük meg `john` home könyvtárában azokat a fájlokat, amelyek `carol` számára is olvashatók!

   Tekintettel arra, hogy ezek ugyanannak a csoportnak a tagjai, legalább egy `r` (4) kell a csoportjogosultságokhoz, és nem törődünk másokkal. Tehát:

   ```
   find /home/john -perm -040
   ```

Impresszum

PDF generálva: 2023-08-22

Az LPI Learning Materials a Linux Professional Institute (https://lpi.org) kezdeményezése. A tananyagok és a fordításaik a https://learning.lpi.org oldalon érhetők el.

Ezzel a kiadással és az egész projekttel kapcsolatos kérdéseket és megjegyzéseket az alábbi e-mail címre küldheti el: learning@lpi.org.