Linux
Professional
Institute

LPIC-1

Verzió 5.0
Magyar

102

Table of Contents

Linux Professional Institute

Témakör 105: Shellek és shell scriptelés

105.1 A shell testreszabása és használata

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 102, Objective 105.1

Súlyozás

4

Kulcsfontosságú ismeretek

- Környezeti változók (pl. PATH) beállítása bejelentkezéskor vagy egy új shell indításakor
- Bash függvények írása gyakran használt parancsok szekvenciájára
- Skeleton mappák fenntartása új felhasználói fiókok számára
- Parancskeresési útvonal beállítása a megfelelő mappával

A használt fájlok, kifejezések és segédprogramok listája

- `.`
- `source`
- `/etc/bash.bashrc`
- `/etc/profile`
- `env`
- `export`
- `set`
- `unset`
- `~/.bash_profile`
- `~/.bash_login`

- `~/.profile`
- `~/.bashrc`
- `~/.bash_logout`
- `function`
- `alias`

105.1 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	105 Shellek és shell scriptelés
Fejezet:	105.1 A shell testre szabása és használata
Lecke:	1/3

Bevezetés

A shell vitathatatlanul a legerősebb eszköz a Linux rendszerben és úgy definiálható, mint a felhasználó és az operációs rendszer magja közötti interfész. A felhasználó által beírt parancsokat értelmezi, ezért minden rendszergazdának jártasnak kell lennie a shell használatában. Amint azt mostanra már bizonyára tudjuk, a Bourne Again Shell (*Bash*) a Linux disztribúciók túlnyomó többségének *de facto* shellje.

Az első dolog, amit a Bash — vagy bármely más parancsértelmező — indítás után csinál, az egy sor startup script végrehajtása. Ezek a scriptek testre szabják a munkamenet környezetét. Vannak rendszerszintű és felhasználó-specifikus scriptek is. Ezekben a scriptekben a személyes preferenciáinkat vagy a felhasználók igényeinek leginkább megfelelő beállításokat helyezhetjük el változók, aliasok és függvények formájában.

Az indítófájlok pontos sorozata egy nagyon fontos paramétertől függ: a shell típusától. Nézzük meg, hogy milyen sokféle shell létezik!

Shell-típusok: Interaktív vs. Non-Interaktív és Login vs. Non-Login

Először is tisztázzuk az *interaktív* és a *login* (bejelentkezés) fogalmát a shellek kontextusában:

Interaktív / non-interaktív shellek

Ez a fajta shell a felhasználó és a shell között zajló interakcióra utal: A felhasználó a terminálba a billentyűzet segítségével parancsokat ír be, a shell pedig a képernyőre kiírt üzenetekkel ad kimenetet.

Login / non-login shellek

Ez a fajta shell arra az esetre utal, amikor egy felhasználó belép egy számítógépes rendszerbe, megadva a hitelesítő adatokat, például a felhasználónevet és a jelszót.

Mind az interaktív és non-interaktív shellek lehetnek login vagy non-login shellek és e típusok minden lehetséges kombinációjának megvan a saját felhasználási módja.

Az *interaktív login shellek* akkor kerülnek végrehajtásra, amikor a felhasználók bejelentkeznek a rendszerbe, és arra szolgálnak, hogy a felhasználók konfigurációit az igényeiknek megfelelően alakítsák. Jó példa az ilyen típusú shellre az azonos osztályhoz tartozó felhasználók egy csoportja, akiknek egy adott változó beállítására van szükségük a munkameneteikben.

Az *interaktív, non-login shellek* alatt minden más shellt értünk, amelyet a felhasználó a rendszerbe való bejelentkezés után megnyit. A felhasználók ezeket a shelleket a munkamenetek során karbantartási és adminisztrációs feladatok elvégzésére használják, mint például változók vagy az idő beállítása, fájlok másolása, scriptek írása stb.

Másrészt a *non-interaktív shellek* nem igényelnek semmiféle emberi beavatkozást. Így ezek a shellek nem kérnek a felhasználótól bemenetet, és a kimenetük — ha van — a legtöbb esetben egy logba íródik.

A *non-interaktív login shellek* meglehetősen ritkák és nem is praktikusak. Felhasználásuk gyakorlatilag nincs példa, és csak a shell viselkedésébe való betekintés kedvéért fogunk róluk nyilatkozni. Néhány furcsa példa: egy script futtatásának kikényszerítése a bejelentkezési shellből a `/bin/bash --login <some_script>` paranccsal, vagy egy parancs standard kimenetének (*stdout*) átvezetése egy ssh kapcsolat standard bemenetére (*stdin*):

```
<some_command> | ssh <some_user>@<some_server>
```

A *non-interaktív, non-login shell* esetében nincs sem interakció, sem bejelentkezés a felhasználó

részéről, tehát itt az automatizált scriptek használatáról van szó. Ezeket a scripteket többnyire ismétlődő adminisztrációs és karbantartási feladatok elvégzésére használják, mint például a cronjobokban lévők. Ilyen esetekben a `bash` nem olvas be semmilyen indítófájlt.

Egy terminál megnyitása

Amikor asztali környezetben vagyunk, megnyithatunk egy terminálalkalmazást, vagy átválthatunk valamelyik rendszerkonzolra. Ezért egy új shell vagy `pts` shell, ha a GUI-ban egy terminál emulátorból nyitjuk meg, vagy `tty` shell, ha egy rendszerkonzolról futtatjuk. Az első esetben nem terminállal, hanem terminálemulátorral van dolgunk. A grafikus munkamenetek részeként az olyan terminálemulátorok, mint a *gnome-terminal* vagy a *konsole*, nagyon gazdag funkciójúak és felhasználóbarátok a szöveges felhasználói felületű terminálokhoz képest. A kevésbé gazdag funkciójú terminál emulátorok közé tartozik — többek között — az *XTerm* és a *sakura*.

A `Ctrl` + `Alt` + `F1` - `F6` kombinációkkal a konzolos bejelentkezésekhez léphetünk, amelyek egy interaktív szöveges login shellt nyitnak meg. `Ctrl` + `Alt` + `F7` visszaváltja a munkamenetet a Desktopra.

NOTE A `tty` a teletypewritter; a `pts` pedig a pseudo terminal slave rövidítése. További információk: `man tty` és `man pts`.

Shellek indítása a `bash` használatával

Bejelentkezés után írjuk be a terminálba a `bash` parancsot, hogy megnyissunk egy új parancsértelmezőt. Gyakorlatilag ez a shell az aktuális shell gyermekprocessze.

A `bash` gyermekprocessz indításakor különböző kapcsolókkal határozhatjuk meg, hogy milyen shellt akarunk indítani. Íme néhány fontos `bash` invokáló/indítási opció:

`bash -l` or `bash --login`
: Egy login shellt invokál.

`bash -i`
: Egy interaktív shellt invokál.

`bash --noprofile`
: A login shellekkel figyelmen kívül hagyja mind a rendszerszintű `/etc/profile` startup fájlt, mind a felhasználói szintű `~/.bash_profile`, `~/.bash_login` és `~/.profile` startup fájlokat.

`bash --norc`
: Az interaktív shellekkel figyelmen kívül hagyja mind a rendszerszintű `/etc/bash.bashrc`

startup fájlt, mind a felhasználói szintű `~/.bashrc` startup fájlt.

`bash --rcfile <file>`

Az interaktív shellekkel a `<file>`-t veszi startup fájlnak, figyelmen kívül hagyva a rendszerszintű `/etc/bash.bashrc`-t és a felhasználói szintű `~/.bashrc`-t.

Az alábbiakban a különböző startup fájlokat tárgyaljuk.

Shellek indítása a `su` és `sudo` segítségével

E két hasonló program használatával különleges shell-típusokat kaphatunk:

`su`

Felhasználói azonosító módosítása vagy szuperfelhasználóvá válás (`root`). Ezzel a paranccsal mind a login, mind a non-login shelleket meg tudjuk hívni:

- A `su - user2`, `su -l user2` vagy `su --login user2` egy interaktív login shellt indít el `user2`-ként.
- A `su user2` egy interaktív, non-loign shellt indít el `user2`-ként.
- `su - root` vagy `su -` egy interaktív login shellt indít el `root`-ként.
- `su root` vagy `su` egy interaktív, non-login shellt indít el `root`-ként.

`sudo`

Hajtsuk végre a parancsokat más felhasználóként (beleértve a szuperfelhasználót is). Mivel ezt a parancsot elsősorban a root jogosultságok ideiglenes megszerzésére használjuk, a parancsot használó felhasználónak szerepelnie kell a `sudoers` fájlban. Ahhoz, hogy felhasználókat adjunk hozzá a `sudoers` állományhoz, `root`-nak kell lennünk, majd futtatnunk kell a következőt

```
root@debian:~# usermod -aG sudo user2
```

A `su`-hoz hasonlóan a `sudo` is lehetővé teszi, hogy login és non-login shelleket egyaránt használjunk:

- A `sudo su - user2`, `sudo su -l user2` vagy `sudo su --login user2` egy interaktív login shellt indít el `user2`-ként.
- A `sudo su user2` egy interaktív, non-login shellt indít el `user2`-ként.
- A `sudo -u user2 -s` egy interaktív, non-login shellt indít el `user2`-ként.
- A `sudo su - root` vagy `sudo su -` egy interaktív login shellt indít el `root`-ként.

- A `sudo -i` egy interaktív logint shellt indít el `root`-ként.
- `sudo -i <some_command>` interaktív login shellt indít `root`-ként, lefuttatja a parancsot, és visszatér az eredeti felhasználóhoz.
- A `sudo su root` vagy `sudo su` egy interaktív, non-login shellt indít el `root`-ként.
- A `sudo -s` vagy a `sudo -u root -s` egy non-login shellt indít el `root`-ként.

A `su` vagy a `sudo` használatakor fontos figyelembe venni az új shell indításának sajátos forgatókönyvét: szükségünk van-e a célfelhasználó környezetére vagy sem? Ha igen, akkor azokat a kapcsolókat használjuk, amelyek a login shelleket hívják meg; ha nem, akkor azokat, amelyek a non-login shelleket hívják meg.

Milyen shell típusaink vannak?

Ahhoz, hogy megtudjuk, milyen típusú shellel dolgozunk, beírhatjuk a `echo $0` parancsot a terminálba, és a következő kimenetet kapjuk:

Interaktív login

`-bash` vagy `-su`

Interaktív non-login

`bash` vagy `/bin/bash`

Non-interaktív non-login (scriptek)

`<name_of_script>`

Hány shellünk van?

Ha meg akarjuk nézni, hogy hány `bash` shell fut a rendszeren, használhatjuk a `ps aux | grep bash` parancsot:

```
user2@debian:~$ ps aux | grep bash
user2       5270  0.1  0.1  25532  5664 pts/0    Ss   23:03   0:00 bash
user2       5411  0.3  0.1  25608  5268 tty1     S+   23:03   0:00 -bash
user2       5452  0.0  0.0  16760   940 pts/0    S+   23:04   0:00 grep --color=auto bash
```

A `user2` a `debian`-nál bejelentkezett egy GUI (vagy X Window System) munkamenetbe és megnyitotta a *gnome-terminalt*, majd megnyomta a `Ctrl` + `Alt` + `F1` billentyűt, hogy egy `tty` terminál munkamenetbe lépjen. Végül a `Ctrl` + `Alt` + `F7` billentyűkombinációval visszalépett a GUI munkamenetbe és beírta a `ps aux | grep bash` parancsot. Így a kimenet egy interaktív, non-

login shellt mutat a terminálemulátoron keresztül (`pts/0`) és egy interaktív login shellt a megfelelő szöveges terminálon keresztül (`tty1`). Figyeljük meg azt is, hogy az egyes sorok utolsó mezője (a parancs) az előbbinél `bash`, az utóbbinál pedig `-bash`!

Honnan kapják a shellek a konfigurációjukat: a startup fájlok

Most, hogy már ismerjük a Linux rendszerben megtalálható shelltípusokat, itt az ideje, hogy megnézzük, milyen startup fájlokat milyen shell hajt végre! Vegyük figyelembe, hogy a rendszerszintű vagy globális scriptek a `/etc/` mappában találhatók, míg a helyi vagy felhasználói szintűek a felhasználó home (`~`) mappájában! Továbbá, ha egynél több fájlt kell keresni, ha az egyiket megtaláltuk és lefuttattuk, a többit figyelmen kívül hagyjuk. Fedezzük fel és tanulmányozzuk ezeket a fájlokat a kedvenc szövegszerkesztőnkkel vagy a `less <startup_file>` beírásával!

NOTE A startup fájlok Bash-specifikus (csak a `bash` konfigurációkra és parancsokra korlátozódó) és általános (a legtöbb shellre vonatkozó) fájlokra oszthatók.

Interaktív login shell

Globális szint (Global Level)

`/etc/profile`

Ez az egész rendszerre kiterjedő `.profile` fájl a Bourne shell és a Bourne kompatibilis shellek (beleértve a `bash`-t is) számára. Ez a fájl egy sor `if` utasításon keresztül számos változót állít be, mint például a `PATH` és `PS1`, valamint — ha létezik — az `/etc/bash.bashrc` és az `/etc/profile.d` mappában lévő változókat.

`/etc/profile.d/*`

Ez a mappa olyan scripteket tartalmazhat, amelyeket az `/etc/profile` hajt végre.

Lokális szint (Local Level)

`~/.bash_profile`

Ez a Bash-specifikus fájl a felhasználói környezet konfigurálására szolgál. A `~/.bash_login` és a `~/.profile` forrásaként is használható.

`~/.bash_login`

Szintén Bash-specifikus, ez a fájl csak akkor lesz végrehajtva, ha nincs `~/.bash_profile` fájl. A neve arra utal, hogy a bejelentkezéskor szükséges parancsok futtatására kell használni.

`~/.profile`

Ez a fájl nem Bash-specifikus, és csak akkor lesz forrás, ha sem a `~/.bash_profile`, sem a `~/.bash_login` nem létezik — ami általában így van. Így a `~/.profile` fő célja az, hogy ellenőrizze, hogy egy Bash shell fut-e, és — ha igen — a `~/.bashrc`-t is forrásként használja, ha létezik. Általában úgy állítja be a `PATH` változót, hogy az tartalmazza a felhasználó privát `~/bin` mappáját, ha az létezik.

`~/.bash_logout`

Ha létezik, ez a Bash-specifikus fájl a shell kilépésekor néhány takarítási műveletet végez. Ez például távoli munkamenetek esetén lehet kényelmes.

Interaktív login shell konfigurációs fájlok vizsgálata

Hadd mutassunk meg néhány ilyen fájlt működés közben a `/etc/profile` és a `/home/user2/.profile` módosításával. Mindegyikhez csatolunk egy-egy sort, amely emlékeztet minket a végrehajtott fájlra:

```
root@debian:~# echo 'echo Hello from /etc/profile' >> /etc/profile
root@debian:~# echo 'echo Hello from ~/.profile' >> ~/.profile
```

NOTE A két átirányítási operátor `>>` egy parancs kimenetét egy meglévő fájlba illeszti — nem írja felül azt. Ha azonban a fájl nem létezik, akkor létrehozza.

Így a megfelelő `echo` parancsok kimenetén keresztül tudjuk, hogy mikor olvassa és hajtja végre az egyes fájlokat. Ennek bizonyítására nézzük meg, mi történik, amikor `user2` bejelentkezik `ssh`-n keresztül egy másik gépről:

```
user2@debian:~$ ssh user2@192.168.1.6
user2@192.168.1.6's password:
Linux debian 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Nov 27 19:57:19 2018 from 192.168.1.10
Hello from /etc/profile
Hello from /home/user2/.profile
```

Amint az utolsó két sorból is látható, ez működött, de vegyünk észre három dolgot:

- Először a globális fájl futott le.
- Nem voltak `.bash_profile` vagy `.bash_login` fájlok a `user2` home mappájában.
- A tilde (`~`) a fájl abszolút elérési útvonalára (`/home/user2/.profile`) bővült.

Interaktív non-login shell

Globális szint

`/etc/bash.bashrc`

Ez az egész rendszerre kiterjedő `.bashrc` fájl az interaktív `bash` shellekhez. Végrehajtása során a `bash` meggyőződik arról, hogy interaktívan fut, minden parancs után ellenőrzi az ablak méretét (szükség esetén frissíti a `LINES` és `COLUMNS` értékeit), és beállít néhány változót.

Lokális szint

`~/.bashrc`

Amellett, hogy a `/etc/bash.bashrc` felhasználói szinten hasonló feladatokat lát el (például az ablak méretének ellenőrzése, vagy ha interaktívan futtatjuk), ez a Bash-specifikus fájl általában beállít néhány előzményváltozót és a `~/.bash_aliases` forrását, ha létezik. Ettől eltekintve ez a fájl általában a felhasználó specifikus aliasok és függvények tárolására szolgál.

Hasonlóképpen érdemes megjegyezni, hogy a `~/.bashrc` beolvasásra kerül, ha a `bash` érzékeli, hogy az `<stdin>` egy hálózati kapcsolat (mint a fenti példában a *Secure Shell* (SSH) kapcsolat esetében).

Interaktív, non-login shell konfigurációs fájlok vizsgálata

Módosítsuk a `/etc/bash.bashrc` és a `/home/user2/.bashrc` fájlokat:

```
root@debian:~# echo 'echo Hello from /etc/bash.bashrc' >> /etc/bash.bashrc
root@debian:~# echo 'echo Hello from ~/.bashrc' >> ~/.bashrc
```

És ez történik, amikor `user2` egy új shell-t indít:

```
user2@debian:~$ bash
Hello from /etc/bash.bashrc
Hello from /home/user2/.bashrc
```

A két fájlt ismét beolvasásra és végrehajtásra került.

WARNING Ne feledjük, hogy a fájlok futtatási sorrendje miatt a helyi fájlok elsőbbséget élveznek a globálisakkal szemben.

Non-Interaktív Login Shell

Az `-l` vagy `--login` kapcsolóval rendelkező non-interaktív shell kénytelen úgy viselkedni, mint egy login shell, így a futtatandó indítófájlok ugyanazok lesznek, mint az interaktív login shell esetében.

Ennek bizonyítására írjunk egy egyszerű scriptet, és tegyük futtathatóvá. Nem fogunk semmilyen *shebang*-ot beilleszteni, mert a parancssorból a *bash executablet* (`/bin/bash` a login kapcsolóval) fogjuk meghívni.

1. Létrehozzuk a `test.sh` scriptet, amely tartalmazza a `echo 'Hello from a script'` sort, hogy bizonyítani tudjuk a script sikeres futását:

```
user2@debian:~$ echo "echo 'Hello from a script'" > test.sh
```

2. Futtathatóvá tesszük a scriptet:

```
user2@debian:~$ chmod +x ./test.sh
```

3. Végül meghívjuk a `bash`-t az `-l` opcióval, hogy futtassuk a scriptet:

```
user2@debian:~$ bash -l ./test.sh
Hello from /etc/profile
Hello from /home/user2/.profile
Hello from a script
```

   Működik! A script futtatása előtt megtörtént a bejelentkezés, és az `/etc/profile` és a `~/.profile` is végrehajtásra került.

NOTE A *shebang*-ekről és a shell scriptelés minden más aspektusáról a következő leckékben fogunk tanulni.

Legyen most az `echo` parancs standard kimenete (*stdout*) az `ssh` kapcsolat standard bemenetére irányítva (*stdin*) egy pipe (`|`) segítségével:

```
user2@debian:~$ echo "Hello-from-a-noninteractive-login-shell" | ssh user2@192.168.1.6
Pseudo-terminal will not be allocated because stdin is not a terminal.
user2@192.168.1.6's password:
Linux debian 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Hello from /etc/profile
Hello from /home/user2/.profile
-bash: line 1: Hello-from-a-noninteractive-login-shell: command not found
```

Az `/etc/profile` és a `~/.profile` ismét végrehajtásra kerül. Ettől eltekintve a kimenet első és utolsó sora elég sokatmondó a shell viselkedését illetően.

Non-Interaktív Non-Login Shell

A scriptek nem olvassák a fent felsorolt fájlok egyikét sem, hanem keresik a `BASH_ENV` környezeti változót, szükség esetén kibővítik az értékét, és azt használják egy indítófájl neveként a parancsok beolvasásához és végrehajtásához. A *környezeti változókról* a következő leckében többet fogunk megtudni.

Mint fentebb említettük, általában az `/etc/profile` és a `~/.profile` gondoskodik arról, hogy a sikeres bejelentkezés után mind az `/etc/bash.bashrc`, mind a `~/.bashrc` végrehajtásra kerüljön. A következő parancs kimenete megmutatja ezt a jelenséget:

```
root@debian:~# su - user2
Hello from /etc/bash.bashrc
Hello from /etc/profile
Hello from /home/user2/.bashrc
Hello from /home/user2/.profile
```

Figyelembe véve a korábban a startup scriptekhez csatolt sorokat — és az interaktív bejelentkezési shell meghívását felhasználói szinten a `su - user2` paranccsal — a négy kimeneti sor a következőképpen magyarázható:

1. `Hello from /etc/bash.bashrc` azt jelenti, hogy az `/etc/profile` a `/etc/bash.bashrc` forrásából származik.

2. `Hello from /etc/profile` azt jelenti, hogy az `/etc/profile` teljes egészében beolvasásra és végrehajtásra került.
3. `Hello from /home/user2/.bashrc` azt jelenti, hogy a `~/.profile` a `~/.bashrc`-t vette át.
4. `Hello from /home/user2/.profile` azt jelenti, hogy a `~/.profile` teljes egészében beolvasásra és végrehajtásra került.

Figyeljük meg, hogy a `su - <felhasználónév>` (valamint a `su -l <felhasználónév>` és `su --login <felhasználónév>`) használatával garantáljuk a login shell meghívását, míg a `su <felhasználónév>` csak az `/etc/bash.bashrc`-t és a `~/.bashrc`-t hívta volna meg.

Sourcing fájlok

Az előző szakaszokban már beszéltünk arról, hogy egyes indítási parancsfájlok más parancsfájlokat tartalmaznak vagy hajtanak végre. Ezt a mechanizmust nevezzük sourcingnek, és ebben a szakaszban ismertetjük.

Sourcing fájlok és a `.`

A pont (`.`) általában a startup fájlokban található.

A Debian szerverünk `.profile` fájljában például a következő blokkot találjuk:

```
# include .bashrc if it exists
if [ -f "$HOME/.bashrc" ]; then
. "$HOME/.bashrc"
fi
```

Már láttuk, hogy egy script végrehajtása hogyan vezethet egy másik script végrehajtásához. Így az `if` utasítás garantálja, hogy a `$HOME/.bashrc` fájl — ha létezik (`-f`) — a bejelentkezéskor sourced (azaz olvasás és végrehajtás) lesz:

```
. "$HOME/.bashrc"
```

NOTE Amint azt a következő leckében megtanuljuk, a `$HOME` egy olyan környezeti változó, amely a felhasználó home mappájában abszolút elérési útvonalára bővül.

Továbbá használhatjuk a `.`-t, ha módosítottunk egy startup fájlt, és a változtatásokat újraindítás nélkül akarjuk érvénybe léptetni. Például:

- alias hozzáadása a `~/.bashrc`-hez:

```
user2@debian:~$ echo "alias hi='echo We salute you.'" >> ~/.bashrc
```

WARNING Amikor egy parancs kimenetét egy fájlba küldjük, ne tévesszük össze az append (hozzáfűzés — `>>`) és az overwrite (felülírás — `>`) parancsokat.

- a `~/.bashrc` utolsó sorának kimenete, hogy ellenőrizzük, minden rendben ment-e:

```
user2@debian:~$ tail -n 1 !$
tail -n 1 ~/.bashrc
alias hi='echo We salute you.'
```

NOTE A `!$` az előző parancs utolsó argumentumára bővül, ami a mi esetünkben a "~/.bashrc".

- a fájl sourceolása kézzel:

```
user2@debian:~$ . ~/.bashrc
```

- és az alias meghívása, hogy bizonyítsuk, működik:

```
user2@debian:~$ hi
We salute you.
```

NOTE A következő leckében megismerkedünk az *aliasokkal* és a *változókkal.*

Fájlok sourcingja a `source` paranccsal

A `source` parancs a `.` szinonimája. Tehát a `~/.bashrc` sourcingolását így is megtehetjük:

```
user2@debian:~$ source ~/.bashrc
```

A shell startup fájlok eredete: `SKEL`

A `SKEL` egy változó, amelynek értéke a `skel` mappa abszolút elérési útvonala. Ez a mappa szolgál sablonként a felhasználók home mappáinak fájlrendszeri struktúrájához. Tartalmazza azokat a fájlokat, amelyeket minden újonnan létrehozott felhasználói fiók örökölni fog (beleértve természetesen a shellek konfigurációs fájljait is). A `SKEL` és más kapcsolódó változókat az `/etc/adduser.conf` állományban tároljuk, amely az `adduser` konfigurációs állománya:

```
user2@debian:~$ grep SKEL /etc/adduser.conf
# The SKEL variable specifies the directory containing "skeletal" user
SKEL=/etc/skel
# If SKEL_IGNORE_REGEX is set, adduser will ignore files matching this
SKEL_IGNORE_REGEX="dpkg-(old|new|dist|save)"
```

A `SKEL` a `/etc/skel`-be van állítva; így a shelljeinket konfiguráló startup scriptek is ebben találhatók:

```
user2@debian:~$ ls -a /etc/skel/
.  ..  .bash_logout  .bashrc  .profile
```

WARNING Ne feledjük, hogy a `.`-al kezdődő fájlok el vannak rejtve, ezért a mappák tartalmának listázásakor az `ls -a` parancsot kell használnunk, hogy lássuk őket!

Most hozzunk létre egy mappát az `/etc/skel`-ben, ahol minden új felhasználó tárolhatja a személyes scriptjeit:

1. `root`-ként lépünk az `/etc/skel`-be:

```
root@debian:~# cd /etc/skel/
root@debian:/etc/skel#
```

2. Kilistázzuk a tartalmát:

```
root@debian:/etc/skel# ls -a
.  ..  .bash_logout  .bashrc  .profile
```

3. Létrehozzuk a mappát és ellenőrizzük, hogy minden az elképzelésnek megfelelően alakult-e:

```
root@debian:/etc/skel# mkdir my_personal_scripts
root@debian:/etc/skel# ls -a
.  ..  .bash_logout  .bashrc  my_personal_scripts  .profile
```

4. Most töröljük a `user2`-t a `home` mappájával együtt:

```
root@debian:~# deluser --remove-home user2
```

```
Looking for files to backup/remove ...
Removing files ...
Removing user `user2' ...
Warning: group `user2' has no more members.
Done.
```

5. Hozzáadjuk `user2`-t újra, így megkapja az új home mappáját:

```
root@debian:~# adduser user2
Adding user `user2' ...
Adding new group `user2' (1001) ...
Adding new user `user2' (1001) with group `user2' ...
Creating home directory `/home/user2' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for user2
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] y
```

6. Végül belépünk `user2`-ként és listázzuk az összes fájlt a `/home/user2`-ben, hogy lássuk, minden az elképzeléseinknek megfelelően alakult-e:

```
root@debian:~# su - user2
user2@debian:~$ pwd
/home/user2
user2@debian:~$ ls -a
.  ..  .bash_history  .bash_logout  .bashrc  my_personal_scripts  .profile
```

It did.

Gyakorló feladatok

1. Tanulmányozzuk, hogyan indultak a shellek a “Shell elindult…” oszlopban, és töltsük ki a szükséges információkat:

Shell elindult…	**Interaktív?**	**Login?**	`echo $0` **eredménye**
`sudo ssh user2@machine2`			
`Ctrl` + `Alt` + `F2`			
`su - user2`			
gnome-terminal			
Egy normál felhasználó a *konzolt* használja a *sakura* példányának elindításához			
Egy `test.sh` nevű script, ami az `echo $0` parancsot tartalmazza			

2. A `su` és `sudo` parancsokkal indítsuk el a megadott shellt:

   Interaktív login shell `user2` felhasználóként

   `su`:

   `sudo`:

   Interaktív login shell `root` felhasználóként

   `su`:

   `sudo`:

   Interaktív non-login shell `root` felhasználóként

   `su`:

   `sudo`:

Interaktív non-login shell `user2` felhasználóként

`su`:

`sudo`:

3. Milyen startup fájl kerül beolvasásra, amikor a “Shell típusa” alatti shell elindul?

Shell típusa	`/etc/profile`	`/etc/bash.bashrc`	`~/.profile`	`~/.bashrc`
Interaktív-login shell `user2` felhasználóként				
Interaktív login shell `root` felhasználóként				
Interaktív non-login shell `root` felhasználóként				
Interaktív non-login shell `user2` felhasználóként				

Gondolkodtató feladatok

1. Bashben írhatunk egy egyszerű `Hello world!` függvényt, ha a következő kódot beillesztjük egy üres fájlba:

```
function hello() {
     echo "Hello world!"
}
```

   - Mi a következő teendőnk, hogy a funkciót elérhetővé tegyük a shell számára?

   - Ha már elérhető az aktuális shell számára, hogyan lehet invokálni?

   - Az automatizálás érdekében melyik fájlba tehetnénk a függvényt és annak meghívását, hogy az akkor is végrehajtódjon, amikor a `user2` megnyit egy terminált egy X Window munkamenetből? Milyen típusú shellről van szó?

   - Milyen fájlban helyezhetnénk el a függvényt és annak invokálását, hogy akkor fusson le, amikor a `root` elindít egy új interaktív shellt, függetlenül attól, hogy bejelentkezik-e vagy sem?

2. Nézzük meg az alábbi, egyszerű `Hello world!` `bash` scriptet:

```
#!/bin/bash

#hello_world: egy egyszerű bash script a scriptek interakciójának megvitatására.

echo "Hello world!"
```

   - Tegyük fel, hogy a scriptet futtathatóvá tesszük és futtatjuk. Ez egy interaktív script lenne? Miért?

   - Mi teszi a scriptet interaktívvá?

3. Képzeljük el, hogy megváltoztattuk néhány változó értékét a `~/.bashrc` állományban, és azt szeretnénk, hogy ezek a változások újraindítás nélkül lépjenek életbe! Hogyan lehetne ezt a home mappából kétféleképpen megvalósítani?

4. John most indított el egy X Window munkamenetet egy Linux-szerveren. Megnyit egy terminál emulátort, hogy elvégezzen néhány adminisztrációs feladatot, de meglepő módon a munkamenet lefagy, és meg kell nyitnia egy szöveges shellt.
   - Hogyan nyithatja meg a `tty` shellt?
   - Melyik startup fájl lesz sourceolva?

5. Linda egy Linux-szerver felhasználója. Kéri a rendszergazdát, hogy legyen egy `~/.bash_login` fájlja, hogy bejelentkezéskor a képernyőre kiírhassa az időt és a dátumot. A többi felhasználónak tetszik az ötlet, és követik a példáját. A rendszergazdának nehézséget okoz a szerver összes többi felhasználója számára létrehozni a fájlt, ezért egy új policy hozzáadása mellett dönt, és minden potenciális új felhasználó számára létrehozza a `~/.bash_login` fájlt. Hogyan tudja a rendszergazda ezt a feladatot elvégezni?

Összefoglalás

Ebben a leckében megtanultuk:

- Azt, hogy hogyan állítják be a shellek a felhasználók környezetét egy Linux rendszerben.
- Azt, hogy a *Bash* az első számú shell a GNU/Linux disztribúciókban.
- Azt, hogy a shell első feladata egy vagy több indítófájl beolvasása és végrehajtása.
- Az *interakció* és *login* fogalma a shellekkel kapcsolatban.
- Azt, hogy hogyan indíthatunk különböző típusú shelleket a `bash`, `su`, `sudo` és `Ctrl` + `Alt` + `F1`-`F6` segítségével.
- Azt, hogy hogyan ellenőrizhetjük a shell típusát a `echo $0` segítségével.
- Azt, hogy a helyi indítófájlok a `~/.bash_profile`, a `~/.profile`, a `~/.bash_login`, a `~/.bash_logout` és a `~/.bashrc`.
- Azt, hogy a globális indítási fájlok az `/etc/profile`, az `/etc/profile.d/*` és az `/etc/bash.bashrc`.
- Azt, hogy a helyi fájlok elsőbbséget élveznek a globálisakkal szemben.
- Azt, hogy hogyan irányíthatjuk át egy parancs kimenetét a `>` (felülírás) és a `>>` (hozzáfűzés) segítségével.
- Azt, hogy mi a `skel` mappa jelentése.
- Azt, hogy hogyan kell sourcolni a fájlokat.

A leckében használt parancsok:

`bash`
: Új shellt hoz létre.

`su`
: Új shellt hoz létre.

`sudo`
: Új shellt hoz létre.

`usermod`
: Módosít egy felhasználói fiókot.

`echo`

Megjelenít egy szöveget.

`ps`

A jelenlegi folyamatokról közöl egy pillanatképet.

`less`

Lapozó hosszabb fájlokhoz.

`ssh`

Elindít egy Open SSH kapcsolatot (távolról).

`chmod`

Megváltoztatja egy fájl üzemmódbitjeit, például futtathatóvá teszi.

`grep`

Egy mintának megfelelő sorokat jelenít meg.

`ls`

Kilistázza egy mappa tartalmát.

`cd`

Mappát vált.

`mkdir`

Létrehoz egy mappát.

`deluser`

Töröl egy felhasználót.

`adduser`

Hozzáad egy új felhasználót.

`.`

Sourceol egy fájlt.

`source`

Sourceol egy fájlt.

`tail`

Kiírja a fájlok utolsó részét.

Válaszok a gyakorló feladatokra

1. Tanulmányozzuk, hogyan indultak a shellek a “Shell elindult…” oszlopban, és töltsük ki a szükséges információkat:

Shell elindult…	**Interaktív?**	**Login?**	`echo $0` **eredménye**
`sudo ssh user2@machine2`	Igen	Igen	`-bash`
`Ctrl` + `Alt` + `F2`	Igen	Igen	`-bash`
`su - user2`	Igen	Igen	`-bash`
gnome-terminal	Igen	Nem	`bash`
Egy normál felhasználó a *konzolt* használja a *sakura* példányának elindításához	Igen	Nem	`/bin/bash`
Egy `test.sh` nevű script, ami az `echo $0` parancsot tartalmazza	Nem	Nem	`./test.sh`

2. A `su` és `sudo` parancsokkal indítsuk el a megadott shellt:

   Interaktív login shell `user2` felhasználóként

   `su`

   `su - user2`, `su -l user2` vagy `su --login user2`

   `sudo`

   `sudo su - user2`, `sudo su -l user2` vagy `sudo su --login user2`

   Interaktív login shell `root` felhasználóként

   `su`

   `su - root` vagy `su -`

   `sudo`

   `sudo su - root`, `sudo su -` vagy `sudo -i`

Interaktív non-login shell `root` felhasználóként

su

`su root` vagy `su`

sudo

`sudo su root`, `sudo su`, `sudo -s` vagy `sudo -u root -s`

Interaktív non-login shell `user2` felhasználóként

su

`su user2`

sudo

`sudo su user2` orvagy`sudo -u user2 -s`

3. Milyen startup fájl kerül beolvasásra, amikor a “Shell típusa” alatti shell elindul?

Shell típusa	`/etc/profile`	`/etc/bash.bashrc`	`~/.profile`	`~/.bashrc`
Interaktív-login shell `user2` felhasználóként	Igen	Igen	Igen	Igen
Interaktív login shell `root` felhasználóként	Igen	Igen	Nem	Nem
Interaktív non-login shell `root` felhasználóként	Nem	Igen	Nem	Nem
Interaktív non-login shell `user2` felhasználóként	Nem	Igen	Nem	Igen

Válaszok a gondolkodtató feladatokra

1. Bashben írhatunk egy egyszerű `Hello world!` függvényt, ha a következő kódot beillesztjük egy üres fájlba:

```
function hello() {
     echo "Hello world!"
}
```

   - Mi a következő teendőnk, hogy a funkciót elérhetővé tegyük a shell számára?

     Ahhoz, hogy a funkciót az aktuális shell számára elérhetővé tegyük, sourceolnunk kell az azt tartalmazó fájlt.

   - Ha már elérhető az aktuális shell számára, hogyan lehet meghívni?

     Úgy, hogy beírjuk a nevét a terminálba.

   - Az automatizálás érdekében melyik fájlba tehetnénk a függvényt és annak meghívását, hogy az akkor is végrehajtódjon, amikor a `user2` megnyit egy terminált egy X Window munkamenetből? Milyen típusú shellről van szó?

     A legjobb fájl a `/home/user2/.bashrc`. Az invokált shell egy interaktív non-login shell lesz.

   - Milyen fájlban helyezhetnénk el a függvényt és annak meghívását, hogy akkor fusson el, amikor a `root` elindít egy új interaktív shellt, függetlenül attól, hogy bejelentkezik-e vagy sem?

     Az `/etc/bash.bashrc` állományban, mivel ez a fájl minden interaktív shell esetében végrehajtásra kerül - akár történik bejelentkezés, akár nem.

2. Nézzük meg az alábbi, egyszerű `Hello world!` `bash` scriptet:

```
#!/bin/bash

#hello_world: egy egyszerű bash script a scriptek interakciójának megvitatására.

echo "Hello world!"
```

   - Tegyük fel, hogy a scriptet futtathatóvá tesszük és futtatjuk. Ez egy interaktív script lenne? Miért?

Nem, mivel nincs benne emberi interakció és felhasználó által beírt parancsok sem.

- Mi teszi a scriptet interaktívvá?

  A tény, hogy felhasználói beavatkozást igényel.

3. Képzeljük el, hogy megváltoztattuk néhány változó értékét a `~/.bashrc` állományban, és azt szeretnénk, hogy ezek a változások újraindítás nélkül lépjenek életbe! Hogyan lehetne ezt a home mappából kétféleképpen megvalósítani?

   ```
   $ source .bashrc
   ```

   vagy

   ```
   $ . .bashrc
   ```

4. John most indított el egy X Window munkamenetet egy Linux-szerveren. Megnyit egy terminál emulátort, hogy elvégezzen néhány adminisztrációs feladatot, de meglepő módon a munkamenet lefagy, és meg kell nyitnia egy szöveges shellt.

   - Hogyan nyithatja meg a `tty` shellt?

     Megteheti a `Ctrl` + `Alt` + `F1`-`F6` lenyomásával, amivel beléphet a hat tty shell egyikébe.

   - Melyik startup fájl lesz sourceolva?

     `/etc/profile`
     `/home/john/.profile`

5. Linda egy Linux-szerver felhasználója. Kéri a rendszergazdát, hogy legyen egy `~/.bash_login` fájlja, hogy bejelentkezéskor a képernyőre kiírhassa az időt és a dátumot. A többi felhasználónak tetszik az ötlet, és követik a példáját. A rendszergazdának nehézséget okoz a szerver összes többi felhasználója számára létrehozni a fájlt, ezért egy új policy hozzáadása mellett dönt, és minden potenciális új felhasználó számára létrehozza a `~/.bash_login` fájlt. Hogyan tudja a rendszergazda ezt a feladatot elvégezni?

   Ezt úgy érheti el, hogy a `.bash_login` állományt a `/etc/skel` mappába helyezi.

Linux Professional Institute

105.1 Lecke 2

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	105 Shellek és shell scriptelés
Fejezet:	105.1 A shell testre szabása és használata
Lecke:	2/3

Bevezetés

Gondoljunk a változóra úgy, mint egy képzeletbeli dobozra, amelyben ideiglenesen elhelyezhetünk egy információt. Az inicializáló scriptekhez hasonlóan a Bash is a változókat *shell/local* (azok, amelyek csak annak a shellnek a határain belül élnek, amelyben létrehozták őket) vagy *environment/global* (azok, amelyeket a gyermek shell-ek és/vagy folyamatok örökölnek) kategóriákba sorolja. Valójában az előző leckében a shelleket és azok konfigurációs vagy inicializálási scriptjeit néztük meg. Most célszerű rámutatni, hogy ezeknek az indítófájloknak az ereje abban rejlik, hogy lehetővé teszik számunkra, hogy olyan változókat — valamint aliasokat és függvényeket — használjunk, amelyek segítségével létrehozhatjuk és testre szabhatjuk a kívánt shell környezetet.

Változók: Hozzárendelés és hivatkozás

A változót úgy lehet definiálni, mint egy értéket tartalmazó nevet.

A Bash nyelvben a névhez való értékadást *változó hozzárendelésnek* (variable assignment) nevezik, és ez a módja annak, ahogyan változókat hozunk létre vagy állítunk be. Másrészt a névben szereplő érték elérésének folyamatát *változó hivatkozásnak* (variable referencing)

nevezzük.

A változók hozzárendelésének szintaxisa a következő:

```
<valtozo_neve>=<valtozo_erteke>
```

Például:

```
$ distro=zorinos
```

A `distro` változó egyenlő `zorinos`-sal, vagyis van egy memóriarész, amely a `zorinos` értéket tartalmazza — a `distro` pedig az erre mutató pointer.

Vegyük azonban figyelembe, hogy a változó hozzárendelésekor az egyenlőségjel egyik oldalán sem lehet szóköz:

```
$ distro =zorinos
-bash: distro: command not found
$ distro= zorinos
-bash: zorinos: command not found
```

A hibánk miatt a Bash a `distro`-t és a `zorinos`-t is parancsnak olvasta.

Egy változóra való hivatkozáshoz (azaz értékének ellenőrzéséhez) az `echo` parancsot használjuk, a változó neve előtt egy `$` jellel:

```
$ echo $distro
zorinos
```

Változók nevei

A változók nevének kiválasztásakor bizonyos szabályokat figyelembe kell vennünk.

A változó neve tartalmazhat betűket (`a-z`,`A-Z`), számokat (`0-9`) és alulvonásokat (`_`):

```
$ distro=zorinos
$ echo $distro
zorinos
$ DISTRO=zorinos
```

```
$ echo $DISTRO
zorinos
$ distro_1=zorinos
$ echo $distro_1
zorinos
$ _distro=zorinos
$ echo $_distro
zorinos
```

Nem kezdődhet számmal, különben Bash összezavarodik:

```
$ 1distro=zorinos
-bash: 1distro=zorinos: command not found
```

Nem tartalmazhat szóközöket (még idézőjelek használatával sem); a konvenció szerint helyette alulvonásokat használunk:

```
$ "my distro"=zorinos
-bash: my: command not found
$ my_distro=zorinos
$ echo $my_distro
zorinos
```

Változók értékei

A változók referenciájával vagy értékével kapcsolatban szintén fontos figyelembe venni néhány szabályt.

A változók tartalmazhatnak bármilyen alfanumerikus karaktert (`a-z`,`A-Z`,`0-9`), valamint a legtöbb egyéb karaktert (`?`,`!`,`*`,`.`,`/`, stb.):

```
$ distro=zorin12.4?
$ echo $distro
zorin12.4?
```

A változók értékeit idézőjelek közé kell zárni, ha szimpla szóközöket tartalmaznak:

```
$ distro=zorin 12.4
-bash: 12.4: command not found
$ distro="zorin 12.4"
```

```
$ echo $distro
zorin 12.4
$ distro='zorin 12.4'
$ echo $distro
zorin 12.4
```

A változó értékeit is idézőjelek közé kell zárni, ha olyan karaktereket tartalmaznak, mint az átirányításhoz használt karakterek (<,>) vagy a pipe (|). A következő parancs csak annyit tesz, hogy létrehoz egy üres `zorin` nevű fájlt:

```
$ distro=>zorin
$ echo $distro

$ ls zorin
zorin
```

Ez azonban akkor működik, ha az idézőjeleket használjuk:

```
$ distro=">zorin"
$ echo $distro
>zorin
```

Az egyszerű és a kettős idézőjelek azonban nem mindig felcserélhetők egymással. Attól függően, hogy mit csinálunk egy változóval (hozzárendelés vagy hivatkozás), az egyik vagy a másik használata következményekkel jár, és különböző eredményeket hoz. A változó hozzárendelésével összefüggésben az egyszerű idézőjelek a változó értékének minden karakterét *szó szerint* átveszik, míg a kettős idézőjelek lehetővé teszik a változó helyettesítését:

```
$ lizard=uromastyx
$ animal='My $lizard'
$ echo $animal
My $lizard
$ animal="My $lizard"
$ echo $animal
My uromastyx
```

Másrészt, amikor olyan változóra hivatkozunk, amelynek értéke tartalmaz néhány kezdeti (vagy extra) szóközt — néha csillagokkal kombinálva --, akkor kötelező dupla idézőjeleket használni a `echo` parancs után, hogy elkerüljük a *mezőfelosztást* (field splitting) és az *elérési út nevének kiterjesztését* (pathname expansion):

```
$ lizard="   genus   |   uromastyx"
$ echo $lizard
genus | uromastyx
$ echo "$lizard"
   genus   |   uromastyx
```

Ha a változó hivatkozása záró felkiáltójelet tartalmaz, akkor ennek kell lennie a string utolsó karakterének (különben a Bash azt fogja hinni, hogy egy `history` eseményre hivatkozunk):

```
$ distro=zorin.?/!os
-bash: !os: event not found
$ distro=zorin.?/!
$ echo $distro
zorin.?/!
```

A backslasheket egy másik backslashsel kell kivédeni (escape). Továbbá, ha a backslash az utolsó karakter a stringben, és nem védjük ki, a Bash úgy fogja értelmezni, hogy sortörést akarunk, és új sort ad nekünk:

```
$ distro=zorinos\
>
$ distro=zorinos\\
$ echo $distro
zorinos\
```

A következő két szakaszban összefoglaljuk a *lokális* (local) és a *környezeti* (environment) változók közötti főbb különbségeket.

Lokális vagy shell változók

A lokális vagy shell változók csak abban a shellben léteznek, amelyben létrehozták őket. A helyi változókat egyezményesen kisbetűvel írjuk.

Néhány teszt elvégzéséhez hozzunk létre egy lokális változót! A fentieknek megfelelően válasszunk egy megfelelő változónevet, és tegyük egyenlővé egy megfelelő értékkel! Például:

```
$ reptile=tortoise
```

Most használjuk a `echo` parancsot, hogy hivatkozzunk a változóra, és ellenőrizzük, hogy minden a

várt módon történt-e:

```
$ echo $reptile
tortoise
```

Bizonyos esetekben - például scriptek írása során - a változtathatatlanság érdekes tulajdonsága lehet a változóknak. Ha azt akarjuk, hogy a változóink megváltoztathatatlanok legyenek, akkor létrehozhatjuk őket `readonly` módon:

```
$ readonly reptile=tortoise
```

Vagy átalakíthatjuk őket a létrehozás után:

```
$ reptile=tortoise
$ readonly reptile
```

Ha most megpróbáljuk megváltoztatni a `reptile` értékét, a Bash elutasítja:

```
$ reptile=lizard
-bash: distro: readonly variable
```

NOTE Az aktuális munkamenetben lévő összes olvasható változó listázásához írjuk be a terminálba a `readonly` vagy `readonly -p` parancsot!

A helyi változók kezelésénél hasznos parancs a `set`.

A `set` kiírja az összes aktuálisan hozzárendelt shellváltozót és függvényt. Mivel ez sok sor lehet (próbáljuk ki!), ajánlott egy olyan lapozóval együtt használni, mint a `less`:

```
$ set | less
BASH=/bin/bash
BASHOPTS=checkwinsize:cmdhist:complete_fullquote:expand_aliases:extglob:extquote:force_figno
re:histappend:interactive_comments:login_shell:progcomp:promptvars:sourcepath
BASH_ALIASES=()
BASH_ARGC=()
BASH_ARGV=()
BASH_CMDS=()
BASH_COMPLETION_COMPAT_DIR=/etc/bash_completion.d
BASH_LINENO=()
```

```
BASH_SOURCE=()
BASH_VERSINFO=([0]="4" [1]="4" [2]="12" [3]="1" [4]="release" [5]="x86_64-pc-linux-gnu")
BASH_VERSION='4.4.12(1)-release'
(...)
```

Ott van a `reptile` változónk?

```
$ set | grep reptile
reptile=tortoise
```

Igen, itt van!

Azonban a `reptile` - mivel lokális változó - nem öröklődik az aktuális shellből indított gyermekfolyamatokra:

```
$ bash
$ set | grep reptile
$
```

És természetesen az értékét sem tudjuk kiíratni:

```
$ echo $reptile
$
```

NOTE A `bash` parancs beírásával a terminálba egy új (gyermek) shellt nyitunk.

Bármely változó (akár helyi, akár globális) visszavonásához az `unset` parancsot használjuk:

```
$ echo $reptile
tortoise
$ unset reptile
$ echo $reptile
$
```

NOTE Az `unset`-et csak a változó nevének kell követnie (nem előzheti meg a `$` szimbólum)!

Globális vagy környezeti változók

A globális (global) vagy környezeti (environment) változók az aktuális shellhez, valamint az abból indított összes későbbi folyamathoz tartoznak. A környezeti változókat konvenció szerint nagybetűvel írjuk:

```
$ echo $SHELL
/bin/bash
```

Ezeknek a változóknak az értékét rekurzívan átadhatjuk más változóknak, és az utóbbiak értéke végül az előbbiek értékére fog bővülni:

```
$ my_shell=$SHELL
$ echo $my_shell
/bin/bash
$ your_shell=$my_shell
$ echo $your_shell
/bin/bash
$ our_shell=$your_shell
$ echo $our_shell
/bin/bash
```

Ahhoz, hogy egy lokális shell változó környezeti változóvá váljon, az `export` parancsot kell használni:

```
$ export reptile
```

Az `export reptile` segítségével a helyi változót környezeti változóvá alakítottuk, hogy a gyermek shellek is felismerhessék és használhassák:

```
$ bash
$ echo $reptile
tortoise
```

Hasonlóképpen, az `export` is használható egy változó beállítására és exportálására, egyszerre:

```
$ export amphibian=frog
```

Most már megnyithatunk egy új Bash-példányt, és sikeresen hivatkozhatunk az új változóra:

```
$ bash
$ echo $amphibian
frog
```

NOTE Az `export -n <VALTOZO_NEVE>` paranccsal a változó visszaváltozik lokális változóvá.

Az `export` parancs önmagában (vagy a `-p` opcióval) beírva az összes létező környezeti változó listáját is megadja:

```
$ export
declare -x HOME="/home/user2"
declare -x LANG="en_GB.UTF-8"
declare -x LOGNAME="user2"
(...)
declare -x PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
declare -x PWD="/home/user2"
declare -x SHELL="/bin/bash"
declare -x SHLVL="1"
declare -x SSH_CLIENT="192.168.1.10 49330 22"
declare -x SSH_CONNECTION="192.168.1.10 49330 192.168.1.7 22"
declare -x SSH_TTY="/dev/pts/0"
declare -x TERM="xterm-256color"
declare -x USER="user2"
declare -x XDG_RUNTIME_DIR="/run/user/1001"
declare -x XDG_SESSION_ID="8"
declare -x reptile="tortoise"
```

NOTE A `declare -x` parancs egyenértékű az `export` paranccsal.

Két további parancs, amely az összes környezeti változó listájának kiírására használható, az `env` és a `printenv`:

```
$ env
SSH_CONNECTION=192.168.1.10 48678 192.168.1.7 22
LANG=en_GB.UTF-8
XDG_SESSION_ID=3
USER=user2
PWD=/home/user2
```

```
HOME=/home/user2
SSH_CLIENT=192.168.1.10 48678 22
SSH_TTY=/dev/pts/0
MAIL=/var/mail/user2
TERM=xterm-256color
SHELL=/bin/bash
SHLVL=1
LOGNAME=user2
XDG_RUNTIME_DIR=/run/user/1001
PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
_=/usr/bin/env
```

Amellett, hogy az `env` szinonimája, a `printenv`-t néha hasonló módon használhatjuk, mint ahogy az `echo` parancsot egy változó értékének ellenőrzésére:

```
$ echo $PWD
/home/user2
$ printenv PWD
/home/user2
```

Figyeljük meg azonban, hogy a `printenv` esetén a változó neve előtt nem szerepel a `$`!

NOTE A `PWD` a jelenlegi mappa elérési útvonalát tárolja. Erről és más gyakori környezeti változókról később fogunk tanulni.

Program futtatása módosított környezetben

Az `env` a shell környezet módosítására használható a program végrehajtásának idején.

Ahhoz, hogy egy új Bash munkamenetet a lehető legüresebb környezettel indítsunk — a legtöbb változót (valamint függvényeket és aliasokat) törölve — az `env`-t fogjuk használni az `-i` kapcsolóval:

```
$ env -i bash
```

Most a legtöbb környezeti változó eltűnt:

```
$ echo $USER
$
```

És csak néhány maradt meg:

```
$ env
LS_COLORS=
PWD=/home/user2
SHLVL=1
_=/usr/bin/printenv
```

Az `env`-t használhatjuk arra is, hogy egy adott változót beállítsunk egy adott programhoz.

Az előző leckében, amikor a *non-interaktív, non-login shelleket* tárgyaltuk, láttuk, hogy a scriptek nem olvasnak be semmilyen szabványos indítófájlt, hanem a `BASH_ENV` változó értékét keresik, és azt használják indítófájlként, ha létezik.

Hadd mutassuk be ezt a folyamatot:

1. Létrehozzuk a saját indítófájlunkat `.startup_script` néven a következő tartalommal:

   ```
   CROCODILIAN=caiman
   ```

2. Írunk egy `test_env.sh` nevű Bash scriptet a következő tartalommal:

   ```
   #!/bin/bash

   echo $CROCODILIAN
   ```

3. Beállítjuk az executable bitet a `test_env.sh` scriptünkhöz:

   ```
   $ chmod +x test_env.sh
   ```

4. Végül az `env` segítségével a `BASH_ENV`-t a `.startup_script`-re állítjuk a `test_env.sh` számára:

   ```
   $ env BASH_ENV=/home/user2/.startup_script ./test_env.sh
   caiman
   ```

   Az `env` parancs akkor is implicit, ha megszabadulunk tőle:

   ```
   $ BASH_ENV=/home/user2/.startup_script ./test_env.sh
   ```

```
caiman
```

NOTE

Ha nem értjük a `#!/bin/bash` sort vagy a `chmod +x` parancsot, ne essünk pánikba! A következő leckékben megtanulunk mindent, ami a shell scriptekről szükséges. Egyelőre csak emlékezzünk arra, hogy egy script saját mappájából történő futtatásához a `./some_script` parancsot használjuk!

Általános környezeti változók

Itt az ideje, hogy áttekintsük a legfontosabb környezeti változókat, amelyek a Bash konfigurációs fájlokban vannak beállítva!

`DISPLAY`

Az X-szerverhez kapcsolódóan ennek a változónak az értéke általában három elemből áll:

- Az a hostnév (ennek hiányában a `localhost`), ahol az X-szerver fut.
- Egy kettőspont, mint elválasztójel.
- Egy szám (ez általában `0` és a számítógép kijelzőjére utal).

```
$ printenv DISPLAY
:0
```

Ha ez a változó üres értéket kap, az X Window System nélküli szervert jelent. Egy extra szám — mint a `my.xserver:0:1` — a képernyők számára utal, ha több is van.

`HISTCONTROL`

Ez a változó szabályozza, hogy milyen parancsok kerüljenek elmentésre a `HISTFILE`-ba (lsd. alább). Három lehetséges érték van:

`ignorespace`

A szóközzel kezdődő parancsok nem kerülnek mentésre.

`ignoredups`

Az előzővel megegyező parancs nem kerül mentésre.

`ignoreboth`

Az előző két kategória bármelyikébe tartozó parancsok nem kerülnek mentésre.

```
$ echo $HISTCONTROL
```

```
ignoreboth
```

HISTSIZE

Ez állítja be a memóriában tárolt parancsok számát, amíg a shell munkamenet tart.

```
$ echo $HISTSIZE
1000
```

HISTFILESIZE

Ez állítja be a munkamenet elején és végén a `HISTFILE`-ba mentendő parancsok számát:

```
$ echo $HISTFILESIZE
2000
```

HISTFILE

Annak a fájlnak a neve, amely az összes parancsot a beírásukkor tárolja. Alapértelmezés szerint ez a fájl a `~/.bash_history` címen található:

```
$ echo $HISTFILE
/home/user2/.bash_history
```

NOTE

A `HISTFILE` tartalmának megtekintéséhez egyszerűen írjuk be a `history` parancsot! Alternatívaként megadhatjuk a látni kívánt parancsok számát, ha a `history`-nak átadunk egy argumentumot (a legutóbbi parancsok számát), mint pl. a `history 3`.

HOME

Ez a változó az aktuális felhasználó home mappájának abszolút elérési útvonalát tárolja, és a felhasználó bejelentkezésekor kerül beállításra.

Ez a kódrészlet — a `~/.profile`-ból — magától értetődő (a `"$HOME/.bashrc"-t forrásként használja, ha létezik):

```
# include .bashrc if it exists
if [ -f "$HOME/.bashrc" ]; then
. "$HOME/.bashrc"
fi
```

NOTE Ha nem igazán értjük az `if` utasítást, ne aggódjunk: csak nézzük meg a shell scriptekről szóló leckéket!

Ne feledjük, hogy a `~` a `$HOME` megfelelője:

```
$ echo ~; echo $HOME
/home/carol
/home/carol
```

NOTE A parancsokat pontosvesszővel lehet összekapcsolni (`;`).

Ezt egy `if` utasítással is bizonyíthatjuk:

```
$ if [ ~ == "$HOME" ]; then echo "true"; else echo "false"; fi
true
```

NOTE Ne feledjük: Az egyenlőségjelet `=` a változó hozzárendeléséhez használjuk. Az `==` az egyenlőség tesztelésére szolgál.

HOSTNAME

Ez a változó tárolja a host számítógép TCP/IP nevét:

```
$ echo $HOSTNAME
debian
```

HOSTTYPE

Ez tárolja a host számítógép processzorának architektúráját:

```
$ echo $HOSTTYPE
x86_64
```

LANG

Ez a változó tárolja a rendszer nyelvi beállításait:

```
$ echo $LANG
en_UK.UTF-8
```

`LD_LIBRARY_PATH`

Ez a változó azon mappák kettősponttal elválasztott halmazából áll, amelyekben a megosztott mappákat a programok megosztják:

```
$ echo $LD_LIBRARY_PATH
/usr/local/lib
```

`MAIL`

Ez a változó tárolja azt a fájlt, amelyben a Bash az e-maileket keresi:

```
$ echo $MAIL
/var/mail/carol
```

Egy másik gyakori értéke ennek a változónak a következő: `/var/spool/mail/$USER`.

`MAILCHECK`

Ez a változó egy numerikus értéket tárol, amely másodpercekben jelzi, hogy a Bash milyen gyakran ellenőrzi az új levelek érkezését:

```
$ echo $MAILCHECK
60
```

`PATH`

Ez a környezeti változó tárolja azoknak a mappáknak a listáját, ahol a Bash futtatható fájlokat keres, amikor bármilyen program futtatása a feladat. A mi példagépünkön ez a változó a rendszerszintű `/etc/profile` fájlon keresztül van beállítva:

```
if [ "`id -u`" -eq 0 ]; then
  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
else
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
fi
export PATH
```

Az `if` utasítással a felhasználó identitását teszteljük, és — a teszt eredményétől függően (`root` vagy más) — megkapjuk az egyik `PATH`-t vagy a másikat. Végül a kiválasztott `PATH`-t az `export` segítségével továbbítjuk.

Két dolgot figyeljünk meg a `PATH` értékével kapcsolatban:

- A mappanevek abszolút elérési utak használatával íródnak.
- A kettőspont elválasztójelként van használva.

Ha a normál felhasználók számára a `/usr/local/sbin` mappát a `PATH`-ban szeretnénk elhelyezni, akkor módosítjuk a sort, hogy így nézzen ki:

```
(...)
else
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/usr/local/sbin"
fi
export PATH
```

Most láthatjuk, hogyan változik a változó értéke, ha normál felhasználóként jelentkezünk be:

```
# su - carol
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/usr/local/sbin
```

NOTE

Hozzáadhattuk volna a `/usr/local/sbin` stringet is a felhasználó `PATH` -jához a parancssorban a `PATH=/usr/local/sbin:$PATH` vagy a `PATH=$PATH:/usr/local/sbin` beírásával — az előbbi, ami a `/usr/local/sbin`-t teszi az első mappának, amelyben futtatható fájlokat kell keresni; az utóbbi az utolsót.

PS1

Ez a változó tárolja a Bash prompt értékét. A következő kódrészletben (szintén az `/etc/profile`-ból) az `if` utasítás teszteli a felhasználó személyazonosságát, és ennek megfelelően egy nagyon diszkrét promptot határoz meg (`#` `root`, vagy `$` normál felhasználók esetén):

```
if [ "`id -u`" -eq 0 ]; then
  PS1='# '
else
  PS1='$ '
fi
```

> **NOTE** A `root` `id`-ja `0`. Váltsunk a `root` felhasználóra és teszteljük az `id -u` segítségével!

További prompt változók például:

`PS2`
: Általában `>`-re van állítva, és hosszú, többsoros parancsok folytatásaként használják.

`PS3`
: A `select` parancs promptjaként használják.

`PS4`
: Általában `+`-ra van állítva, és hibakeresésre használják.

`SHELL`

Ez a változó az aktuális shell abszolút útvonalát tárolja:

```
$ echo $SHELL
/bin/bash
```

`USER`

Ez tárolja az aktuális felhasználó nevét:

```
$ echo $USER
carol
```

Gyakorló feladatok

1. Figyeljük meg a változóhozzárendelést a “Parancs(ok)” oszlopban és jelezzük, hogy a kapott változó “Local” (lokális) vagy “Global” (globális):

Parancs(ok)	Local	Global
`debian=mother`		
`ubuntu=deb-based`		
`mint=ubuntu-based; export mint`		
`export suse=rpm-based`		
`zorin=ubuntu-based`		

2. Figyeljük meg a “Parancs” és a “Kimenet” oszlopokat és magyarázzuk el a jelentésüket:

Parancs	Kimenet	Jelentés
`echo $HISTCONTROL`	`ignoreboth`	
`echo ~`	`/home/carol`	
`echo $DISPLAY`	`reptilium:0:2`	
`echo $MAILCHECK`	`60`	
`echo $HISTFILE`	`/home/carol/.bash_history`	

3. A “Hibás parancs” oszlopban helytelenül vannak beállítva a változók. Adjuk meg a hiányzó információkat a “Helyes parancs” és “Változóhivatkozás” oszlopokban, hogy az eredmény a “Várható kimenet” értéke legyen:

Hibás parancs	Helyes parancs	Változóhivatkozás	Várható kimenet
`lizard =chameleon`			`chameleon`
`cool lizard=chameleon`			`chameleon`
`lizard=cha|me|leon`			`cha|me|leon`
`lizard=/** chameleon **/`			`/** chamelon **/`

Hibás parancs	Helyes parancs	Változóhivatkozás	Várható kimenet
`win_path=C:\path\to\dir\`			`C:\path\to\dir\`

4. Nézzük meg a célt és írjuk be a megfelelő parancsot:

Cél	Parancs
Az aktuális shell nyelvének beállítása spanyol UTF-8-ra (`es_ES.UTF-8`).	
Az aktuális mappa nevének kiírása.	
Hivatkozás az `ssh` kapcsolatokról szóló információkat tároló környezeti változóra.	
A `PATH` beállítása, hogy a `/home/carol/scripts` legyen az utolsó mappa, amelyben futtatható fájlokat keres.	
A `my_path` értékének beállítása a `PATH`-ra.	
A `my_path` értékének beállítása az adott `PATH` értékére.	

5. Hozzunk létre egy lokális változót `mammal` néven és legyen az értéke `gnu`:

6. A változóhelyettesítés segítségével hozzunk létre egy másik helyi változót `var_sub` néven a megfelelő értékkel, hogy amikor a `echo $var_sub` segítségével hivatkozunk rá, a következőt kapjuk: `The value of mammal is gnu`:

7. Alakítsuk a `mammal`-t környezeti változóvá:

8. Keressük meg a `set` és a `grep` segítségével:

9. Keressük meg az `env` és a `grep` segítségével:

10. Hozzunk létre két egymást követő parancsban egy `BIRD` nevű környezeti változót, amelynek értéke `penguin`:

11. Hozzunk létre egyetlen paranccsal egy `NEW_BIRD` nevű környezeti változót, amelynek értéke `yellow-eyed pengiun`:

12. Tételezzük fel, hogy mi vagyunk a `user2` felhasználó, hozzunk létre egy `bin` mappát a home mappánk alatt:

13. Írjuk be a parancsot ami `~/bin` mappát hozzáadja a `PATH-hoz` úgy, hogy ez legyen az első mappa, amelyben a `bash` binárisokat keres:

14. Annak garantálására, hogy a `PATH` értéke változatlan maradjon az újraindítások során, milyen kódrészletet - egy `if` utasítás formájában - tehetnénk a `~/.profile`-ba?

Gondolkodtató feladatok

1. `let`: több mint aritmetikai kifejezés kiértékelése:
   - Keressünk rá a manpage-ben vagy az interneten a `let`-re és annak hatására a változók beállításakor, és hozzunk létre egy új helyi változót `my_val` néven, amelynek értéke `10` — az 5 és 5 hozzáadásának eredményeként:
   - Most hozzunk létre egy másik változót `your_val` néven, aminek az értéke `5` — a `my_val` értékének 2-vel történő osztásának eredményeként:

2. Egy parancs eredménye egy változóban? Természetesen lehetséges; ezt hívják *parancshelyettesítésnek* (command substitution). Vizsgáljuk meg, és tanulmányozzuk a következő `music_info` nevű függvényt:

```
music_info(){
latest_music=`ls -l1t ~/Music | head -n 6`
echo -e "Your latest 5 music files:\n$latest_music"
}
```

   Az `ls -l1t ~/Music | head -n 6` parancs eredménye lesz a `latest_music` változó értéke. Ezután a `latest_music` változóra hivatkozik a `echo` parancs (amely kiadja a `Music` mappa által elfoglalt összes bájt számát és a `Music` mappában tárolt legutóbbi öt zenefájlt — soronként egyet).

   Az alábbiak közül melyik érvényes szinonimája a következőknek

```
latest_music=`ls -l1t ~/Music | head -n 6`
```

   A lehetőség:

```
latest_music=$(ls -l1t ~/Music| head -n 6)
```

   B lehetőség:

```
latest_music="(ls -l1t ~/Music| head -n 6)"
```


C lehetőség:

```
latest_music=((ls -l1t ~/Music| head -n 6))
```

Összefoglalás

Ebben a leckében megtanultuk:

- A változók nagyon fontos részét képezik a shell környezetének, mivel maga a shell és más programok is használják őket.
- Hogyan rendelhetünk hozzá és hivatkozhatunk változókra.
- A *local* (lokális) és *global* (globális) (vagy *environment* (környezeti)) változók közötti különbségek.
- Hogyan tegyük a változókat *readonly*-vá.
- Hogyan lehet egy helyi változót környezeti változóvá alakítani az `export` paranccsal.
- Hogyan lehet felsorolni az összes környezeti változót.
- Hogyan futtassunk egy programot módosított környezetben.
- Hogyan lehet a változókat állandóvá tenni a startup scriptek segítségével.
- Néhány gyakori környezeti változó: `DISPLAY`, `HISTCONTROL`, `HISTSIZE`, `HISTFILESIZE`, `HISTFILE`, `HOME`, `HOSTNAME`, `HOSTTYPE`, `LANG`, `LD_LIBRARY_PATH`, `MAIL`, `MAILCHECK`, `PATH`, `PS1` (és más prompt változók), `SHELL` és `USER`.
- A tilde (`~`) jelentése.
- Az `if` utasítások alapjai.

A leckében használt parancsok:

`echo`

Hivatkozás egy változóra.

`ls`

Mappa tartalmának listázása.

`readonly`

Változók megváltoztathatatlanná tétele. Az aktuális munkamenet összes olvasható változójának listázása.

`set`

Az aktuális munkamenet összes változójának és függvényének listázása.

`grep`

Mintának megfelelő sorok nyomtatása.

`bash`

Új shell indítása

`unset`

Változók visszavonása.

`export`

Egy helyi változót környezeti változóvá alakít. Környezeti változók listázása.

`env`

Környezeti változók listázása. Program futtatása módosított környezetben.

`printenv`

Környezeti változók listázása. Hivatkozás egy változóra.

`chmod`

Egy fájl üzemmódbitjeinek módosítása, például futtathatóvá tétele.

`history`

Előzőleg használt parancsok listázása.

`su`

Felhasználói azonosító módosítása vagy szuperfelhasználóvá válás.

`id`

Felhasználói ID kiíratása.

Válaszok a gyakorló feladatokra

1. Figyeljük meg a változóhozzárendelést a “Parancs(ok)” oszlopban és jelezzük, hogy a kapott változó “Local” (lokális) vagy “Global” (globális):

Parancs(ok)	Local	Global
`debian=mother`	Igen	Nem
`ubuntu=deb-based`	Igen	Nem
`mint=ubuntu-based; export mint`	Nem	Igen
`export suse=rpm-based`	Nem	Igen
`zorin=ubuntu-based`	Igen	Nem

2. Figyeljük meg a “Parancs” és a “Kimenet” oszlopokat és magyarázzuk el a jelentésüket:

Parancs	Kimenet	Jelentés
`echo $HISTCONTROL`	`ignoreboth`	A duplikált és a szóközzel kezdődő parancsok nem kerülnek elmentésre a `history`-ba.
`echo ~`	`/home/carol`	A `HOME` mappája `carol`-nak `/home/carol`.
`echo $DISPLAY`	`reptilium:0:2`	A `reptilium` gépen fut egy X szerver, és a kijelző második képernyőjét használjuk.
`echo $MAILCHECK`	`60`	A levelezés minden percben ellenőrzésre kerül.
`echo $HISTFILE`	`/home/carol/.bash_history`	A `history` a `/home/carol/.bash_history`-ban kerül mentésre.

3. A “Hibás parancs” oszlopban helytelenül vannak beállítva a változók. Adjuk meg a hiányzó információkat a “Helyes parancs” és “Változóhivatkozás” oszlopokban, hogy az eredmény a “Várható kimenet” értéke legyen:

Hibás parancs	Helyes parancs	Változóhivatkozás	Várható kimenet
`lizard =chameleon`	`lizard=chameleon`	`echo $lizard`	`chameleon`
`cool lizard=chameleon`	`cool_lizard=chameleon` (for example)	`echo $cool_lizard`	`chameleon`
`lizard=cha|me|leon`	`lizard="cha|me|leon"` or `lizard='cha|me|leon'`	`echo $lizard`	`cha|me|leon`
`lizard=/** chameleon **/`	`lizard="/** chameleon **/"` or `lizard='/** chameleon **/'`	`echo "$lizard"`	`/** chamelon **/`
`win_path=C:\path\to\dir\`	`win_path=C:\\path\\to\\dir\\`	`echo $win_path`	`C:\path\to\dir\`

4. Nézzük meg a célt és írjuk be a megfelelő parancsot:

Cél	Parancs
Az aktuális shell nyelvének beállítása spanyol UTF-8-ra (`es_ES.UTF-8`).	`LANG=es_ES.UTF-8`
Az aktuális mappa nevének kiírása	`echo $PWD` or `pwd`
Hivatkozás az `ssh` kapcsolatokról szóló információkat tároló környezeti változóra.	`echo $SSH_CONNECTION`
A `PATH` beállítása, hogy a `/home/carol/scripts` legyen az utolsó mappa, amelyben futtatható fájlokat keres.	`PATH=$PATH:/home/carol/scripts`
A `my_path` értékének beállítása a `PATH`-ra.	`my_path=PATH`
A `my_path` értékének beállítása az adott `PATH` értékére.	`my_path=$PATH`

5. Hozzunk létre egy lokális változót `mammal` néven és legyen az értéke `gnu`:

```
mammal=gnu
```

6. A változóhelyettesítés segítségével hozzunk létre egy másik helyi változót `var_sub` néven a

megfelelő értékkel, hogy amikor a `echo $var_sub` segítségével hivatkozunk rá, a következőt kapjuk: `The value of mammal is gnu`:

```
var_sub="The value of mammal is $mammal"
```

7. Alakítsuk a `mammal`-t környezeti változóvá:

```
export mammal
```

8. Keressük meg a `set` és a `grep` segítségével:

```
set | grep mammal
```

9. Keressük meg a `set` és a `grep` segítségével:

```
env | grep mammal
```

10. Hozzunk létre két egymást követő parancsban egy `BIRD` nevű környezeti változót, amelynek értéke `penguin`:

```
BIRD=penguin; export BIRD
```

11. Hozzunk létre egyetlen paranccsal egy `NEW_BIRD` nevű környezeti változót, amelynek értéke `yellow-eyed pengiun`:

```
export NEW_BIRD="yellow-eyed penguin"
```

or

```
export NEW_BIRD='yellow-eyed penguin'
```

12. Tételezzük fel, hogy mi vagyunk az `user2` felhasználó, hozzunk létre egy `bin` mappát a home mappánk alatt:

```
mkdir ~/bin
```

vagy

```
mkdir /home/user2/bin
```

vagy

```
mkdir $HOME/bin
```

13. Írjuk be a parancsot ami `~/bin` mappát hozzáadja a `PATH-hoz` úgy, hogy ez legyen az első mappa, amelyben a `bash` binárisokat keres:

```
PATH="$HOME/bin:$PATH"
```

A `PATH=~/bin:$PATH` vagy a `PATH=/home/user2/bin:$PATH` egyformán helyesek.

14. Annak garantálására, hogy a `PATH` értéke változatlan maradjon az újraindítások során, milyen kódrészletet - egy `if` utasítás formájában - tehetnénk a `~/.profile`-ba?

```
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi
```

Válaszok a gondolkodtató feladatokra

1. `let`: több mint aritmetikai kifejezés kiértékelése:
   - Keressünk rá a manpage-ben vagy az interneten a `let`-re és annak hatására a változók beállításakor, és hozzunk létre egy új helyi változót `my_val` néven, amelynek értéke `10` — az 5 és 5 hozzáadásának eredményeként:

     ```
     let "my_val = 5 + 5"
     ```

     vagy

     ```
     let 'my_val = 5 + 5'
     ```

   - Most hozzunk létre egy másik változót `your_val` néven, aminek az értéke `5` — a `my_val` értékének 2-vel történő osztásának eredményeként:

     ```
     let "your_val = $my_val / 2"
     ```

     vagy

     ```
     let 'your_val = $my_val / 2'
     ```

2. Egy parancs eredménye egy változóban? Természetesen lehetséges; ezt hívják *parancshelyettesítésnek* (command substitution). Vizsgáljuk meg, és tanulmányozzuk a következő `music_info` nevű függvényt:

   ```
   music_info(){
   latest_music=`ls -l1t ~/Music | head -n 6`
   echo -e "Your latest 5 music files:\n$latest_music"
   }
   ```

   Az `ls -l1t ~/Music | head -n 6` parancs eredménye lesz a `latest_music` változó értéke. Ezután a `latest_music` változóra hivatkozik a `echo` parancs (amely kiadja a `Music` mappa által elfoglalt összes bájt számát és a `Music` mappában tárolt legutóbbi öt zenefájlt — soronként egyet).

   Az alábbiak közül melyik érvényes szinonimája a következőknek

```
latest_music=`ls -l1t ~/Music | head -n 6`
```

Az A lehetőség:

```
latest_music=$(ls -l1t ~/Music| head -n 6)
```

105.1 Lecke 3

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	105 Shellek és shell scriptelés
Fejezet:	105.1 A shell testre szabása és használata
Lecke:	3/3

Bevezetés

Miután az előző leckékben áttekintettük a shelleket, a startup scripteket és a változókat, a shell testre szabásának témáját két nagyon érdekes elemmel zárjuk le: *alias* és *functions* (függvények). Valójában a változók, az aliasok és a függvények teljes csoportja — és ezek egymásra gyakorolt hatása — alkotja a shell környezetét.

E két rugalmas és időtakarékos eszköz fő erőssége az egységbezárás koncepciójához kapcsolódik: lehetőséget nyújtanak arra, hogy egyetlen parancs segítségével ismétlődő vagy visszatérő parancsok sorozatát állítsuk össze.

Aliasok létrehozása

Az alias más parancs(ok) helyettesítő neve. Futtatható, mint egy hagyományos parancs, de helyette egy másik parancsot hajt végre az alias definíciójának megfelelően.

Az aliasok deklarálásának szintaxisa meglehetősen egyszerű. Az aliasokat az `alias` kulcsszó, majd az alias hozzárendelés megírásával kell deklarálni. Az alias hozzárendelés viszont az *alias névből*, egy *egyenlőségjelből* és egy vagy több *parancsból* áll:

```
alias alias_name=command(s)
```

Például:

```
$ alias oldshell=sh
```

Ez a furcsa alias elindítja az eredeti `sh` shell egy példányát, amikor a felhasználó beírja az `oldshell`-t a terminálba:

```
$ oldshell
$
```

Az aliasok ereje abban rejlik, hogy lehetővé teszik a hosszú parancsok rövid változatainak megírását

```
$ alias ls='ls --color=auto'
```

NOTE Az `ls`-ről és a színeiről információt úgy kaphatunk, hogy beírjuk a `man dir_colors` parancsot a terminálba.

Hasonlóképpen, aliasokat hozhatunk létre egymáshoz kapcsolt parancsok sorozatához — a pontosvesszőt (`;`) használjuk elválasztóként. Lehet például egy olyan aliasunk, amely információt ad a `git` futtatható állomány helyéről és verziójáról:

```
$ alias git_info='which git;git --version'
```

Egy alias meghívásához írjuk be a nevét a terminálba:

```
$ git_info
/usr/bin/git
git version 2.7.4
```

Az `alias` parancs listát készít a rendszerben elérhető összes aliasról:

```
$ alias
alias git-info='which git;git --version'
alias ls='ls --color=auto'
```

```
alias oldshell='sh'
```

Az `unalias` parancs eltávolítja az aliasokat. Például `unalias git-info`, és láthatjuk, hogy eltűnik a listából:

```
$ unalias git-info
$ alias
alias ls='ls --color=auto'
alias oldshell='sh'
```

Amint azt az előző leckében az `alias hi='echo We salute you.'`-nál láttuk, a parancsokat idézőjelekbe kell zárnunk (akár szimpla, akár dupla), ha — az argumentumok vagy paraméterek miatt — szóközöket tartalmaznak:

```
$ alias greet='echo Hello world!'
$ greet
Hello world!
```

A szóközöket tartalmazó parancsok közé tartoznak az opciókat tartalmazó parancsok is:

```
$ alias ll='ls -al'
```

Most az `ll` az összes fájlt — beleértve a rejtetteket (`a`) — hosszú formátumban (`l`) fogja felsorolni.

Aliasokban hivatkozhatunk változókra:

```
$ reptile=uromastyx
$ alias greet='echo Hello $reptile!'
$ greet
Hello uromastyx!
```

A változó az aliason belül is hozzárendelhető:

```
$ alias greet='reptile=tortoise; echo Hello $reptile!'
$ greet
Hello tortoise!
```

Az aliasokat a `\` segítségével védhetjük ki (escape):

```
$ alias where?='echo $PWD'
$ where?
/home/user2
$ \where?
-bash: where?: command not found
```

Az aliasok kivédése akkor hasznos, ha az aliasnak ugyanaz a neve, mint egy hagyományos parancsnak. Ebben az esetben az alias elsőbbséget élvez az eredeti paranccsal szemben, amely azonban továbbra is elérhető, ha az aliast kivédjük.

Hasonlóképpen, egy alias-t egy másik alias-on belül is elhelyezhetünk:

```
$ where?
/home/user2
$ alias my_home=where?
$ my_home
/home/user2
```

Ráadásul egy függvényt egy aliason belül is elhelyezhetünk, ahogy az később látható lesz.

Idézőjelek bővítése és kiértékelése az aliasokban

Amikor idézőjeleket használunk a környezeti változókkal, a szimpla idézőjelek dinamikussá teszik a bővítést:

```
$ alias where?='echo $PWD'
$ where?
/home/user2
$ cd Music
$ where?
/home/user2/Music
```

Dupla idézőjelek esetén azonban a bővítés statikusan történik:

```
$ alias where?="echo $PWD"
$ where?
/home/user2
$ cd Music
$ where?
/home/user2
```

Az aliasok perzisztanciája: startup scriptek

A változókhoz hasonlóan, ahhoz, hogy az aliasaink tartósan fennmaradjanak, inicializáló scriptekbe kell helyeznünk őket, amelyeket az indításkor lekérdezünk. Mint már tudjuk, a felhasználók számára a `~/.bashrc` egy jó fájl a személyes aliasaik elhelyezésére. Valószínűleg néhány alias már megtalálható ott (a legtöbbjük ki van kommentelve és készen áll a használatra a vezető `#` eltávolításával):

```
$ grep alias .bashrc
# enable color support of ls and also add handy aliases
    alias ls='ls --color=auto'
    #alias dir='dir --color=
    #alias vdir='vdir --color=
    #alias grep='grep --color=
    #alias fgrep='fgrep --color'
    #alias egrep='egrep --color=
# some more ls aliases
#ll='ls -al'
#alias la='ls -A'
#alias l='ls -CF'
# ~/.bash_aliases, instead of adding them here directly.
if [ -f ~/.bash_aliases ]; then
   . ~/.bash_aliases
```

Ahogy az utolsó három sorban olvasható, lehetőségünk van arra, hogy saját alias-dedikált fájlunk legyen — `~/.bash_aliases` — és azt minden rendszerindításkor a `.bashrc` sourceolja. Tehát választhatjuk ezt a lehetőséget, és létrehozhatunk és feltölthetünk tartalommal egy ilyen fájlt:

```
############
# .bash_aliases:
# a file to be populated by the user's personal aliases (and sourced by ~/.bashrc).
############
alias git_info='which git;git --version'
alias greet='echo Hello world!'
alias ll='ls -al'
alias where?='echo $PWD'
```

Függvények létrehozása

Az aliasokhoz képest a függvények sokkal programozhatóbbak és rugalmasabbak, különösen, ha a *Bash* speciális beépített változóiban és a pozicionális paraméterekben rejlő lehetőségek teljes

kihasználásáról van szó. Nagyszerűek az olyan folyamatvezérlő struktúrákkal való munkához is, mint a ciklusok vagy feltételes függvények. A függvényt olyan parancsnak tekinthetjük, amely logikát tartalmaz más parancsok blokkjain vagy gyűjteményein keresztül.

Két szintaxis függvények létrehozásához

A függvények definiálására két érvényes szintaxis létezik.

A `function` kulcsszó használata

Egyrészt használhatjuk a `function` kulcsszót, amelyet a függvény neve és a kapcsos zárójelek között lévő parancsok követnek:

```
function function_name {
command #1
command #2
command #3
.
.
.
command #n
}
```

A `()` használata

Másrészt elhagyhatjuk a `function` kulcsszót, és helyette két zárójelet használhatunk közvetlenül a függvény neve után:

```
function_name() {
command #1
command #2
command #3
.
.
.
command #n
}
```

Gyakori, hogy a függvényeket fájlokban vagy scriptekben helyezzük el. Azonban közvetlenül a parancsértelmezőbe is be lehet írni őket, minden egyes parancsot külön sorba írva — figyeljük meg a `PS2(>)` jelzést, amely a sortörés utáni új sort jelzi:

```
$ greet() {
> greeting="Hello world!"
> echo $greeting
> }
```

Bármelyik esetről is legyen szó — és függetlenül attól, hogy milyen szintaxist választunk — , ha úgy döntünk, hogy kihagyjuk a sortörést és egy függvényt csak egy sorban írunk, a parancsokat pontosvesszővel kell elválasztani egymástól (figyeljük meg a pontosvesszőt az utolsó parancs után is):

```
$ greet() { greeting="Hello world!"; echo $greeting; }
```

A `bash` nem panaszkodott, amikor megnyomtuk az Entert, így a függvényünk készen áll a meghívásra. Egy függvény meghívásához be kell írnunk a nevét a terminálba:

```
$ greet
Hello world!
```

A változókhoz és aliasokhoz hasonlóan, ha azt akarjuk, hogy a függvények a rendszer újraindítása után is megmaradjanak, akkor a shell inicializáló scriptekbe kell helyeznünk őket, mint például a `/etc/bash.bashrc` (globális) vagy a `~/.bashrc` (lokális).

WARNING

Miután aliasokat vagy függvényeket adtunk hozzá bármelyik startup script fájlhoz, az ilyen fájlokat a `.` vagy a `source` segítségével sourceolni kell, hogy a módosítások érvénybe lépjenek, ha nem akarunk kijelentkezni és újra belépni, vagy újraindítani a rendszert.

Beépített speciális Bash változók

A *Bourne Again Shell* egy sor speciális változóval rendelkezik, amelyek különösen hasznosak a függvények és scriptek számára. Ezek azért különlegesek, mert csak hivatkozni lehet rájuk — hozzárendelni nem. Itt van egy lista a legfontosabbakról:

`$?`

Ennek a változónak a hivatkozása a legutóbbi parancsfuttatás eredményére bővül. A `0` érték sikert jelent:

```
$ ps aux | grep bash
user2      420  0.0  0.4  21156  5012 pts/0    Ss   17:10   0:00 -bash
```

```
user2      640  0.0  0.0  12784   936 pts/0    S+   18:04   0:00 grep bash
$ echo $?
0
```

Bármi más érték, ami nem `0`, hibát jelent:

```
user1@debian:~$ ps aux |rep bash
-bash: rep: command not found
user1@debian:~$ echo $?
127
```

`$$`

Kiterjed a shell PID-jére (process ID):

```
$ ps aux | grep bash
user2      420  0.0  0.4  21156  5012 pts/0    Ss   17:10   0:00 -bash
user2      640  0.0  0.0  12784   936 pts/0    S+   18:04   0:00 grep bash
$ echo $$
420
```

`$!`

A legutóbbi háttérmunkához tartozó PID-re bővül:

```
$ ps aux | grep bash &
[1] 663
$ user2      420  0.0  0.4  21156  5012 pts/0    Ss+  17:10   0:00 -bash
user2      663  0.0  0.0  12784   972 pts/0    S    18:08   0:00 grep bash
^C
[1]+  Done                    ps aux | grep bash
$ echo $!
663
```

NOTE Ne feledjük, hogy az és jelet (`&`) a háttérben futó folyamatok indítására használják!

A `$0`-tól `$9`-ig terjedő pozicionális paraméterek

A függvénynek (aliasnak vagy scriptnek) átadott paraméterekre vagy argumentumokra bővülnek — a `$0` a script vagy a shell nevére bővül.

Hozzunk létre egy függvényt a pozicionális paraméterek bemutatására — figyeljük meg az `PS2`

(>) jelzést, amely a sortörés utáni új sorokat jelzi:

```
$ special_vars() {
> echo $0
> echo $1
> echo $2
> echo $3
}
```

Most meghívjuk a függvényt (`special_vars`) úgy, hogy átadunk három paramétert (`debian`, `ubuntu`, `zorin`):

```
$ special_vars debian ubuntu zorin
-bash
debian
ubuntu
zorin
```

Az elvárásnak megfelelően működik.

WARNING

Bár a pozicionális paraméterek átadása az aliasoknak technikailag lehetséges, ez egyáltalán nem funkcionális, mivel — az aliasoknál — a pozicionális paraméterek mindig a végén kerülnek átadásra:

```
$ alias great_editor='echo $1 is a great text editor'
$ great_editor emacs
is a great text editor emacs
```

További beépített, speciális Bash-változók:

`$#`

A parancsnak átadott argumentumok számával bővül.

`$@`, `$*`

A parancsnak átadott argumentumokkal bővülnek.

`$_`

Az utolsó paraméterre vagy a script nevére bővül (többek között; lsd. `man bash`, hogy többet megtudjunk!):

Változók a függvényekben

Természetesen függvényekben is használhatunk változókat.

Ennek bizonyítására ezúttal létrehozunk egy új üres fájlt `funed` néven, és a következő függvényt helyezzük bele:

```
editors() {

editor=emacs

echo "My editor is: $editor. $editor is a fun text editor."
}
```

Amint azt már sejthetjük, először sourceolni kell a fájlt ahhoz, hogy meg tudjuk hívni a függvényt:

```
$ . funed
```

Most már tesztelhetjük:

```
$ editors
My editor is emacs. emacs is a fun text editor.
```

Mint láthatjuk, ahhoz, hogy az `editors` funkció megfelelően működjön, először az `editor` változót kell beállítani. Ennek a változónak a hatókörét az aktuális shell lokálisan határozza meg, és addig hivatkozhatunk rá, amíg a munkamenet tart:

```
$ echo $editor
emacs
```

A helyi változókkal együtt környezeti változókat is bevonhatunk a függvényünkbe:

```
editors() {

editor=emacs

echo "The text editor of $USER is: $editor."
}
```

```
editors
```

Figyeljük meg, hogy ezúttal úgy döntöttünk, hogy a függvényt magából a fájlból hívjuk meg (`editors` az utolsó sorban). Így, amikor a fájlt forrásként használjuk, a függvény is meghívásra kerül — egyszerre:

```
$ . funed
The text editor of user2 is: emacs.
```

Pozicionális paraméterek a függvényekben

Valami hasonló történik a pozicionális paraméterekkel.

Átadhatjuk őket függvényeknek a fájlból vagy scriptből (figyeljük meg az utolsó sort: `editors tortoise`):

```
editors() {

editor=emacs

echo "The text editor of $USER is: $editor."
echo "Bash is not a $1 shell."
}

editors tortoise
```

Sourceoljuk a fájlt, hogy bizonyítsuk, működik:

```
$ . funed
The text editor of user2 is: emacs.
Bash is not a tortoise shell.
```

A parancssorban pedig pozicionális paramétereket is átadhatunk függvényeknek. Ennek bizonyítására megszabadulunk a fájl utolsó sorától:

```
editors() {

editor=emacs

echo "The text editor of $USER is: $editor."
```

```
echo "Bash is not a $1 shell."
}
```

Majd sourceoljuk a fájlt:

```
$ . funed
```

Végül meghívjuk a függvényt a parancssorban a `$1` pozicionális paraméterként megadott `tortoise` értékkel:

```
$ editors tortoise
The text editor of user2 is: emacs.
Bash is not a tortoise shell.
```

Függvények a scriptekben

Függvényeket leginkább Bash scriptekben találhatunk.

A `funed` fájlunk scriptté alakítása (a neve `funed.sh` lesz) tényleg gyerekjáték:

```
#!/bin/bash

editors() {

editor=emacs

echo "The text editor of $USER is: $editor."
echo "Bash is not a $1 shell."
}

editors tortoise
```

Ennyi! Csak két sort adtunk hozzá:

- Az első sor a *shebang* meghatározza, hogy melyik program fogja értelmezni a scriptet: `#!/bin/bash`. Érdekes módon ez a program maga a `bash`.
- Az utolsó sor egyszerűen csak a függvény meghívása.

Most már csak egy dolog maradt hátra — futtathatóvá kell tennünk a scriptet:

```
$ chmod +x funed.sh
```

Már készen is áll a futtatásra:

```
$ ./funed.sh
The text editor of user2 is: emacs.
Bash is not a tortoise shell.
```

NOTE A következő leckékben mindent megtudhatunk a *shell scriptelésről*.

Függvény aliason belül

Ahogy már fentebb említettük, függvényeket aliasokon belül is elhelyezhetünk:

```
$ alias great_editor='gr8_ed() { echo $1 is a great text editor; unset -f gr8_ed; }; gr8_ed'
```

Ez a hosszú érték magyarázatot érdemel, daraboljuk tehát fel:

- Először maga a függvény: `gr8_ed() { echo $1 is a great text editor; unset -f gr8_ed; }`
- A függvény utolsó parancsa — `unset -f gr8_ed` — visszavonja a függvényt, hogy az ne maradjon a jelenlegi `bash` munkamenetben az alias meghívása után.
- Végül, de nem utolsósorban az alias sikeres meghívásához először magát a függvényt is meg kell hívni: `gr8_ed`.

Hívjuk meg az aliast, hogy kiderüljön, működik-e:

```
$ great_editor emacs
emacs is a great text editor
```

Ahogy a fenti `unset -f gr8_ed` parancsban is látható, az `unset` parancsot nem csak változók, hanem függvények visszavonására is használhatjuk. Valójában vannak speciális kapcsolók vagy opciók:

`unset -v`
: változókhoz

`unset -f`

függvényekhez

Ha kapcsolók nélkül használjuk, az `unset` először egy változót próbál visszaállítani, és — ha ez nem sikerül — akkor egy függvényt.

Függvény függvényen belül

Most tegyük fel, hogy két dolgot szeretnénk közölni a `user2`-vel minden alkalommal, amikor bejelentkezik a rendszerbe:

- Köszönni és ajánlani/dicsérni egy szövegszerkesztőt.
- Mivel kezd sok `Matroska` videófájlt elhelyezni a `$HOME/Video` mappájában, ezért figyelmeztetni is szeretnénk.

Ennek érdekében a következő két függvényt helyeztük el a `/home/user2/.bashrc` állományban:

Az első függvény (`check_vids`) az `.mkv` fájlokat ellenőrzi és figyelmeztet:

```
check_vids() {
    ls -1 ~/Video/*.mkv > /dev/null 2>&1
    if [ "$?" = "0" ];then
        echo -e "Ne felejtsd, nem tarolhatsz 5 videofajlnal tobbet a Video
mappadban.\nThanks."
    else
    echo -e "Nincsenek videok a Video mappadban. Maximum 5-ot nyugodtan
tarolhatsz.\nThanks."
    fi
}
```

A `check_vids` három dolgot csinál:

- Kilistázza az `mkv` fájlokat a `~/Video` mappában,a kimenetet — és az esetleges hibákat — az úgynevezett *bit-bucket*-be (`/dev/null`) küldi.
- Az előző parancs kimeneti eredményét vizsgálja.
- A teszt eredményétől függően két üzenet egyikét adja ki.

A második függvény az `editors` függvényünk módosított változata:

```
editors() {
```

```
editor=emacs

echo "Hi, $USER!"
echo "Az $editor tobb, mint egy szovegszerkeszto!"

check_vids
}

editors
```

Fontos, hogy két dolgot megfigyeljünk:

- Az `editors` utolsó parancsa meghívja a `check_vids` parancsot, így mindkét függvény láncolva lesz: Az üdvözlés, a dicséret, valamint az ellenőrzés és a figyelmeztetés egymás után hajtódik végre.
- Az `editors` maga a belépési pont a függvénysorozatba, ezért az utolsó sorban (`editors`) hívjuk meg.

Jelentkezzük be `user2`-ként és lássuk, működik-e:

```
# su - user2
Hi, user2!
Az emacs tobb, mint egy szovegszerkeszto!
Ne felejtsd, nem tarolhatsz 5 videofajlnal tobbet a Video mappadban.
Thanks.
```

Gyakorló feladatok

1. Töltsük ki a táblázatot “Igen” vagy “Nem” szavakkal, az aliasok és függvények képességeinek figyelembevételével:

Funkció	**Aliasok?**	**Függvények?**
Használhatók lokális változók		
Használhatók környezeti változók		
Kivédhető a `\` használatával		
Lehet rekurzív		
Nagyon produktív, ha pozicionális paraméterekkel használjuk		

2. Írjuk be a parancsot, ami a rendszerben lévő összes aliast kilistázza:

3. Írjunk egy `logg` nevű aliast, ami az összes `ogg` fájlt kilistázza a `~/Music` mappában — soronként egyet:

4. Hívjuk meg az aliast, hogy megbizonyosodjunk róla, működik-e:

5. Most módosítsuk az aliast úgy, hogy az aktuális felhasználó nevét és a felsorolás előtt egy kettőspontot is kiírjon:

6. Hívjuk meg újra, hogy megbizonyosodjunk, ez az új verzió is működik-e:

7. Listázzuk ki újra az összes aliast, hogy megnézzük, a `logg` aliasunk szerepel-e a listában:

8. Töröljük az aliast:

9. Tanulmányozzuk az “Alias neve” és az “Aliasolt parancs(ok)” oszlopokat és rendeljük az aliasokat az értékeikhez:

Alias neve	Aliasolt parancs(ok)	Alias hozzárendelés
`b`	`bash`	
`bash_info`	`which bash` + `echo "$BASH_VERSION"`	
`kernel_info`	`uname -r`	
`greet`	`echo Hi, $USER!`	
`computer`	`pc=slimbook` + `echo My computer is a $pc`	

10. Mint `root`, írjunk egy `my_fun` nevű függvényt az `/etc/bash.bashrc`-be! A függvénynek köszönnie kell a felhasználónak, és meg kell mondania, hogy mi az elérési útvonala. Indítsuk el úgy, hogy a felhasználó mindkét üzenetet megkapja minden egyes bejelentkezéskor:

11. Lépjünk be `user2`-ként, hogy megnézzük, működik-e:

12. Írjuk meg ugyanezt a függvényt mindössze egy sorban:

13. Hívjuk meg a függvényt:

14. Vonjuk vissza (unset) a függvényt:

15. Ez a `special_vars` függvény módosított változata:

```
$ special_vars2() {
> echo $#
> echo $_
> echo $1
> echo $4
> echo $6
> echo $7
> echo $_
```

```
> echo $@
> echo $?
> }
```

Ezzel a paranccsal hívjuk meg:

```
$ special_vars2 crying cockles and mussels alive alive oh
```

Találjuk ki a kimeneteket:

Referencia	**Érték**
`echo $#`	
`echo $_`	
`echo $1`	
`echo $4`	
`echo $6`	
`echo $7`	
`echo $_`	
`echo $@`	
`echo $?`	

16. A “Függvény a függvényben” című szekcióban található mintafüggvény (`check_vids`) alapján írjunk egy `check_music` nevű függvényt, amelyet beilleszthetünk egy `bash` startup scriptbe, amely elfogad pozicionális paramétereket, hogy könnyen módosíthassuk:
    - az ellenőrizni kívánt fájlformátum: `ogg`
    - a mappa, ahol a fájlok vannak: `~/Music`
    - a tárolt fájlok típusa: `music`
    - a tárolt fájlok száma: `7`

Gondolkodtató feladatok

1. A csak olvasható függvények azok, amelyek tartalmát nem tudjuk módosítani. Végezzünk kutatást a *readonly functions* témakörben, és töltsük ki az alábbi táblázatot:

Függvény neve	**Csak olvashatóvá tétel**	**Minden csak olvasható függvény listázása**
`my_fun`		

2. Keressük meg az interneten, hogyan lehet módosítani az `PS1`-t és bármi mást, amire szükségünk lehet, hogy írjunk egy `fyi` nevű függvényt (amit egy startup scriptben kell elhelyezni), amely a következő információkat adja meg a felhasználónak:
   - a felhasználó neve
   - home mappa
   - a host neve
   - az operációs rendszer típusa
   - a futtatható fájlok keresési útvonala
   - levelezési mappa
   - milyen gyakran kerül ellenőrzésre a levelezés
   - hány shell mélységű az aktuális munkamenet
   - prompt (ezt úgy kell módosítani, hogy `<user>@<host-date>` szerepeljen)

Összefoglalás

Ebben a leckében megtanultuk:

- Mind az aliasok, mind a függvények a shell fontos funkciói, amelyek lehetővé teszik számunkra, hogy ismétlődő kódblokkokat foglaljunk egységbe bennük.
- Az aliasok hasznosak a hosszú és/vagy bonyolult parancsok rövidebb változatainak elkészítéséhez.
- A függvények olyan eljárások, amelyek logikát valósítanak meg, és lehetővé teszik a feladatok automatizálását, különösen, ha scriptekben használjuk őket.
- Az aliasok és függvények írásának szintaxisát.
- Különböző parancsok összekapcsolását a pontosvessző (`;`) segítségével.
- Hogyan használjuk helyesen az idézőjeleket az aliasoknál.
- Hogyan lehet az aliasokat és függvényeket állandóvá tenni.
- Bash speciális beépített változók: `$?`, `$$`, `$!`, pozicionális paraméterek (`$0`-`$9`), `$#`, `$@`, `$*` és `$_`.
- Hogyan használjunk változókat és pozicionális paramétereket függvényekkel.
- Hogyan használjunk függvényeket scriptekben.
- Hogyan hívhatunk meg egy függvényt egy aliasból.
- Hogyan hívhatunk meg egy függvényt egy másik függvényből.
- A `bash` scriptek készítésének alapjai.

A leckében használt parancsok és kulcsszavak

`alias`
: Aliasok létrehozása.

`unalias`
: Aliasok eltávolítása.

`cd`
: Mappaváltás.

`grep`
: Mintának megfelelő sorok kiíratása.

`function`

Függvények létrehozásához való shell kulcsszó.

`.`

Fájl sourceolása.

`source`

Fájl sourceolása.

`ps`

Pillanatfelvétel a jelenlegi folyamatokról.

`echo`

Egy szöveg megjelenítése.

`chmod`

Fájl üzemmódbitjeinek módosítása, például a fájl futtathatóvá tétele.

`unset`

Változók és függvények visszavonása.

`su`

Felhasználó ID megváltoztatása vagy superuserre váltás.

Válaszok a gyakorló feladatokra

1. Töltsük ki a táblázatot “Igen” vagy “Nem” szavakkal, az aliasok és függvények képességeinek figyelembevételével:

Funkció	**Aliasok?**	**Függvények?**
Használhatók lokális változók	Igen	Igen
Használhatók környezeti változók	Igen	Igen
Kivédhető a `\` használatával	Igen	Nem
Lehet rekurzív	Igen	Igen
Nagyon produktív, ha pozicionális paraméterekkel használjuk	Nem	Igen

2. Írjuk be a parancsot, ami a rendszerben lévő összes aliast kilistázza:

```
alias
```

3. Írjunk egy `logg` nevű aliast, ami az összes `ogg` fájlt kilistázza a `~/Music` mappában — soronként egyet:

```
alias logg='ls -1 ~/Music/*ogg'
```

4. Hívjuk meg az aliast, hogy megbizonyosodjunk róla, működik-e:

```
logg
```

5. Most módosítsuk az aliast úgy, hogy az aktuális felhasználó nevét és a felsorolás előtt egy kettőspontot is kiírjon:

```
alias logg='echo $USER:; ls -1 ~/Music/*ogg'
```

6. Hívjuk meg újra, hogy megbizonyosodjunk, ez az új verzió is működik-e:

```
logg
```

7. Listázzuk ki újra az összes aliast, hogy megnézzük, a `logg` aliasunk szerepel-e a listában:

```
alias
```

8. Töröljük az aliast:

```
unalias logg
```

9. Tanulmányozzuk az “Alias neve” és az “Aliasolt parancs(ok)” oszlopokat és rendeljük az aliasokat az értékeikhez:

Alias neve	Aliasolt parancs(ok)	Alias hozzárendelés
`b`	`bash`	`alias b=bash`
`bash_info`	`which bash` + `echo "$BASH_VERSION"`	`alias bash_info='which bash; echo "$BASH_VERSION"'`
`kernel_info`	`uname -r`	`alias kernel_info='uname -r'`
`greet`	`echo Hi, $USER!`	`alias greet='echo Hi, $USER'`
`computer`	`pc=slimbook` + `echo My computer is a $pc`	`alias computer='pc=slimbook; echo My computer is a $pc'`

NOTE Az aposztrófok idézőjelekre cserélhetők.

10. Mint `root`, írjunk egy `my_fun` nevű függvényt az `/etc/bash.bashrc`-be! A függvénynek köszönnie kell a felhasználónak, és meg kell mondania, hogy mi az elérési útvonala. Indítsuk el úgy, hogy a felhasználó mindkét üzenetet megkapja minden egyes bejelentkezéskor:

A lehetőség:

```
my_fun() {
echo Hello, $USER!
```

```
echo Your path is: $PATH
}
my_fun
```

B lehetőség:

```
function my_fun {
echo Hello, $USER!
echo Your path is: $PATH
}
my_fun
```

11. Lépjünk be `user2`-ként, hogy megnézzük, működik-e:

```
su - user2
```

12. Írjuk meg ugyanezt a függvényt mindössze egy sorban:

A lehetőség:

```
my_fun() { echo "Hello, $USER!"; echo "Your path is: $PATH"; }
```

B lehetőség:

```
function my_fun { echo "Hello, $USER!"; echo "Your path is: $PATH"; }
```

13. Hívjuk meg a függvényt:

```
my_fun
```

14. Vonjuk vissza (unset) a függvényt:

```
unset -f my_fun
```

15. Ez a `special_vars` függvény módosított változata:

```
$ special_vars2() {
```

```
> echo $#
> echo $_
> echo $1
> echo $4
> echo $6
> echo $7
> echo $_
> echo $@
> echo $?
> }
```

Ezzel a paranccsal hívjuk meg:

```
$ special_vars2 crying cockles and mussels alive alive oh
```

Találjuk ki a kimeneteket:

Referencia	**Érték**
`echo $#`	`7`
`echo $_`	`7`
`echo $1`	`crying`
`echo $4`	`mussels`
`echo $6`	`alive`
`echo $7`	`oh`
`echo $_`	`oh`
`echo $@`	`crying cockles and mussels alive alive oh`
`echo $?`	`0`

16. A “Függvény a függvényben” című szekcióban található mintafüggvény (`check_vids`) alapján írjunk egy `check_music` nevű függvényt, amelyet beilleszthetünk egy `bash` startup scriptbe, amely elfogad pozicionális paramétereket, hogy könnyen módosíthassuk:
    - az ellenőrizni kívánt fájlformátum: `ogg`
    - a mappa, ahol a fájlok vannak: `~/Music`
    - a tárolt fájlok típusa: `music`

- a tárolt fájlok száma: `7`

```
check_music() {
    ls -1 ~/$1/*.$2 > ~/.mkv.log 2>&1
    if [ "$?" = "0" ];then
        echo -e "Ne felejtsd, nem tarolhatsz $3 $4 fajlnal tobbet a $1
mappaban.\nThanks."
    else
     echo -e "Nincs $4 fajl az $1 mappadban. $3 fajlt nyugodtan tarolhatsz.\nThanks."
    fi
}

check_music Music ogg 7 music
```

Válaszok a gondolkodtató feladatokra

A csak olvasható függvények azok, amelyek tartalmát nem tudjuk módosítani. Végezzünk kutatást a *readonly functions* témakörben, és töltsük ki az alábbi táblázatot:

+

Függvény neve	Csak olvashatóvá tétel	Minden csak olvasható függvény listázása
`my_fun`	`readonly -f my_fun`	`readonly -f`

1. Keressük meg az interneten, hogyan lehet módosítani a `PS1`-t és bármi mást, amire szükségünk lehet, hogy írjunk egy `fyi` nevű függvényt (amit egy startup scriptben kell elhelyezni), amely a következő információkat adja meg a felhasználónak:
   - a felhasználó neve
   - home mappa
   - a host neve
   - az operációs rendszer típusa
   - a futtatható fájlok keresési útvonala
   - levelezési mappa
   - milyen gyakran kerül ellenőrzésre a levelezés
   - hány shell mélységű az aktuális munkamenet
   - prompt (ezt úgy kell módosítani, hogy `<user>@<host-date>` szerepeljen)

```
fyi() {
    echo -e "For your Information:\n
    Username: $USER
    Home directory: $HOME
    Host: $HOSTNAME
    Operating System: $OSTYPE
    Path for executable files: $PATH
    Your mail directory is $MAIL and is searched every $MAILCHECK seconds.
    The current level of your shell is: $SHLVL"
    PS1="\u@\h-\d "
}

fyi
```

105.2 Egyszerű scriptek testre szabása vagy írása

Hivatkozás az LPI célkitűzésre

LPIC-1 5.0, Exam 102, Objective 105.2

Súlyozás

4

Kulcsfontosságú ismeretek

- Standard sh szintaxis (ciklusok, tesztek) használata
- Parancshelyettesítés használata
- Visszatérési értékek tesztelése a siker vagy a sikertelenség, illetve a parancs által szolgáltatott egyéb információk tekintetében
- Láncolt parancsok végrehajtása
- Feltételes üzenetküldés a szuperfelhasználónak
- A scriptértelmező helyes kiválasztása a shebang (#!) soron keresztül
- A scriptek helyének, tulajdonjogának, végrehajtásának és suid-jogainak kezelése

A használt fájlok, kifejezések és segédprogramok listája

- `for`
- `while`
- `test`
- `if`
- `read`
- `seq`
- `exec`

- ||
- &&

105.2 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	105 Shellek és shell scriptelés
Fejezet:	105.2 Egyszerű scriptek testre szabása vagy írása
Lecke:	1/2

Bevezetés

A Linux shell környezet lehetővé teszi olyan fájlok — úgynevezett *scriptek* — használatát, amelyek a rendszerben elérhető bármely program parancsait tartalmazzák, a shell beépített parancsaival kombinálva, ezzel automatizálhatóvá téve a felhasználó és/vagy a rendszer egyéni feladatait. Valójában az operációs rendszer karbantartási feladatainak nagy részét parancsokból, döntési struktúrákból és feltételes ciklusokból álló scriptek végzik. Bár a scripteket legtöbbször magával az operációs rendszerrel kapcsolatos feladatokhoz tervezik, hasznosak lehetnek felhasználóorientált feladatokhoz is, mint például fájlok tömeges átnevezése, adatgyűjtés és -elemzés, vagy bármilyen más módon ismétlődő parancssori tevékenység.

A scriptek nem mások, mint szövegfájlok, amelyek úgy viselkednek, mint a programok. Egy tényleges program — az interpreter (értelmező) — olvassa és végrehajtja a script-fájlban felsorolt utasításokat. Az interpreter interaktív munkamenetet is indíthat, ahol a parancsok — beleértve a scripteket is — beolvasásra és végrehajtásra kerülnek, ahogyan azokat beírják, mint a Linux shell munkamenetek esetében. A script-fájlok csoportosíthatják azokat az utasításokat és parancsokat, amelyek túl bonyolulttá válnak ahhoz, hogy aliasokként vagy egyéni shell-funkcióként lehessen őket megvalósítani. Továbbá a script-fájlok a hagyományos programokhoz hasonlóan

karbantarthatók, és mivel csak szövegfájlok, bármely egyszerű szövegszerkesztővel létrehozhatók és módosíthatók.

Script-struktúra és -végrehajtás

Alapvetően egy script-fájl parancsok rendezett sorozata, amelyeket egy megfelelő parancsértelmezőnek kell végrehajtania. Az, hogy egy parancsértelmező hogyan olvassa be a parancsfájlt, változó, és a Bash shell munkamenetben is különböző módjai vannak, de a parancsfájl alapértelmezett interpretere az lesz, amelyet a parancsfájl első sorában, közvetlenül a `#!` (az úgynevezett *shebang*) karakterek után jelzünk. Egy Bash shell utasításokat tartalmazó scriptben az első sorban a `#!/bin/bash` sornak kell állnia. Ennek a sornak a feltüntetésével a fájlban lévő összes utasítás értelmezője a `/bin/bash` lesz. Az első sor kivételével az összes többi, `#` hash karakterrel kezdődő sort a program figyelmen kívül hagyja, így azok emlékeztetők és megjegyzések elhelyezésére használhatók. Az üres sorok szintén figyelmen kívül maradnak. Egy nagyon tömör shell script-fájlt tehát a következőképpen lehet megírni:

```
#!/bin/bash

# A very simple script

echo "Cheers from the script file! Current time is: "

date +%H:%M
```

Ez a script csak két utasítást tartalmaz a `/bin/bash` interpreter számára: a beépített `echo` parancsot és a `date` parancsot. Egy script-fájl futtatásának legalapvetőbb módja az interpreter futtatása a script elérési útvonalával, mint argumentummal. Tehát, feltételezve, hogy az előző példát egy `script.sh` nevű script-fájlba mentettük az aktuális mappába, a Bash a következő paranccsal fogja beolvasni és értelmezni:

```
$ bash script.sh
Cheers from the script file! Current time is:
10:57
```

A `echo` parancs automatikusan új sort ad a tartalom megjelenése után, de az `-n` kapcsolóval ez elnyomható. Így az `echo -n` használata a scriptben azt eredményezi, hogy mindkét parancs kimenete ugyanabban a sorban jelenik meg:

```
$ bash script.sh
```

```
Cheers from the script file! Current time is: 10:57
```

Bár nem kötelező használnunk, a `.sh` utótag segít a shell-scriptek azonosításában a fájlok listázása és keresése során.

TIP

A Bash a `#!` után megadott parancsot hívja meg a script-fájl interpreteréként. Hasznos lehet például a shebang alkalmazása más script-nyelvek, például a *Python* (`#!/usr/bin/python`), *Perl* (`#!/usr/bin/perl`) vagy *awk* (`#!/usr/bin/awk`) esetén.

Ha a script-fájlt a rendszer más felhasználói is futtatni akarják, fontos ellenőrizni, hogy a megfelelő olvasási jogosultságok be vannak-e állítva. A `chmod o+r script.sh` parancs olvasási jogosultságot ad a rendszer összes felhasználójának, így a `bash` parancs argumentumaként a script-fájl elérési útvonalát megadva a `script.sh` parancsot futtathatják. Alternatív megoldásként a script-fájlnak megadható a végrehajtási bit jogosultsága, így a fájl hagyományos parancsként is végrehajtható. A végrehajtási bitet a `chmod` paranccsal aktiválhatjuk a script-fájlon:

```
$ chmod +x script.sh
```

A végrehajtási bit engedélyezése esetén az aktuális mappában lévő `script.sh` nevű script-fájl közvetlenül a `./script.sh` paranccsal hajtható végre. A `PATH` környezeti változóban felsorolt mappában elhelyezett scriptek a teljes elérési útvonaluk nélkül is elérhetők lesznek.

WARNING

A korlátozott műveleteket végrehajtó scriptek SUID jogosultsága aktiválva lehet, így a közönséges felhasználók is futtathatják a scriptet root jogosultságokkal. Ebben az esetben nagyon fontos biztosítani, hogy a root-on kívül más felhasználó ne rendelkezzen írási jogosultsággal a fájlban. Ellenkező esetben egy közönséges felhasználó módosíthatja a fájlt, hogy tetszőleges és potenciálisan káros műveleteket hajtson végre.

A parancsok elhelyezése és behúzása a script-fájlokban nem túl szigorú. A shell script minden sora közönséges shell-parancsként kerül végrehajtásra, ugyanabban a sorrendben, ahogy a sor a script-fájlban megjelenik, és ugyanazok a szabályok, amelyek a shell promptra vonatkoznak, minden egyes script sorra külön-külön is érvényesek. Lehetőség van két vagy több parancsot is elhelyezni egy sorban, pontosvesszővel elválasztva:

```
echo "Cheers from the script file! Current time is:" ; date +%H:%M
```

Bár ez a formátum időnként kényelmes, használata opcionális, mivel soronként egy-egy parancsot is elhelyezhetünk, és ezek ugyanúgy végrehajtódnak, mintha pontosvesszővel lennének elválasztva. Más szóval a pontosvesszőt a Bash script-fájlokban helyettesítheti egy új sor karakter.

Egy script végrehajtásakor a benne szereplő parancsok nem közvetlenül az aktuális munkamenetben kerülnek végrehajtásra, hanem egy új Bash-folyamat, az úgynevezett *sub-shell* hajtja végre őket. Ez megakadályozza, hogy a script felülírja az aktuális munkamenet környezeti változóit, és azt is, hogy felügyelet nélkül maradjanak módosítások az aktuális munkamenetben. Ha a cél az, hogy a script tartalma az aktuális shell munkamenetben fusson, akkor a `source script.sh` vagy `. script.sh` (figyeljünk arra, hogy a pont és a script neve között szóköz van) parancsszóval kell végrehajtani.

Mint bármely más parancs végrehajtásakor, a shell prompt csak akkor lesz újra elérhető, amikor a script befejezi a végrehajtását, és a kilépési állapotkódja a `$?` változóban lesz elérhető. Ennek a viselkedésnek a megváltoztatásához, hogy az aktuális shell is véget érjen, amikor a script futása befejeződik, a script — vagy bármely más parancs — előtt az `exec` parancsnak kell állnia. Ez a parancs az aktuális shell munkamenet kilépési állapotkódját is lecseréli a sajátjára.

Változók

A shell scriptekben a változók ugyanúgy viselkednek, mint az interaktív munkamenetekben, mivel az interpreter ugyanaz. Például a `SOLUTION=42` formátum (szóközök nélkül az egyenlőségjel körül) a `SOLUTION` nevű változóhoz a `42` értéket rendeli. A konvenció szerint a változók neveinél nagybetűket használunk, de ez nem kötelező. A változók nevei azonban nem kezdődhetnek nem alfabetikus karakterekkel.

A felhasználó által létrehozott közönséges változókon kívül a Bash scriptek egy sor speciális változóval is rendelkeznek, amelyeket paramétereknek (*parameters*) neveznek. A közönséges változókkal ellentétben a paraméterek nevei egy nem betűjeles karakterrel kezdődnek, amely a funkciót jelöli. A scriptnek átadott argumentumok és egyéb hasznos információk a `$0`, `$*`, `$?` stb. típusú paraméterekben tárolódnak, ahol a dollárjelet követő karakter jelzi a lekérdezendő információt:

`$*`

A scriptnek átadott összes argumentum.

`$@`

A scriptnek átadott összes argumentum. Ha idézőjelekkel együtt használjuk, mint `"$@"`, akkor minden argumentumot idézőjelek közé zárunk.

`$#`

Az argumentumok száma.

`$0`

A script-fájl neve.

`$!`

Az utoljára végrehajtott program PID-je.

`$$`

Az aktuális shell PID-je.

`$?`

Az utolsó befejezett parancs numerikus kilépési állapotkódja. A POSIX szabványos folyamatok esetében a `0` numerikus érték azt jelenti, hogy az utolsó parancs sikeresen végrehajtásra került, ami a shell scriptekre is vonatkozik.

A *pozicionális paraméter* olyan paraméter, amelyet egy vagy több számjegy jelöl, kivéve az egyjegyű `0`-t. Például az `$1` változó a scriptnek adott első argumentumnak felel meg (első pozicionális paraméter), `$2` a második argumentumnak, és így tovább. Ha egy paraméter pozíciója nagyobb, mint kilenc, akkor a paraméterre kapcsos zárójelben kell hivatkozni, mint például `${10}`, `${11}`, stb.

A közönséges változók viszont arra szolgálnak, hogy manuálisan beillesztett értékeket vagy más parancsok által generált kimenetet tároljanak. A `read` parancsot például arra használhatjuk a scriptben, hogy a végrehajtása során a felhasználótól inputot kérjünk:

```
echo "Do you want to continue (y/n)?"
read ANSWER
```

A visszakapott érték az `ANSWER` változóban kerül tárolásra. Ha a változó neve nincs megadva, akkor alapértelmezés szerint a `REPLY` változó neve lesz használva. A `read` paranccsal egynél több változó egyidejű olvasására is lehetőség van:

```
echo "Type your first name and last name:"
read NAME SURNAME
```

Ebben az esetben minden szóközzel elválasztott kifejezés a `NAME` és a `SURNAME` változókhoz lesz hozzárendelve. Ha a megadott kifejezések száma nagyobb, mint a változók száma, akkor a többlet kifejezések az utolsó változóba kerülnek. A `read` maga is megjelenítheti az üzenetet a felhasználónak a `-p` kapcsolóval, így az `echo` parancs ebben az esetben feleslegessé válik:

```
read -p "Type your first name and last name:" NAME SURNAME
```

A rendszerfeladatokat végrehajtó scripteknek gyakran szükségük van más programok által

szolgáltatott információkra. A *backtick notáció* arra használható, hogy egy parancs kimenetét egy változóban tároljuk:

```
$ OS=`uname -o`
```

A példában az `uname -o` parancs kimenete az `OS` változóban lesz tárolva. A `$()` paranccsal azonos eredményt kapunk:

```
$ OS=$(uname -o)
```

Egy változó hossza, azaz a benne lévő karakterek száma úgy adódik vissza, hogy a változó neve elé egy hasht (`#`) írunk. Ez a funkció azonban megköveteli, hogy a változót a kapcsos zárójeles szintaxissal jelezzük:

```
$ OS=$(uname -o)
$ echo $OS
GNU/Linux
$ echo ${#OS}
9
```

A Bash egydimenziós tömbváltozókat is tartalmaz, így az egymáshoz kapcsolódó elemek halmaza egyetlen változónévvel tárolható. Egy tömb minden eleméhez tartozik egy numerikus index, amelyet a megfelelő elemben lévő értékek írásához és olvasásához kell használni. A közönséges változóktól eltérően a tömböket a Bash beépített `declare` parancsával kell deklarálni. Például egy `SIZES` nevű változó tömbként való deklarálásához:

```
$ declare -a SIZES
```

A tömbök implicit módon is deklarálhatók, amikor egy előre definiált elemlistából töltjük fel őket, a zárójeles jelölés használatával:

```
$ SIZES=( 1048576 1073741824 )
```

A példában a két nagy egész számot (integer) a `SIZES` tömbben tároltuk el. A tömb elemeire a hivatkozást kapcsos és szögletes zárójelekkel kell megadni, különben a Bash nem fogja helyesen megváltoztatni vagy megjeleníteni az elemet. Mivel a tömbindexek 0-val kezdődnek, az első elem tartalma a `${SIZES[0]}`-ban, a második elem a `${SIZES[1]}`-ban van, és így tovább:

```
$ echo ${SIZES[0]}
1048576
$ echo ${SIZES[1]}
1073741824
```

Az olvasással szemben a tömbelem tartalmának módosítása a kapcsos zárójelek nélkül történik (pl. `SIZES[0]=1048576`). A közönséges változókhoz hasonlóan a tömb elemének hosszát hash karakterrel adjuk vissza (pl. `${#SIZES[0]}` az `SIZES` tömb első elemének hossza). A tömb összes elemének számát kapjuk meg, ha indexként `@` vagy `*` szerepel:

```
$ echo ${#SIZES[@]}
2
$ echo ${#SIZES[*]}
2
```

A tömbök deklarálhatók úgy is, hogy egy parancs kimenetét használják kezdeti elemként a parancs behelyettesítésével. A következő példa egy olyan Bash tömb létrehozását mutatja be, amelynek elemei az aktuális rendszer támogatott fájlrendszerei:

```
$ FS=( $(cut -f 2 < /proc/filesystems) )
```

A `cut -f 2 < /proc/filesystems` parancs megjeleníti a futó kernel által jelenleg támogatott összes fájlrendszert (ahogyan az a `/proc/filesystems` fájl második oszlopában szerepel), így az `FS` tömb most már minden támogatott fájlrendszerhez tartalmaz egy elemet. Bármilyen szöveges tartalom használható a tömb inicializálására, mivel alapértelmezés szerint minden *szóköz* (space), *tab* vagy *újsor* (newline) karakterekkel határolt kifejezés tömbelemmé válik.

TIP A Bash a környezeti változó `$IFS` (*Input Field Separator*) minden egyes karakterét elválasztóként kezeli. Ha például a mezőhatárolót csak újsor karakterekre szeretnénk változtatni, akkor az IFS változót az `IFS=$'\n'` paranccsal kell visszaállítani.

Aritmetikai kifejezések

A Bash a beépített `expr` paranccsal praktikus módszert biztosít az egész számokkal végzett aritmetikai műveletek elvégzésére. Két numerikus változó, például `$VAL1` és `$VAL2` összeadható a következő paranccsal:

```
$ SUM=`expr $VAL1 + $VAL2`
```

A példa eredményeinek értéke a `$SUM` változóban lesz elérhető. Az `expr` parancsot helyettesíthetjük`$(())-vel, így az előző példa az alábbi módon átírható: `SUM=$(($VAL1 + $VAL2))`. A hatványkifejezésekhez a dupla csillag operátort használhatjuk, így a korábbi tömbdeklaráció `SIZES=(1048576 1073741824)` az alábbi is lehet `SIZES=($((1024**2)) $((1024**3)))`.

A parancshelyettesítés aritmetikai kifejezésekben is használható. Például a `/proc/meminfo` fájl részletes információkat tartalmaz a rendszermemóriáról, beleértve a RAM-ban lévő szabad bájtok számát is:

```
$ FREE=$(( 1000 * `sed -nre '2s/[^[:digit:]]//gp' < /proc/meminfo` ))
```

A példa azt mutatja, hogy a `sed` parancs hogyan használható a `/proc/meminfo` tartalmának feldolgozására az aritmetikai kifejezésen belül. A `/proc/meminfo` fájl második sora a szabad memória mennyiségét tartalmazza ezer bájtban, így az aritmetikai kifejezés megszorozza 1000-rel, hogy megkapja a RAM-ban lévő szabad bájtok számát.

Feltételes végrehajtás

Egyes parancsfájlok általában nem a parancsfájlban lévő összes parancsot hajtják végre, hanem csak azokat, amelyek megfelelnek egy előre meghatározott feltételnek. Például egy karbantartási script csak akkor küldhet figyelmeztető üzenetet a rendszergazda e-mail címére, ha egy parancs végrehajtása sikertelen. A Bash speciális módszereket biztosít a parancsok végrehajtásának sikerességének értékelésére, valamint általános feltételes struktúrákat, amelyek inkább a népszerű programozási nyelvekben találhatóakhoz hasonlítanak.

A parancsok `&&`-vel való elválasztásával a jobb oldali parancs csak akkor kerül végrehajtásra, ha a bal oldali parancs nem ütközött hibába, azaz ha a kilépési állapota egyenlő volt a `0-val`:

```
COMMAND A && COMMAND B && COMMAND C
```

Az ellenkező viselkedés akkor következik be, ha a parancsokat `||`-vel választjuk el. Ebben az esetben a következő parancs csak akkor kerül végrehajtásra, ha az előző parancs hibát szenvedett, vagyis ha a visszatérő állapotkódja eltér a 0-tól.

Minden programozási nyelv egyik legfontosabb jellemzője, hogy a parancsokat előre meghatározott feltételek függvényében lehet végrehajtani. A parancsok feltételes végrehajtásának legegyszerűbb módja a Bash beépített `if` parancsának használata, amely egy vagy több parancsot csak akkor hajt végre, ha az argumentumként megadott parancs 0 (siker) állapotkódot ad vissza. Egy másik parancs, a `test`, sokféle speciális feltétel értékelésére használható, ezért többnyire a `if`

-vel együtt használják. A következő példában a `Confirmed: /bin/bash is executable.` üzenet jelenik meg, ha a `/bin/bash` fájl létezik és futtatható:

```
if test -x /bin/bash ; then
  echo "Confirmed: /bin/bash is executable."
fi
```

A `-x` kapcsoló hatására a `test` parancs csak akkor ad 0 státuszkódot, ha a megadott elérési út egy futtatható fájl. A következő példa egy másik módot mutat, amellyel pontosan ugyanazt az eredményt érhetjük el, mivel a szögletes zárójelek a `test` helyettesítésére is használhatók:

```
if [ -x /bin/bash ] ; then
  echo "Confirmed: /bin/bash is executable."
fi
```

Az `else` utasítás opcionális az `if` esetén, de ha szerepel, akkor egy olyan parancsot vagy parancssorozatot határozhat meg, amelyet akkor kell végrehajtani, ha a feltételes kifejezés nem igaz:

```
if [ -x /bin/bash ] ; then
  echo "Confirmed: /bin/bash is executable."
else
  echo "No, /bin/bash is not executable."
fi
```

Az `if` struktúráknak mindig `fi`-vel kell végződniük, hogy a Bash interpreter tudja, hol végződnek a feltételes parancsok.

Scriptek kimenete

Még akkor is, ha a script célja csak fájlorientált műveletek elvégzése, fontos, hogy a standard kimeneten megjelenjenek az előrehaladással kapcsolatos üzenetek, hogy a felhasználó folyamatosan értesüljön az esetleges problémákról, és végül ezeket az üzeneteket felhasználhassa a műveleti logok létrehozásához.

A Bash beépített `echo` parancsát általában egyszerű szöveges stringek megjelenítésére használjuk, de néhány bővített funkcióval is rendelkezik. Az `-e` kapcsolóval az `echo` parancs képes speciális karakterek megjelenítésére kivédett (escape) szekvenciák (egy speciális karaktert jelölő backslash szekvencia) használatával. Például:

```
#!/bin/bash

# Get the operating system's generic name
OS=$(uname -o)

# Get the amount of free memory in bytes
FREE=$(( 1000 * `sed -nre '2s/[^[:digit:]]//gp' < /proc/meminfo` ))

echo -e "Operating system:\t$OS"
echo -e "Unallocated RAM:\t$(( $FREE / 1024**2 )) MB"
```

Míg az idézőjelek használata kapcsolók nélküli `echo` használatakor opcionális, addig az `-e` kapcsoló használata esetén kötelező, különben a speciális karakterek nem jelenhetnek meg helyesen. Az előző scriptben mindkét `echo` parancs a `\t` tabulátor karaktert használja a szöveg igazítására, ami a következő kimenetet eredményezi:

```
Operating system:       GNU/Linux
Unallocated RAM:        1491 MB
```

A kimeneti sorok elválasztására a `\n` újsor karakter használható, így pontosan ugyanazt a kimenetet kapjuk, ha a két `echo` parancsból egyet csinálunk:

```
echo -e "Operating system:\t$OS\nUnallocated RAM:\t$(( $FREE / 1024**2 )) MB"
```

Bár a legtöbb szöveges üzenet megjelenítésére alkalmas, a `echo` parancs nem biztos, hogy megfelelő a specifikusabb szövegminták megjelenítésére. A Bash beépített `printf` parancsa nagyobb kontrollt ad a változók megjelenítésének módjára. A `printf` parancs az első argumentumot használja a kimenet formátumaként, ahol a placeholdereket a következő argumentumok helyettesítik a parancssorban való megjelenésük sorrendjében. Például az előző példa üzenete a következő `printf` paranccsal generálható:

```
printf "Operating system:\t%s\nUnallocated RAM:\t%d MB\n" $OS $(( $FREE / 1024**2 ))
```

A `%s` placeholder szöveges tartalomra vonatkozik (a `$OS` változó kerül a helyére), a `%d` pedig egész számokra (helyébe a RAM-ban lévő szabad megabájtok száma kerül). A `printf` nem illeszt újsor karaktert a szöveg végére, ezért a `\n` karaktert kell a minta végére tenni, ha szükséges. A teljes mintát egyetlen argumentumként kell értelmezni, ezért azt idézőjelek közé kell zárni.

TIP A `printf` placeholder-formátuma testre szabható a C programozási nyelv `printf`

> függvénye által használt formátummal. A `printf` függvény teljes hivatkozása megtalálható a kézikönyv oldalán, amelyet a `man 3 printf` paranccsal érhetünk el.

A `printf` esetében a változók a szövegmintán kívül helyezkednek el, ami lehetővé teszi, hogy a szövegmintát egy külön változóban tároljuk:

```
MSG='Operating system:\t%s\nUnallocated RAM:\t%d MB\n'
printf "$MSG" $OS $(( $FREE / 1024**2 ))
```

Ez a módszer különösen hasznos a különböző kimeneti formátumok megjelenítéséhez, a felhasználó igényeitől függően. Megkönnyíti például egy olyan script megírását, amely egy külön szövegmintát alkalmaz, ha a felhasználó egy CSV (*Comma Separated Values*) listát használna az alapértelmezett kimeneti üzenet helyett.

Gyakorló feladatok

1. A `-s` opció a `read` parancshoz hasznos a jelszavak beviteléhez, mivel nem jeleníti meg a képernyőn a beírt tartalmat. Hogyan lehetne a `read` parancsot arra használni, hogy a felhasználó által bevitt értéket a `PASSWORD` változóban tárolja, miközben a beírt tartalmat elrejti?

2. A `whoami` parancs egyetlen célja, hogy megjelenítse a parancsot hívó felhasználó felhasználónevét, így leginkább a parancsot futtató felhasználó azonosítására használják a scripteken belül. Egy Bash scriptben hogyan lehetne a `whoami` parancs kimenetét a `WHO` nevű változóban tárolni?

3. Milyen Bash operátornak kell lennie az `apt-get dist-upgrade` és a `systemctl reboot` parancsok között, ha a root felhasználó csak akkor akarja végrehajtani a `systemctl reboot` parancsot, ha az `apt-get dist-upgrade` sikeresen befejeződött?

Gondolkodtató feladatok

1. Egy újonnan létrehozott Bash script futtatása után a felhasználó a következő hibaüzenetet kapja:

   ```
   bash: ./script.sh: Permission denied
   ```

   Figyelembe véve, hogy a `./script.sh` fájlt ugyanez a felhasználó hozta létre, mi lehet a hiba valószínű oka?

2. Tegyük fel, hogy a `do.sh` nevű script-fájl futtatható, és az `undo.sh` nevű szimbolikus link mutat rá. A scriptből hogyan tudnánk azonosítani, hogy a hívó fájl neve `do.sh` vagy `undo.sh`?

3. Egy megfelelően konfigurált e-mail szolgáltatással rendelkező rendszerben a `mail -s "Maintenance Error" root <<<<"Scheduled task error"` parancs a root felhasználónak küldi el az értesítő e-mailt. Egy ilyen parancsot az olyan felügyelet nélküli feladatokban lehet használni, mint például a *cronjobs*, hogy tájékoztassa a rendszergazdát egy váratlan problémáról. Írjunk egy *if* szerkezetet, amely végrehajtja a fent említett `mail` parancsot, ha az előző parancs — bármi is volt az — kilépési állapota sikertelen.

Összefoglalás

Ez a lecke a Bash shell scriptek megértésének és írásának alapfogalmait tárgyalja. A Shell scriptek minden Linux disztribúció alapvető részét képezik, mivel nagyon rugalmas módot kínálnak a shell környezetben végzett felhasználói és rendszerfeladatok automatizálására. A lecke az alábbiakat veszi végig:

- Shell script struktúra és helyes script-fájl jogosultságok
- Script paraméterek
- Változók használata a felhasználói bemenet beolvasására és a parancsok kimenetének tárolására
- Bash tömbök
- Egyszerű tesztek és feltételes végrehajtás
- Kimenet formázása

A tárgyalt parancsok és eljárások a következők voltak:

- Bash beépített jelölés a parancsok helyettesítéséhez, tömbök bővítéséhez és aritmetikai kifejezésekhez
- Feltételes parancsvégrehajtás a `||` és `&&` operátorokkal
- `echo`
- `chmod`
- `exec`
- `read`
- `declare`
- `test`
- `if`
- `printf`

Válaszok a gyakorló feladatokra

1. A `-s` opció a `read` parancshoz hasznos a jelszavak beviteléhez, mivel nem jeleníti meg a képernyőn a beírt tartalmat. Hogyan lehetne a `read` parancsot arra használni, hogy a felhasználó által bevitt értéket a `PASSWORD` változóban tárolja, miközben a beírt tartalmat elrejti?

   `read -s PASSWORD`

2. A `whoami` parancs egyetlen célja, hogy megjelenítse a parancsot hívó felhasználó felhasználónevét, így leginkább a parancsot futtató felhasználó azonosítására használják a scripteken belül. Egy Bash scriptben hogyan lehetne a `whoami` parancs kimenetét a `WHO` nevű változóban tárolni?

   `WHO=\`whoami\`` vagy `WHO=$(whoami)`

3. Milyen Bash operátornak kell lennie az `apt-get dist-upgrade` és a `systemctl reboot` parancsok között, ha a root felhasználó csak akkor akarja végrehajtani a `systemctl reboot` parancsot, ha az `apt-get dist-upgrade` sikeresen befejeződött?

   Az `&&` operátor, mint `apt-get dist-upgrade && systemctl reboot`.

Válaszok a gondolkodtató feladatokra

1. Egy újonnan létrehozott Bash-script futtatása után a felhasználó a következő hibaüzenetet kapja:

   ```
   bash: ./script.sh: Permission denied
   ```

   Figyelembe véve, hogy a `./script.sh` fájlt ugyanez a felhasználó hozta létre, mi lehet a hiba valószínű oka?

   A `./script.sh` fájlon nincs engedélyezve a futattási jogosultság.

2. Tegyük fel, hogy a `do.sh` nevű script-fájl futtatható, és az `undo.sh` nevű szimbolikus link mutat rá. A scriptből hogyan tudnánk azonosítani, hogy a hívó fájl neve `do.sh` vagy `undo.sh`?

   A `$0` speciális változó tartalmazza a script hívásához használt fájlnevet.

3. Egy megfelelően konfigurált e-mail szolgáltatással rendelkező rendszerben a `mail -s "Maintenance Error" root <<<<"Scheduled task error"` parancs a root felhasználónak küldi el az értesítő e-mailt. Egy ilyen parancsot az olyan felügyelet nélküli feladatokban lehet használni, mint például a *cronjobs*, hogy tájékoztassa a rendszergazdát egy váratlan problémáról. Írjunk egy *if* szerkezetet, amely végrehajtja a fent említett `mail` parancsot, ha az előző parancs — bármi is volt az — kilépési állapota sikertelen.

   `if [ "$?" -ne 0 ]; then mail -s "Maintenance Error" root <<<"Scheduled task error"; fi`

105.2 Lecke 2

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	105 Shellek és shell scriptelés
Fejezet:	105.2 Egyszerű scriptek testre szabása vagy írása
Lecke:	2/2

Bevezetés

A shell scriptek általában arra szolgálnak, hogy automatizálják a fájlokkal és mappákkal kapcsolatos műveleteket, ugyanazokat a műveleteket, amelyeket manuálisan a parancssorból is el lehet végezni. A shell scriptek lefedettsége azonban nem csak a felhasználói dokumentumokra korlátozódik, mivel a Linux operációs rendszer számos aspektusának konfigurálása és interakciója is script-fájlokon keresztül valósul meg.

A Bash shell számos hasznos beépített parancsot kínál shell scriptek írásához, de ezeknek a scripteknek a teljes ereje a Bash beépített parancsai és a Linux rendszerben elérhető számos parancssori segédprogram kombinációjában rejlik.

Kibővített tesztek

A Bash, mint scriptnyelv leginkább a fájlokkal való munkára összpontosít, így a Bash beépített `test` parancsa számos opcióval rendelkezik a fájlrendszer objektumainak (lényegében fájloknak és mappáknak) a tulajdonságainak kiértékelésére. A fájlokra és mappákra fókuszáló tesztek hasznosak például annak ellenőrzésére, hogy egy adott feladat végrehajtásához szükséges fájlok

és mappák jelen vannak-e és olvashatók-e. Ezután egy *if* feltételes szerkezettel társítva a megfelelő műveletsorozatot hajtjuk végre, ha a teszt sikeres.

A `test` parancs kétféle szintaxissal értékelhet kifejezéseket: a tesztkifejezések megadhatók a `test` parancs argumentumaként, vagy szögletes zárójelek közé helyezhetők, ahol a `test` parancs implicit módon van megadva. Így a `/etc` érvényes mappára vonatkozó tesztet írhatjuk `test -d /etc` vagy `[ -d /etc]` formában:

```
$ test -d /etc
$ echo $?
0
$ [ -d /etc ]
$ echo $?
0
```

Amint azt a `$?` speciális változóban szereplő kilépési státuszkódok is megerősítik — a 0 érték azt jelenti, hogy a teszt sikeres volt --, mindkét formátum érvényes mappaként értékelte az `/etc`-t. Feltételezve, hogy egy fájl vagy mappa elérési útvonalát a `$VAR` változóban tároltuk, a következő kifejezések használhatók a `test` argumentumaként vagy szögletes zárójelben:

`-a "$VAR"`
: Kiértékelődik, ha a `VAR`-ban megadott elérési út létezik a fájlrendszerben, és az egy fájl.

`-b "$VAR"`
: Kiértékelődik, ha a `VAR`-ban megadott elérési útvonal egy speciális blokkfájl.

`-c "$VAR"`
: Kiértékelődik, ha a `VAR`-ban szereplő elérési útvonal egy speciális karakteres fájl.

`-d "$VAR"`
: Kiértékelődik, ha a `VAR`-ban szereplő elérési útvonal egy mappa.

`-e "$VAR"`
: Kiértékelődik, ha a `VAR`-ban megadott elérési út létezik a fájlrendszerben.

`-f "$VAR"`
: Kiértékelődik, ha a `VAR`-ban szereplő elérési út létezik, és az egy normál fájl.

`-g "$VAR"`
: Kiértékelődik, ha a `VAR`-ban szereplő elérési útvonal rendelkezik-e SGID jogosultsággal.

`-h "$VAR"`

Kiértékelődik, ha a `VAR`-ban szereplő elérési útvonal szimbolikus hivatkozás.

`-L "$VAR"`

Kiértékelődik, ha a `VAR`-ban szereplő elérési út szimbolikus hivatkozás (mint a `-h`-ban).

`-k "$VAR"`

Kiértékelődik, ha a `VAR`-ban szereplő útvonal rendelkezik a *sticky* bit jogosultsággal.

`-p "$VAR"`

Kiértékelődik, ha a `VAR`-ban szereplő elérési útvonal *pipe* fájl.

`-r "$VAR"`

Kiértékelődik, ha a `VAR`-ban szereplő elérési útvonal olvasható az aktuális felhasználó számára.

`-s "$VAR"`

Kiértékelődik, ha a `VAR`-ban lévő elérési út létezik, és nem üres.

`-S "$VAR"`

Kiértékelődik, ha a `VAR`-ban szereplő elérési útvonal egy socket fájl.

`-t "$VAR"`

Kiértékelődik, ha a `VAR`-ban szereplő elérési útvonal nyitva van egy terminálban.

`-u "$VAR"`

Kiértékelődik, ha a `VAR`-ban szereplő elérési útvonal rendelkezik SUID jogosultsággal.

`-w "$VAR"`

Kiértékelődik, ha a `VAR`-ban szereplő elérési útvonal írható az aktuális felhasználó számára.

`-x "$VAR"`

Kiértékelődik, ha a `VAR`-ban szereplő elérési útvonal az aktuális felhasználó által végrehajtható.

`-O "$VAR"`

Kiértékelődik, ha a `VAR`-ban szereplő útvonal az aktuális felhasználó tulajdonában van.

`-G "$VAR"`

Kiértékelődik, ha a `VAR`-ban szereplő elérési útvonal az aktuális felhasználó effektív csoportjához tartozik.

`-N "$VAR"`

Kiértékelődik, ha a `VAR`-ban szereplő elérési útvonal módosult a legutóbbi elérés óta.

`"$VAR1" -nt "$VAR2"`

Kiértékelődik, ha a `VAR1`-ben lévő elérési útvonal újabb, mint a `VAR2`-ben lévő elérési útvonal, a módosítási dátumok szerint.

`"$VAR1" -ot "$VAR2"`

Kiértékelődik, ha a `VAR1`-ben lévő elérési útvonal régebbi, mint a `VAR2`.

`"$VAR1" -ef "$VAR2"`

Ez a kifejezés True értéket ad, ha a `VAR1`-ben lévő elérési útvonal hardlink a `VAR2`-hez.

A tesztelt változók körül azért ajánlott a kettős idézőjelek használata, mert ha a változó üres, az a `test` parancsnál szintaktikai hibát okozhat. A tesztelési kapcsolóknak szükségük van egy operandus argumentumra, és egy idézőjel nélküli üres változó hibát okozna a szükséges argumentum hiánya miatt. Vannak tetszőleges szöveges változókra vonatkozó tesztek is, amelyeket az alábbiak szerint írunk le:

`-z "$TXT"`

Kiértékelődik, ha a `TXT` változó üres (nulla méretű).

`-n "$TXT"` vagy `test "$TXT"`

Kiértékelődik, ha a `TXT` változó nem üres.

`"$TXT1" = "$TXT2"` vagy `"$TXT1" == "$TXT2"`

Kiértékelődik, ha `TXT1` és `TXT2` egyenlőek.

`"$TXT1" != "$TXT2"`

Kiértékelődik, ha `TXT1` és `TXT2` nem egyenlőek.

`"$TXT1" < "$TXT2"`

Kiértékelődik, ha az `TXT1` az `TXT2` előtt van, ábécé sorrendben.

`"$TXT1" > "$TXT2"`

Kiértékelődik, ha az `TXT1` az `TXT2` után következik, ábécé sorrendben.

Az egyes nyelvek eltérő szabályokkal rendelkezhetnek az ábécé sorrendre vonatkozóan. A konzisztens eredmények elérése érdekében, függetlenül attól, hogy milyen lokalizációs beállításokkal rendelkezik az a rendszer, ahol a scriptet végrehajtjuk, ajánlott a `LANG` környezeti változót `C`-re állítani, mint például a `LANG=C`, mielőtt ábécé sorrendbe rendezéssel kapcsolatos

műveleteket végeznénk. Ez a meghatározás a rendszerüzeneteket is az eredeti nyelven tartja, ezért csak a script hatókörén belül érdemes használni.

A numerikus összehasonlításoknak saját tesztelési módszereik vannak:

`$NUM1 -lt $NUM2`
: Kiértékeli, ha `NUM1` kisebb, mint `NUM2`.

`$NUM1 -gt $NUM2`
: Kiértékelődik, ha `NUM1` nagyobb, mint `NUM2`.

`$NUM1 -le $NUM2`
: Kiértékelődik, ha `NUM1` kisebb vagy egyenlő, mint `NUM2`.

`$NUM1 -ge $NUM2`
: Kiértékelődik, ha `NUM1` nagyobb vagy egyenlő, mint `NUM2`.

`$NUM1 -eq $NUM2`
: Kiértékelődik, ha `NUM1` egyenlő `NUM2`-vel.

`$NUM1 -ne $NUM2`
: Kiértékelődik, ha `NUM1` nem egyenlő `NUM2`-vel.

Minden teszt a következő módosítókat kaphatja:

`! EXPR`
: Kiértékelődik, ha `EXPR` kifejezés hamis.

`EXPR1 -a EXPR2`
: Kiértékelődik, ha az `EXPR1` és az `EXPR2` kifejezés is igaz.

`EXPR1 -o EXPR2`
: Kiértékelődik, ha a két kifejezés közül legalább az egyik igaz.

Egy másik feltételes szerkezet, a `case`, az *if* egy változatának tekinthető. A `case` utasítás akkor hajtja végre a megadott parancsok listáját, ha egy megadott elem — például egy változó tartalma — megtalálható a *csővezetékekkel* (pipe) (a függőleges vonallal `|`) elválasztott és `)`-vel végződő elemek listájában. A következő mintascript azt mutatja be, hogy a `case` szerkezet hogyan használható egy adott Linux-disztribúció megfelelő szoftvercsomagolási formátumának megadására:

```
#!/bin/bash

DISTRO=$1

echo -n "Distribution $DISTRO uses "
case "$DISTRO" in
    debian | ubuntu | mint)
    echo -n "the DEB"
  ;;
    centos | fedora | opensuse )
    echo -n "the RPM"
  ;;
    *)
    echo -n "an unknown"
  ;;
esac
echo " package format."
```

A minták és a hozzájuk tartozó parancsok listáját a `;;`, `;&` vagy `;;&` jelekkel kell lezárni. Az utolsó minta, egy csillag, akkor fog megfelelni, ha előtte nem volt más megfelelő minta. Az `esac` utasítás (*case* visszafelé) lezárja a `case` szerkezetet. Feltételezve, hogy az előző mintascript neve `script.sh`, és az `opensuse` első argumentummal hajtjuk végre, a következő kimeneti eredményt kapjuk:

```
$ ./script.sh opensuse
Distribution opensuse uses the RPM package format.
```

TIP

A Bash rendelkezik egy `nocasematch` nevű opcióval, amely lehetővé teszi a case-insensitive mintaillesztést a `case` szerkezethez és más feltételes parancsokhoz. A `shopt` beépített parancs a shell opcionális viselkedését szabályozó beállítások értékeit váltogatja: a `shopt -s` engedélyezi (*set*) az adott opciót, a `shopt -u` pedig letiltja (*unset*) az adott opciót. Ezért a case szerkezet elé helyezett `shopt -s nocasematch` engedélyezi a case-insensitive mintaillesztést. A `shopt` által módosított opciók csak az aktuális munkamenetre lesznek hatással, így a sub-shellben futó scripteken belül módosított opciók — ami a scriptek futtatásának a szokásos módja — nem befolyásolják a szülő munkamenet opcióit.

A keresett elem és a mintázatok átesnek a tilde kiterjesztésen, a paraméterek kiterjesztésén, a parancsok helyettesítésén és az aritmetikai kiterjesztésen. Ha a keresett elemet idézőjelekkel adjuk meg, akkor azok eltávolításra kerülnek, mielőtt a megfeleltetés megtörténne.

Loop szerkezetek (ciklusok)

A scripteket gyakran használják ismétlődő feladatok automatizálásának eszközeként, ugyanannak a parancskészletnek a végrehajtásával, amíg egy leállítási kritérium nem teljesül. A Bash három ciklus utasítással rendelkezik - a `for`, az `until` és a `while` -, amelyek kissé eltérő cikluskonstrukciókhoz készültek.

A `for` szerkezet végigmegy egy adott elemlistán - általában szavak, vagy más szóközzel elválasztott szövegszegmensek listáján - és minden egyes elemen ugyanazt a parancskészletet hajtja végre. Minden egyes iteráció előtt a `for` utasítás az aktuális elemet hozzárendeli egy változóhoz, amelyet aztán a mellékelt parancsok használhatnak. A folyamat addig ismétlődik, amíg az elemek el nem fogynak. A `for` konstrukció szintaxisa a következő:

```
for VARNAME in LIST
do
    COMMANDS
done
```

A `VARNAME` egy tetszőleges shellváltozó neve, a `LIST` pedig egy tetszőleges szeparált kifejezésekből álló sorozat. A lista elemeit szeparáló érvényes elválasztó karaktereket (delimiter karakterek) az `IFS` környezeti változó határozza meg, amelyek alapértelmezés szerint a *szóköz*, *tab* és *újsor* karakterek. A végrehajtandó parancsok listáját a `do` és `done` utasítások határolják, így a parancsok annyi sort foglalhatnak el, amennyit csak szükséges.

A következő példában a `for` parancs a megadott listából minden egyes elemet - számok sorozatát - a `NUM` változóhoz rendeli, egyszerre egy-egy elemet:

```
#!/bin/bash

for NUM in 1 1 2 3 5 8 13
do
    echo -n "$NUM is "
    if [ $(( $NUM % 2 )) -ne 0 ]
    then
        echo "odd."
    else
        echo "even."
  fi
done
```

A példában egy beágyazott `if` szerkezetet használunk egy aritmetikai kifejezéssel együtt annak

kiértékelésére, hogy az aktuális `NUM` változóban lévő szám páros (even) vagy páratlan(odd). Feltételezve, hogy az előző példascript neve `script.sh`, és az aktuális mappában van, a következő kimenet keletkezik:

```
$ ./script.sh
1 is odd.
1 is odd.
2 is even.
3 is odd.
5 is odd.
8 is even.
13 is odd.
```

A Bash egy alternatív formátumot is támogat a `for` szerkezetkhez, mégpedig a dupla zárójeles jelölést. Ez a jelölés hasonlít a C programozási nyelv `for` utasítás szintaxisára, és különösen hasznos a tömbökkel való munkához:

```
#!/bin/bash

SEQ=( 1 1 2 3 5 8 13 )

for (( IDX = 0; IDX < ${#SEQ[*]}; IDX++ ))
do
    echo -n "${SEQ[$IDX]} is "
    if [ $(( ${SEQ[$IDX]} % 2 )) -ne 0 ]
    then
        echo "odd."
    else
        echo "even."
  fi
done
```

Ez a mintascript pontosan ugyanazt a kimenetet fogja generálni, mint az előző példa. Azonban ahelyett, hogy a `NUM` változót használná az egyes elemek tárolására, az `IDX` változót használja a tömb aktuális indexének növekvő sorrendben történő követésére, 0-tól kezdve és folyamatosan hozzáadva, amíg az a `SEQ` tömbben lévő elemek száma alatt van. Az aktuális elemet a tömbi pozíciójából a `${SEQ[$IDX]}` segítségével keressük ki.

Hasonló módon az `until` szerkezet addig hajt végre egy parancssorozatot, amíg egy tesztparancs — mint maga a `test` — 0 (sikeres) állapottal nem terminálódik. Például az előző példában szereplő ciklust az `until`-lal a következőképpen lehet megvalósítani:

```
#!/bin/bash

SEQ=( 1 1 2 3 5 8 13 )

IDX=0

until [ $IDX -eq ${#SEQ[*]} ]
do
    echo -n "${SEQ[$IDX]} is "
    if [ $(( ${SEQ[$IDX]} % 2 )) -ne 0 ]
    then
        echo "odd."
    else
        echo "even."
  fi
  IDX=$(( $IDX + 1 ))
done
```

Az `until` szerkezetek több utasítást igényelhetnek, mint a `for` szerkezetek, de alkalmasabbak lehetnek a `test` kifejezések vagy bármely más parancs által megadott nem számszerű megállási feltételekhez. Fontos, hogy olyan műveleteket tartalmazzon, amelyek biztosítják az érvényes leállási kritériumot, például egy számlálóváltozó inkrementálását, különben a ciklus a végtelenségig futhat.

A `while` utasítás hasonló az `until` utasításhoz, de a `while` akkor ismétli meg a parancsok sorát, ha a tesztparancs 0 (sikeres) állapottal zárul. Ezért az `until [ $IDX -eq ${#SEQ[*]} ]` utasítás az előző példából egyenértékű a `while [ $IDX -lt ${#SEQ[*]} ]` utasítással, mivel a ciklusnak addig kell ismétlődnie, amíg a tömb indexe *kisebb*, mint a tömbben lévő elemek száma.

Egy részletesebb példa

Képzeljük el, hogy egy felhasználó rendszeresen szinkronizálni szeretné fájljainak és mappáinak gyűjteményét egy másik tárolóeszközzel, amelyet a fájlrendszer egy tetszőleges csatlakoztatási pontjához csatoltak, egy teljes körű biztonsági mentési rendszer pedig túlzásnak tűnik. Mivel ezt a tevékenységet rendszeresen kell elvégezni, ezért ez egy jó alkalom arra, hogy egy shellscript segítségével automatizáljuk.

A feladat egyszerű: szinkronizáljunk minden listában szereplő fájlt és mappát az első script argumentumként megadott kiindulási mappából a második script argumentumként megadott célmappába. A lista elemeinek könnyebb hozzáadása vagy eltávolítása érdekében a listát egy különálló fájlban, a `~/.sync.list` fájlban tároljuk, soronként egy elemmel:

```
$ cat ~/.sync.list
Documents
To do
Work
Family Album
.config
.ssh
.bash_profile
.vimrc
```

A fájl fájlok és mappák keverékét tartalmazza, némelyiknek a nevében szóköz is van. Ez egy jó forgatókönyv a beépített `mapfile` Bash parancs számára, amely elemzi az adott szöveges tartalmat, és létrehoz belőle egy tömbváltozót, minden egyes sort külön tömbelemként elhelyezve. A script-fájl neve `sync.sh` lesz, és a következő scriptet tartalmazza:

```
#!/bin/bash

set -ef

# List of items to sync
FILE=~/.sync.list

# Origin directory
FROM=$1

# Destination directory
TO=$2

# Check if both directories are valid
if [ ! -d "$FROM" -o ! -d "$TO" ]
then
  echo Usage:
  echo "$0 <SOURCEDIR> <DESTDIR>"
  exit 1
fi

# Create array from file
mapfile -t LIST < $FILE

# Sync items
for (( IDX = 0; IDX < ${#LIST[*]}; IDX++ ))
do
```

```
    echo -e "$FROM/${LIST[$IDX]} \u2192 $TO/${LIST[$IDX]}";
    rsync -qa --delete "$FROM/${LIST[$IDX]}" "$TO";
done
```

A script első művelete az, hogy a `set` paranccsal átdefiniál két shellparamétert: az `-e` kapcsoló azonnal kilép a végrehajtásból, ha egy parancs nem nulla státusszal terminálódik, az `-f` pedig kikapcsolja a fájlnév-globbingot. Mindkét opciót `-ef`-ként lehet rövidíteni. Ez nem egy kötelező lépés, de segít csökkenteni a váratlan viselkedés valószínűségét.

A script-fájl tényleges alkalmazásorientált utasításai három részre oszthatók:

1. *script paraméterek összegyűjtése és ellenőrzése*

   A `FILE` változó a másolandó elemek listáját tartalmazó fájl elérési útvonala: `~/.sync.list`. A `FROM` és `TO` változók a kiindulási és a cél útvonalak. Mivel ezt a két paramétert a felhasználó adja meg, egy egyszerű validációra van szükség, amit az `if` szerkezet végez el: ha a kettő közül bármelyik nem érvényes mappa — a `[ ! -d "$FROM" -o ! -d "$TO" ]` teszt alapján --, a script egy rövid üzenetet jelenít meg és 1-es kilépési státusszal fejeződik be.

2. *Fájlok és mappák listájának betöltése*

   Miután minden paramétert definiáltunk, a `mapfile -t LIST < $FILE` paranccsal létrehozunk egy tömböt, amely a másolandó elemek listáját tartalmazza. A `mapfile -t` kapcsolója minden sorból eltávolítja az újsor karaktert, mielőtt a `LIST` tömbváltozóba felvenné a sorokat. A `FILE` változóban megadott fájl — `~/.sync.list` — tartalmát a bemeneti átirányítással olvassa be.

3. *Másolás végrehajtása és a felhasználó tájékoztatása*

   Egy `for` ciklus kettős zárójeles jelölést használva végigjárja az elemek tömbjét, az `IDX` változó pedig nyomon követi az index növekedését. A `echo` parancs tájékoztatja a felhasználót minden egyes másolt elemről. A kimeneti üzenetben jelen van a *jobb nyíl* karaktert jelképező, kivédett (escape) `\u2192` unicode karakter, ezért az `echo` parancs `-e` kapcsolóját kell használni. Az `rsync` parancs szelektíven csak a módosított fájldarabokat másolja az eredetiből, ezért ajánlott a használata az ilyen feladatokhoz. Az `rsync` `-q` és `-a` kapcsolók (összevonva `-qa`), meggátolják az `rsync` üzeneteket és aktiválják az *archive* módot, ahol minden fájl tulajdonsága megmarad. A `--delete` kapcsoló hatására az `rsync` törölni fog egy olyan elemet a célmappában, amely már nem létezik az eredetiben, ezért óvatosan kell használni.

Feltételezve, hogy a listában szereplő összes elem létezik a `carol` felhasználó home mappájában, a `/home/carol`-ban, és a `/media/carol/backup` célmappa egy csatlakoztatott külső tárolóeszközre mutat, a `sync.sh /home/carol /media/carol/backup` parancs a következő

kimenetet fogja generálni:

```
$ sync.sh /home/carol /media/carol/backup
/home/carol/Documents → /media/carol/backup/Documents
/home/carol/"To do" → /media/carol/backup/"To do"
/home/carol/Work → /media/carol/backup/Work
/home/carol/"Family Album" → /media/carol/backup/"Family Album"
/home/carol/.config → /media/carol/backup/.config
/home/carol/.ssh → /media/carol/backup/.ssh
/home/carol/.bash_profile → /media/carol/backup/.bash_profile
/home/carol/.vimrc → /media/carol/backup/.vimrc
```

A példa azt is feltételezi, hogy a scriptet a root vagy a `carol` felhasználó hajtja végre, mivel a legtöbb fájl más felhasználók számára olvashatatlan lenne. Ha a `script.sh` nem a `PATH` környezeti változóban szereplő mappában található, akkor a teljes elérési útvonalát kell megadnunk.

Gyakorló feladatok

1. Hogyan lehetne a `test` parancsot használni annak ellenőrzésére, hogy a `FROM` változóban tárolt fájl elérési útvonala újabb-e, mint egy olyan fájlé, amelynek elérési útvonalát a `TO` változó tárolja?

2. A következő scriptnek egy 0-tól 9-ig terjedő számsort kellene kiírnia, de ehelyett végtelen mennyiségű 0-t ír ki. Mit kell tennünk, hogy a várt kimenetet kapjuk?

```
#!/bin/bash

COUNTER=0

while [ $COUNTER -lt 10 ]
do
  echo $COUNTER
done
```

3. Tegyük fel, hogy egy felhasználó írt egy scriptet, amely a felhasználónevek rendezett listáját igényli. Az így kapott rendezett lista a következőképpen jelenik meg a számítógépén:

```
carol
Dave
emma
Frank
Grace
henry
```

   A kollégája számítógépén azonban ugyanez a lista a következőképpen van elrendezve:

```
Dave
Frank
Grace
carol
emma
henry
```

   Mi lehet a magyarázat a két rendezett lista közötti különbségekre?

Gondolkodtató feladatok

1. Hogyan lehetne a script összes parancssori argumentumát felhasználni egy Bash tömb inicializálására?

2. Miért van az, hogy a `test 1 > 2` parancs a megérzéseinkkel szemben igaznak értékelődik ki?

3. Hogyan változtathatná meg egy felhasználó ideiglenesen az alapértelmezett mezőelválasztót csak újsor karakterre, miközben továbbra is képes marad az eredeti tartalom visszaállítására?

Összefoglalás

Ez a lecke mélyebben megvizsgálja a `test` parancshoz elérhető teszteket, valamint más feltételes és ciklusszerkezeteket, amelyek a bonyolultabb shell scriptek írásához szükségesek. Egy egyszerű fájlszinkronizálási scriptet mutatunk be példaként a shell-scriptelésre. A lecke a következő lépéseket veszi át:

- Kibővített tesztek a `if` és `case` feltételes szerkezetekhez.
- Shell loop szerkezetek: `for`, `until` és `while`.
- Tömbökön és paramétereken való iterálás.

A bemutatott parancsok és eljárások a következők voltak:

`test`
: A parancshoz megadott elemek összehasonlítása.

`if`
: A parancsfájlokban használt logikai szerkezet, amely valamit igaznak vagy hamisnak értékel, majd az eredmények alapján elágazik a parancs végrehajtása.

`case`
: Több érték kiértékelése egyetlen változóval szemben. A scriptparancs végrehajtása ezután a `case` parancs eredményétől függően történik.

`for`
: Egy parancs végrehajtásának megismétlése egy adott kritérium alapján.

`until`
: Egy parancs végrehajtásának megismétlése, amíg egy kifejezés értéke hamis nem lesz.

`while`
: A parancs végrehajtásának megismétlése, amíg egy adott kifejezés igaznak értékelődik ki.

Válaszok a gyakorló feladatokra

1. Hogyan lehetne a `test` parancsot használni annak ellenőrzésére, hogy a `FROM` változóban tárolt fájl elérési útvonala újabb-e, mint egy olyan fájlé, amelynek elérési útvonalát a `TO` változó tárolja?

   A `test "$FROM" -nt "$TO"` parancs 0 státuszkódot ad vissza, ha a `FROM` változóban lévő fájl újabb, mint a `TO` változóban lévő fájl.

2. A következő scriptnek egy 0-tól 9-ig terjedő számsort kellene kiírnia, de ehelyett végtelen mennyiségű 0-t ír ki. Mit kell tennünk, hogy a várt kimenetet kapjuk?

   ```
   #!/bin/bash

   COUNTER=0

   while [ $COUNTER -lt 10 ]
   do
     echo $COUNTER
   done
   ```

   A `COUNTER` változó értékét növelni kell, ami egy aritmetikai kifejezéssel `COUNTER=$(( $COUNTER + 1 ))` oldható meg, hogy végül elérjük a megállási kritériumot és a ciklus véget érjen.

3. Tegyük fel, hogy egy felhasználó írt egy scriptet, amely a felhasználónevek rendezett listáját igényli. Az így kapott rendezett lista a következőképpen jelenik meg a számítógépén:

   ```
   carol
   Dave
   emma
   Frank
   Grace
   henry
   ```

   A kollégája számítógépén azonban ugyanez a lista a következőképpen van elrendezve:

   ```
   Dave
   Frank
   Grace
   carol
   ```

```
emma
henry
```

Mi lehet a magyarázat a két rendezett lista közötti különbségekre?

A rendezés az aktuális rendszerváltozaton alapul. Az ellentmondások elkerülése érdekében a rendezési feladatokat úgy kell elvégezni, hogy a `LANG` környezeti változó `C`-re van állítva.

Válaszok a gondolkodtató feladatokra

1. Hogyan lehetne a script összes parancssori argumentumát felhasználni egy Bash tömb inicializálására?

   A `PARAMS=( $* )` vagy `PARAMS=( "$@" )` parancsok létrehoznak egy `PARAMS` nevű tömböt, benne az összes argumentummal.

2. Miért van az, hogy a `test 1 > 2` parancs a megérzéseinkkel szemben igaznak értékelődik ki?

   A `>` operátor stringek tesztelésére van, nem számokéra.

3. Hogyan változtathatná meg egy felhasználó ideiglenesen az alapértelmezett mezőelválasztót csak újsor karakterre, miközben továbbra is képes marad az eredeti tartalom visszaállítására?

   Az `IFS` változó másolata egy másik változóban tárolható: `OLDIFS=$IFS`. Ezután az új sorválasztó az `IFS=$'\n'`-el definiálható, az IFS változó pedig az `IFS=$OLDIFS`-el állítható vissza.

Linux Professional Institute

Témakör 106: Felhasználói felületek és asztalok

106.1 Az X11 telepítése és konfigurálása

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 102, Objective 106.1

Súlyozás

2

Kulcsfontosságú ismeretek

- Az X11 architektúra megértése
- Az X Window konfigurációs fájl alapvető megértése és ismerete
- A Xorg konfiguráció bizonyos aspektusainak, például a billentyűzetkiosztásnak a felülírása
- Az asztali környezetek összetevőinek, például a kijelző- és ablakkezelőknek a megértése
- Az X-szerverhez való hozzáférés kezelése és alkalmazások megjelenítése távoli X-szervereken
- A Wayland ismerete

A használt fájlok, kifejezések és segédprogramok listája

- `/etc/X11/xorg.conf`
- `/etc/X11/xorg.conf.d/`
- `~/.xsession-errors`
- `xhost`
- `xauth`
- `DISPLAY`
- `X`

106.1 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	106 Felhasználói felületek és asztalok
Fejezet:	106.1 Az X11 telepítése és konfigurálása
Lecke:	1 of 1

Bevezetés

Az X Window System egy olyan szoftvercsomag, amely szöveg és grafika megjelenítésére szolgál a képernyőn. Az X-kliens általános megjelenését és kialakítását nem az X Window System határozza meg, hanem minden egyes X-kliens, egy *ablakkezelő* (window manager) (pl. Window Maker, Tab Window Manager) vagy egy teljes *asztal-környezet* (desktop environment), mint például a KDE, GNOME vagy Xfce. Az asztali környezetekkel egy későbbi leckében foglalkozunk. Ez a lecke az X Window System mögöttes architektúrájára és az X Window System általános eszközeire összpontosít, amelyeket egy rendszergazda az X konfigurálásához használhat.

Az X Window System platformokon ível át, és különböző operációs rendszereken fut, mint például a Linux, a BSD-k, a Solaris és más Unix-szerű rendszerek. Az Apple macOS és a Microsoft Windows számára is elérhetőek implementációk.

A modern Linux disztribúciókban használt X protokoll elsődleges verziója az *X.org* 11-es verziója, amelyet általában *X11*-ként láthatunk leírva. Az X-protokoll az X-kliens és az X-szerver közötti kommunikációs mechanizmus. Az X-kliens és az X-szerver közötti különbségről az alábbiakban lesz szó.

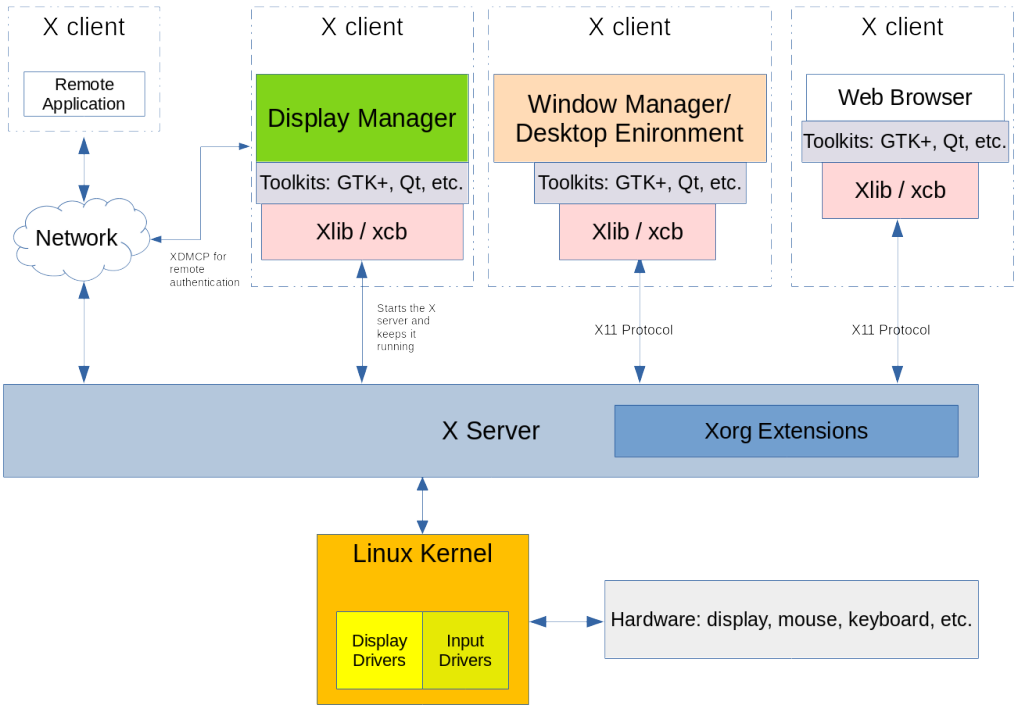
NOTE

Az X Window System elődje egy *W* nevű window system volt, amely az IBM, a DEC és az MIT közös fejlesztési munkája volt. Ez a szoftver az 1984-es *Project Athena* projekt keretében született. Amikor a fejlesztők egy új megjelenítő szerveren kezdtek dolgozni, az angol ábécé következő betűjét választották: “X”. Az X Window System fejlődését jelenleg a *MIT X Consortium* irányítja.

X Window System Architecture

Az X Window System biztosítja az alapvető kétdimenziós (és a kiterjesztések révén háromdimenziós) alakzatok kijelzőre rajzolásának mechanizmusait. A rendszer egy kliensre és egy szerverre oszlik, és a legtöbb olyan telepítés esetén, ahol grafikus asztalra van szükség, mindkét komponens ugyanazon a számítógépen van jelen. A kliens egy alkalmazás, például egy terminálemulátor, egy játék vagy egy webböngésző formájában jelenik meg. Minden kliensalkalmazás tájékoztatja az X-szervert az ablakának helyéről és méretéről a számítógép képernyőjén. A kliens kezeli azt is, hogy mi kerüljön az adott ablakba, az X-szerver pedig a kért ábrázolást rajzolja ki a képernyőre. Az X Window System kezeli az olyan eszközökről érkező bemenetet is, mint az egerek, billentyűzetek, trackpadek stb.

X Window System alapvető struktúrája



Az X Window System hálózaton is használható, és egy hálózat különböző számítógépeiről több X-

kliens is küldhet rajzkéréseket egyetlen távoli X-szervernek. Ennek oka az, hogy a rendszergazda vagy a felhasználó egy távoli rendszeren hozzáférhessen egy olyan grafikus alkalmazáshoz, amely a helyi rendszerén esetleg nem érhető el.

Az X Window System egyik legfontosabb jellemzője, hogy moduláris. Az X Window System fennállása során újabb és újabb funkciókat fejlesztettek ki és adtak hozzá a keretrendszerhez. Ezeket az új komponenseket csak az X-szerver bővítéseként adták hozzá, az X11 protokoll alapját érintetlenül hagyva. Ezek a kiterjesztések a *Xorg* könyvtárfájlokban találhatók. Példák az Xorg könyvtárakra: Az X-szervert kibővített funkciókkal ellátó `libXrandr`, `libXcursor`, `libX11`, `libxkbfile`, valamint még sok más.

A *display manager* (kijelzőkezelő) grafikus bejelentkezést biztosít a rendszerbe. Ez a rendszer lehet egy helyi vagy egy hálózaton keresztüli számítógép. A kijelzőkezelő a számítógép indítása után indul el, és elindít egy X-szerver munkamenetet a hitelesített felhasználó számára. A kijelzőkezelő felelős az X-szerver működésben tartásáért is. Példaként a GDM, az SDDM és a LightDM kijelzőkezelőket említhetjük.

A futó X-szerver minden egyes példánya rendelkezik egy *display name* azonosítóval, ez pedig az alábbiakat tartalmazza:

```
hostname:displaynumber.screennumber
```

A display name azt is megadja a grafikus alkalmazásnak, hogy hol és melyik hoston kell megjeleníteni (ha távoli X-kapcsolatot használ).

A `hostname` annak a rendszernek a nevére utal, amely az alkalmazást megjeleníti. Ha a hostnév hiányzik a megjelenítési névből, akkor azt feltételezi, hogy a lokális host az.

A `displaynumber` a használatban lévő “képernyők” gyűjteményére utal, legyen az egyetlen laptop képernyője vagy egy munkaállomáson lévő több képernyő. Minden futó X-szerver munkamenetnek van egy kijelzőszáma, amely a `0`-val kezdődik..

Az alapértelmezett `screennumber` a `0`. Ez akkor fordulhat elő, ha csak egy fizikai képernyő vagy több fizikai képernyő van úgy konfigurálva, hogy egy képernyőként működjön. Ha egy többmonitoros beállításban az összes képernyő egyetlen logikai képernyővé van összevonva, az alkalmazásablakok szabadon mozgathatók a képernyők között. Azokban a helyzetekben, amikor az egyes képernyők úgy vannak konfigurálva, hogy egymástól függetlenül működjenek, minden egyes képernyő a bennük megnyíló alkalmazásablakokat tartalmazza és az ablakok nem mozgathatók egyik képernyőről a másikra. Minden független képernyőhöz saját számot rendelnek. Ha csak egy logikai képernyő van használatban, akkor a pont és a képernyőszám elhagyható.

A futó X-munkamenet megjelenített nevét a `DISPLAY` környezeti változó tárolja:

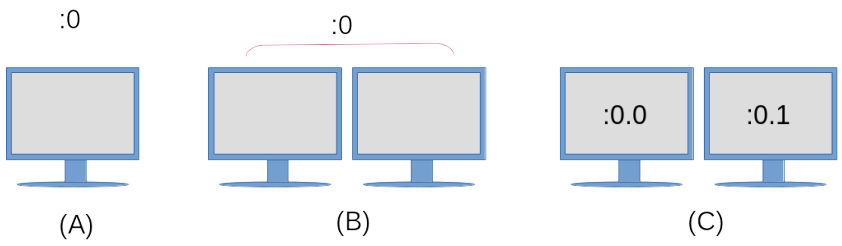
```
$ echo $DISPLAY
:0
```

A kimenet a következőket tartalmazza:

1. A használatban lévő X-szerver a helyi rendszeren van, ezért a kettőspont után balra nem jelenik meg semmi.
2. Az aktuális X-szerver munkamenet az első, amit a kettőspontot közvetlenül követő `0` jelez.
3. Csak egy logikai képernyő van használatban, ezért a képernyőszám nem látható.

A koncepció további ábrázolása a következő ábrán látható:

Példa a kijelző konfigurációkra



(A)

Egyetlen monitor, egyetlen kijelzőkonfigurációval és egyetlen képernyővel.

(B)

Egyetlen kijelzőként konfigurálva, két fizikai monitorral, amelyek egy képernyőként vannak konfigurálva. Az alkalmazásablakok szabadon mozgathatók a két monitor között.

(C)

Egyetlen kijelzőkonfiguráció (ahogyan azt a `:0` jelzi), azonban minden egyes monitor egy független képernyő. A két képernyő továbbra is ugyanazokat a beviteli eszközöket, például billentyűzetet és egeret használja, azonban a `:0.0` képernyőn megnyitott alkalmazás nem mozgatható át a `:0.1` képernyőre és fordítva.

Ha egy alkalmazást egy adott képernyőn szeretnénk elindítani, akkor az alkalmazás elindítása előtt adjuk meg a képernyő számát a `DISPLAY` környezeti változónak:

```
$ DISPLAY=:0.1 firefox &
```

Ez a parancs elindítja a Firefox webböngészőt a fenti ábrán jobbra látható képernyőn. Egyes eszköztárak parancssori opciókat is biztosítanak, amelyekkel utasíthatjuk az alkalmazást, hogy egy megadott képernyőn fusson. Lsd a `--screen` és `--display` parancsokat a `gtk-options(7)` man oldalon.

X-szerver konfiguráció

Hagyományosan az X-szerver konfigurálására használt elsődleges konfigurációs fájl az `/etc/X11/xorg.conf` fájl. A modern Linux disztribúciókban az X-szerver futás közben, az X-szerver indításakor konfigurálja magát, így előfordulhat, hogy nem létezik `xorg.conf` fájl.

Az `xorg.conf` fájl különálló szakaszokra van felosztva, amelyeket *szekcióknak* (sections) nevezünk. Minden szakasz a `Section` kifejezéssel kezdődik, majd ezt követően a *section name* következik, amely egy komponens konfigurációjára utal. Minden `Section` egy megfelelő `EndSection` szakasz végződéssel zárul. Egy tipikus `xorg.conf` fájl a következő szakaszokat tartalmazza:

`InputDevice`

Egy adott billentyűzet- vagy egérmodell konfigurálására szolgál.

`InputClass`

A modern Linux disztribúciókban ez a szakasz általában egy külön konfigurációs fájlban található, amely az `/etc/X11/xorg.conf.d/` alatt található. Az `InputClass` a hardvereszközök *osztályának*, például billentyűzeteknek és egereknek a konfigurálására szolgál, nem pedig egy adott hardverre. Az alábbiakban látható egy példa az `/etc/X11/xorg.conf.d/00-keyboard.conf` fájlra:

```
Section "InputClass"
        Identifier "system-keyboard"
        MatchIsKeyboard "on"
        Option "XkbLayout" "us"
        Option "XkbModel" "pc105"
EndSection
```

Az `XkbLayout` opció meghatározza a billentyűk elrendezését a billentyűzeten, például nyelv, Dvorak, bal- vagy jobbkezes és QWERTY. Az `XkbModel` opció a használt billentyűzet típusának meghatározására szolgál. A modellek, elrendezések és leírásaik táblázatát az `xkeyboard-`

`config(7)` beállításban találjuk. A billentyűzetkiosztásokhoz tartozó fájlok a `/usr/share/X11/xkb` állományban találhatók. Egy példa a görög Polytonic billentyűzetkiosztásra egy Chromebook számítógépen a következőképpen néz ki:

```
Section "InputClass"
        Identifier "system-keyboard"
        MatchIsKeyboard "on"
        Option "XkbLayout" "gr(polytonic)"
        Option "XkbModel" "chromebook"
EndSection
```

Alternatív megoldásként a billentyűzetkiosztás egy futó X munkamenet alatt is módosítható a `setxkbmap` paranccsal. Íme egy példa arra, ahogy ez a parancs a görög Polytonic elrendezést állítja be egy Chromebook számítógépen:

```
$ setxkbmap -model chromebook -layout "gr(polytonic)"
```

Ez a beállítás csak addig tart, amíg a használt X munkamenet. Ha véglegesíteni szeretnénk ezeket a változtatásokat, módosítsuk az `/etc/X11/xorg.conf.d/00-keyboard.conf` fájlt úgy, hogy tartalmazza a szükséges beállításokat.

NOTE A `setxkbmap` parancs az *X Keyboard Extension* (XKB) parancsot használja. Ez egy példa az X Window System kiegészítő funkcionalitására a kiterjesztések használata révén.

A modern Linux disztribúciók biztosítják a `localectl` parancsot a `systemd` segítségével, amely szintén használható a billentyűzetkiosztás módosítására, és automatikusan létrehozza az `/etc/X11/xorg.conf.d/00-keyboard.conf` konfigurációs fájlt. Íme még egy példa a görög Polytonic billentyűzet beállítására egy Chromebookon, ezúttal a `localectl` parancs segítségével:

```
$ localectl --no-convert set-x11-keymap "gr(polytonic)" chromebook
```

A `--no-convert` kapcsolót itt arra használjuk, hogy megakadályozzuk, hogy a `localectl` módosítsa a host konzol billentyűzettérképét.

`Monitor`

A `Monitor` szakasz leírja a használt fizikai monitort és annak csatlakozási helyét. Az alábbi példakonfigurációban egy hardveres monitor van csatlakoztatva a második kijelzőporthoz, és

elsődleges monitorként van használva.

```
Section "Monitor"
        Identifier  "DP2"
        Option      "Primary" "true"
EndSection
```

`Device`

A `Device` szakasz a használt fizikai videokártyát írja le. A szakasz tartalmazza a videokártya meghajtójaként használt kernelmodult és annak fizikai helyét az alaplapon.

```
Section "Device"
        Identifier  "Device0"
        Driver      "i915"
        BusID       "PCI:0:2:0"
EndSection
```

`Screen`

A `Screen` szakasz köti össze a `Monitor` és a `Device` szekciókat. Például egy `Screen` szakasz így nézhet ki:

```
Section "Screen"
        Identifier "Screen0"
        Device     "Device0"
        Monitor    "DP2"
EndSection
```

`ServerLayout`

A `ServerLayout` szakasz az összes szekciót, például az egeret, a billentyűzetet és a képernyőket egy X Window System interfészbe csoportosítja.

```
Section "ServerLayout"
    Identifier   "Layout-1"
    Screen       "Screen0" 0 0
    InputDevice  "mouse1"  "CorePointer"
    InputDevice  "system-keyboard"  "CoreKeyboard"
EndSection
```

NOTE Nem minden szakasz található meg egy konfigurációs fájlban. Azokban az

> esetekben, amikor egy szakasz hiányzik, az alapértelmezett értékeket a futó X-szerver példány adja meg.

A felhasználó által megadott konfigurációs fájlok szintén az `/etc/X11/xorg.conf.d/` állományban találhatók. A disztribúció által biztosított konfigurációs fájlok a `/usr/share/X11/xorg.conf.d/` állományban találhatók. Az `/etc/X11/xorg.conf.d/` állományban található konfigurációs fájlok elemzése a `/etc/X11/xorg.conf` állomány előtt történik, ha az elérhető a rendszerben.

Az `xdpyinfo` parancs egy számítógépen a futó X-szerver példányról szóló információk megjelenítésére szolgál. Az alábbiakban a parancs kimeneti mintája látható:

```
$ xdpyinfo
name of display:    :0
version number:    11.0
vendor string:    The X.Org Foundation
vendor release number:    12004000
X.Org version: 1.20.4
maximum request size:  16777212 bytes
motion buffer size:  256
bitmap unit, bit order, padding:    32, LSBFirst, 32
image byte order:    LSBFirst
number of supported pixmap formats:    7
supported pixmap formats:
    depth 1, bits_per_pixel 1, scanline_pad 32
    depth 4, bits_per_pixel 8, scanline_pad 32
    depth 8, bits_per_pixel 8, scanline_pad 32
    depth 15, bits_per_pixel 16, scanline_pad 32
    depth 16, bits_per_pixel 16, scanline_pad 32
    depth 24, bits_per_pixel 32, scanline_pad 32
    depth 32, bits_per_pixel 32, scanline_pad 32
keycode range:    minimum 8, maximum 255
focus:  None
number of extensions:    25
    BIG-REQUESTS
    Composite
    DAMAGE
    DOUBLE-BUFFER
    DRI3
    GLX
    Generic Event Extension
    MIT-SCREEN-SAVER
    MIT-SHM
```

```
    Present
    RANDR
    RECORD
    RENDER
    SECURITY
    SHAPE
    SYNC
    X-Resource
    XC-MISC
    XFIXES
    XFree86-VidModeExtension
    XINERAMA
    XInputExtension
    XKEYBOARD
    XTEST
    XVideo
default screen number:    0
number of screens:    1

screen #0:
  dimensions:    3840x1080 pixels (1016x286 millimeters)
  resolution:    96x96 dots per inch
  depths (7):    24, 1, 4, 8, 15, 16, 32
  root window id:    0x39e
  depth of root window:    24 planes
  number of colormaps:    minimum 1, maximum 1
  default colormap:    0x25
  default number of colormap cells:    256
  preallocated pixels:    black 0, white 16777215
  options:    backing-store WHEN MAPPED, save-unders NO
  largest cursor:    3840x1080
  current input event mask:    0xda0033
    KeyPressMask            KeyReleaseMask          EnterWindowMask
    LeaveWindowMask         StructureNotifyMask     SubstructureNotifyMask
    SubstructureRedirectMask PropertyChangeMask      ColormapChangeMask
  number of visuals:    270
...
```

A kimenet fontosabb részei félkövérrel vannak szedve, mint például a kijelző neve (ami megegyezik a `DISPLAY` környezeti változó tartalmával), a használt X-szerver verzióinformációja, a használt Xorg-bővítmények száma és listája, valamint további információk magáról a képernyőről.

Egy egyszerű Xorg konfigurációs fájl létrehozása

Annak ellenére, hogy az X a modern Linux telepítéseken a rendszer indítása után létrehozza a saját konfigurációját, az `xorg.conf` fájl továbbra is használható. Egy állandó `/etc/X11/xorg.conf` fájl létrehozásához futtassuk a következő parancsot:

```
$ sudo Xorg -configure
```

NOTE

Ha már fut egy X munkamenet, akkor a parancsban egy másik `DISPLAY`-t kell megadnunk, például:

```
$ sudo Xorg :1 -configure
```

Egyes Linux disztribúciókban az `X` parancs használható az `Xorg` helyett, mivel az `X` egy szimbolikus link az `Xorg`-ra.

Egy `xorg.conf.new` fájl jön létre az aktuális mappában. Ennek a fájlnak a tartalma abból származik, amit az X-szerver a helyi rendszer hardverében és illesztőprogramjaiban elérhetőként talált. A fájlt a használathoz át kell helyezni az `/etc/X11/` mappába, és át kell nevezni `xorg.conf` -ra:

```
$ sudo mv xorg.conf.new /etc/X11/xorg.conf
```

NOTE

Az alábbi manual oldalak további információkat nyújtanak az X Window System komponenseiről: `xorg.conf(5)`, `Xserver(1)`, `X(1)` és `Xorg(1)`.

Wayland

A Wayland az újabb megjelenítési protokoll, amelyet az X Window System leváltására terveztek. Sok modern Linux disztribúció használja alapértelmezett megjelenítő szerverként. Célja, hogy a rendszer erőforrásait kevésbé terhelje, és kisebb legyen a telepítési lábnyom, mint az X esetén. A projekt 2010-ben indult, és még mindig aktív fejlesztés alatt áll, beleértve az aktív és korábbi X.org fejlesztők munkáját is.

Az X Window Systemtől eltérően nincs szerverpéldány, amely a kliens és a kernel között fut. Ehelyett a kliensablak a saját kódjával vagy egy eszközkészlet (például a Gtk+ vagy a Qt) kódjával dolgozik a megjelenítésen. A rendereléshez a Wayland protokollon keresztül kérés érkezik a Linux kernelhez. A kernel a kérést a Wayland protokollon keresztül továbbítja a Wayland *compositor*-nak, amely az eszközbemenetet, az ablakkezelést és a kompozíciót kezeli. A

kompozitor a rendszer azon része, amely a renderelt elemeket a képernyőn megjelenő vizuális kimenetté egyesíti.

A legtöbb modern eszközkészlet, például a Gtk+ 3 és a Qt 5 is frissítésre került, hogy lehetővé tegye a renderelést akár X Window Systemet, akár Waylandet futtató számítógépen. Egyelőre még nem minden önálló alkalmazást írtak meg úgy, hogy támogassa a Waylandben történő renderelést. Az olyan alkalmazások és keretrendszerek esetében, amelyek még mindig az X Window System futtatását célozzák, az alkalmazás *XWayland*-en belül futtatható. Az XWayland rendszer egy különálló X-szerver, amely egy Wayland-kliensen belül fut, és így a kliensablak tartalmát egy önálló X-szerver példányon belül rendereli.

Ahogy az X Window System a `DISPLAY` környezeti változót használja a használt képernyők nyilvántartására, úgy a Wayland protokoll a `WAYLAND_DISPLAY` környezeti változóval teszi ugyanezt. Az alábbiakban egy Wayland kijelzőt futtató rendszer kimeneti mintája látható:

```
$ echo $WAYLAND_DISPLAY
wayland-0
```

Ez a környezeti változó nem érhető el az X-et futtató rendszereken.

Gyakorló feladatok

1. Milyen parancsot használnhatnánk annak meghatározására, hogy milyen Xorg-bővítmények állnak rendelkezésre a rendszeren?

2. Elkezdtünk használni egy vadonatúj 10 gombos egeret a számítógéphez, azonban további konfigurációra van szükség ahhoz, hogy az összes gomb megfelelően működjön. Az X-szerver többi konfigurációjának módosítása nélkül melyik mappában hoznánk létre egy új konfigurációs fájlt ehhez az egérhez, és milyen konkrét konfigurációs szakaszokat használnánk ebben a fájlban?

3. A Linux telepítés melyik komponense felelős az X-szerver működésének fenntartásáért?

4. Milyen parancssori kapcsolót használunk az `X` paranccsal egy új `xorg.conf` konfigurációs fájl létrehozásához?

Gondolkodtató feladatok

1. Mi lenne a `DISPLAY` környezeti változó tartalma egy `lab01` nevű rendszeren, amely egyetlen kijelző konfigurációt használ? Tegyük fel, hogy a `DISPLAY` környezeti változót egy terminál emulátorban a harmadik független képernyőn nézzük!

2. Milyen parancs segítségével hozható létre billentyűzetkonfigurációs fájl az X Window System számára?

3. Egy tipikus Linux telepítésnél a felhasználó a billentyűzet `Ctrl` + `Alt` + `F1`-`F6` billentyűk megnyomásával válthat virtuális terminálra. Felkértek minket egy grafikus felülettel rendelkező kioszk rendszer telepítésére, ahol ezt a funkciót le kell tiltani, hogy megakadályozzuk a rendszer illetéktelen manipulálását. Úgy döntünk, hogy létrehozunk egy `/etc/X11/xorg.conf.d/10-kiosk.conf` konfigurációs fájlt. A `ServerFlags` szakaszban (amely a szerver globális Xorg-opciók beállítására szolgál), milyen kapcsolót kell megadni? Az `xorg(1)` man oldalon megtalálhatjuk a választ.

Összefoglalás

Ez a lecke a Linuxon használt X Window Systemet mutatja be. Az X Window System a különböző konfigurációs fájlokban meghatározott képek és szövegek képernyőre rajzolására szolgál. Az X Window Systemet gyakran használják a beviteli eszközök, például egerek és billentyűzetek konfigurálására. Ez a lecke a következő pontokat foglalta össze:

- Az X Window System architektúrája magas szinten.
- Milyen konfigurációs fájlokat használhatunk az X Window System konfigurálásához, és ezek hol vannak a fájlrendszerben.
- Hogyan használjuk a `DISPLAY` környezeti változót egy X rendszert futtató rendszeren.
- A Wayland megjelenítési protokoll rövid bemutatása.

A tárgyalt parancsok és konfigurációs fájlok a következők voltak:

- A billentyűzetkiosztás módosítása egy Xorg telepítésen belül a `setxkbmap` és a `localectl` segítségével.
- Az `Xorg` parancs egy új `/etc/X11/xorg.conf` konfigurációs fájl létrehozására.
- Az Xorg konfigurációs fájlok tartalma a következő helyen található: `/etc/X11/xorg.conf`, `/etc/X11/xorg.conf.d/` és `/usr/share/X11/xorg.conf.d/`.
- Az `xdpyinfo` parancs a futó X-szerver munkamenetre vonatkozó általános információk megjelenítésére.

Válaszok a gyakorló feladatokra

1. Milyen parancsot használnhatnánk annak meghatározására, hogy milyen Xorg-bővítmények állnak rendelkezésre a rendszeren?

   ```
   $ xdpyinfo
   ```

2. Elkezdtünk használni egy vadonatúj 10 gombos egeret a számítógéphez, azonban további konfigurációra van szükség ahhoz, hogy az összes gomb megfelelően működjön. Az X-szerver többi konfigurációjának módosítása nélkül melyik mappában hoznánk létre egy új konfigurációs fájlt ehhez az egérhez, és milyen konkrét konfigurációs szakaszokat használnánk ebben a fájlban?

   A felhasználó által definiált konfigurációkat az `/etc/X11/xorg.conf.d/` állományban kell elhelyezni, és az egér konfigurációjához szükséges speciális szakasz az `InputDevice`.

3. A Linux telepítés melyik komponense felelős az X-szerver működésének fenntartásáért?

   A display manager.

4. Milyen parancssori kapcsolót használunk az `X` paranccsal egy új `xorg.conf` konfigurációs fájl létrehozásához?

   ```
   -configure
   ```

   Ne feledjük, hogy az `X` parancs az `Xorg` parancsra mutató szimbolikus link.

Válaszok a gondolkodtató feladatokra

1. Mi lenne a `DISPLAY` környezeti változó tartalma egy `lab01` nevű rendszeren, amely egyetlen kijelző konfigurációt használ? Tegyük fel, hogy a `DISPLAY` környezeti változót egy terminál emulátorban a harmadik független képernyőn nézzük!

```
$ echo $DISPLAY
lab01:0.2
```

2. Milyen parancs segítségével hozható létre billentyűzetkonfigurációs fájl az X Window System számára?

```
$ localectl
```

3. Egy tipikus Linux telepítésnél a felhasználó a billentyűzet `Ctrl` + `Alt` + `F1`-`F6` billentyűk megnyomásával válthat virtuális terminálra. Felkértek minket egy grafikus felülettel rendelkező kioszk rendszer telepítésére, ahol ezt a funkciót le kell tiltani, hogy megakadályozzuk a rendszer illetéktelen manipulálását. Úgy döntünk, hogy létrehozunk egy `/etc/X11/xorg.conf.d/10-kiosk.conf` konfigurációs fájlt. A `ServerFlags` szakaszban (amely a szerver globális Xorg-opciók beállítására szolgál), milyen kapcsolót kell megadni? Az `xorg(1)` man oldalon megtalálhatjuk a választ.

```
Section "ServerFlags"
    Option  "DontVTSwitch"  "True"
EndSection
```

106.2 Grafikus asztali környezetek

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 102, Objective 106.2

Súlyozás

1

Kulcsfontosságú ismeretek

- A főbb asztali környezetek ismerete
- A távoli asztali munkamenetek eléréséhez szükséges protokollok ismerete

A használt fájlok, kifejezések és segédprogramok listája

- KDE
- Gnome
- Xfce
- X11
- XDMCP
- VNC
- Spice
- RDP

Linux Professional Institute

106.2 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	106 Felhasználói felületek és asztalok
Fejezet:	106.2 Grafikus asztali környezetek
Lecke:	1/1

Bevezetés

A Linux-alapú operációs rendszerek fejlett parancssori felületükről ismertek, de ez a nem-technikai felhasználók számára ijesztő lehet. A számítógép-használat intuitívabbá tételének szándékával a nagy felbontású kijelzők és a mutatóeszközök kombinációja életre hívta a képvezérelt felhasználói felületeket. Míg a parancssori felület előzetes ismereteket igényel a programnevekről és azok beállítási lehetőségeiről, addig a *grafikus felhasználói felületen* (graphical user interface - GUI) a programfunkciókat az ismerős vizuális elemekre való mutatással lehet elindítani, így a tanulási görbe kevésbé meredek. Továbbá a grafikus felület a legalkalmasabb multimédiás és egyéb vizuálisan orientált tevékenységekhez.

A grafikus felhasználói felület szinte szinonimája a számítógépes felületnek, és a legtöbb Linux-disztribúció alapértelmezésben telepített grafikus felülettel érkezik. Nincs azonban egyetlen monolitikus program sem, amely felelős lenne a Linux rendszerekben elérhető teljes értékű grafikus asztalokért. Ehelyett minden grafikus asztal valójában programok és függőségeik nagy gyűjteménye, amelyek a disztribúció választásától függően a felhasználó személyes ízlésének megfelelően változnak.

X Window System

A Linuxban és más Unix-szerű operációs rendszerekben, ahol használják, az *X Window System* (más néven *X11* vagy csak *X*) biztosítja a grafikus felület megjelenítéséhez és a felhasználói interakcióhoz kapcsolódó alacsony szintű erőforrásokat, mint például:

- A beviteli események kezelése, mint például az egérmozgások vagy billentyűleütések.
- A szöveges tartalom kivágásának, másolásának és beillesztésének képessége különálló alkalmazások között.
- A programozási felület, amelyet más programok a grafikus elemek rajzolásához használnak.

Bár az X Window System felelős a grafikus megjelenítés vezérléséért (maga a video-driver is az X része), nem célja, hogy önmagában összetett vizuális elemeket rajzoljon. Az alakzatokat, színeket, árnyalatokat és minden más vizuális hatást az X tetején futó alkalmazás generálja. Ez a megközelítés nagy teret ad az alkalmazásoknak a testre szabott felületek létrehozására, ugyanakkor az alkalmazás hatókörén túlmutató fejlesztési többletköltségekhez, valamint más programfelületekhez képest következetlenségekhez vezethet a megjelenés és a viselkedés terén.

A fejlesztők szemszögéből nézve az *asztali környezet* (desktop environment) bevezetése megkönnyíti az alapul szolgáló alkalmazásfejlesztéshez kötött GUI-programozást, míg a felhasználó szemszögéből nézve a különböző alkalmazások között egységes élményt nyújt. Az asztali környezetek olyan programozási felületeket, könyvtárakat és támogató programokat egyesítenek, amelyek együttműködnek a hagyományos, de még mindig fejlődő tervezési koncepciók megvalósítása érdekében.

Asztali környezet

A hagyományos asztali számítógép grafikus felhasználói felülete különböző ablakokból áll — az *ablak* kifejezés itt bármely autonóm képernyőterületre utal --, amelyek a futó folyamatokhoz kapcsolódnak. Mivel az X Window System önmagában csak alapvető interaktív funkciókat kínál, a teljes felhasználói élmény az asztali környezet által biztosított komponensektől függ.

Az asztali környezet talán legfontosabb összetevője, a *window manager* (ablakkezelő) az ablakok elhelyezését és dizájnját szabályozza. Az ablakkezelő adja hozzá az ablakhoz a címsorokat, a vezérlőgombokat — általában a kicsinyítés, a nagyítás és a bezárás műveletekhez kapcsolódóan — és kezeli a nyitott ablakok közötti váltást.

NOTE

Az asztali számítógépek grafikus felületeinek alapkoncepciói a tényleges irodai munkaterületekről vett ötletekből származnak. Metaforikusan szólva a számítógép képernyője az íróasztal, ahol olyan objektumok, mint a dokumentumok és mappák

találhatók. A dokumentum tartalmát tartalmazó alkalmazásablak olyan fizikai műveleteket utánoz, mint egy űrlap kitöltése vagy egy kép megfestése. Mint a valódi íróasztalok, a számítógépes asztalok is rendelkeznek szoftveres kiegészítőkkel, például jegyzettömbökkel, órákkal, naptárakkal stb., amelyek többsége a "valódi" megfelelőik alapján készült.

Minden asztali környezet biztosít egy ablakkezelőt, amely megfelel a *widget toolkit* megjelenésének. A widgetek az alkalmazásablakon belül elosztott informatív vagy interaktív vizuális elemek, például gombok vagy szövegbeviteli mezők. A szabványos asztali komponensek — mint az alkalmazásindító, a feladatsor stb. — és maga az ablakkezelő is ilyen widget eszközkészletekre támaszkodik a felületük összeállításához.

Az olyan szoftverkönyvtárak, mint a *GTK+* és a *Qt*, olyan widgeteket biztosítanak, amelyek segítségével a programozók bonyolult grafikus felületeket készíthetnek alkalmazásaikhoz. Korábban a GTK+ segítségével fejlesztett alkalmazások nem úgy néztek ki, mint a Qt-vel készült alkalmazások, és fordítva, de a mai asztali környezetek jobb téma-támogatása miatt a különbség kevésbé nyilvánvaló.

Összességében a GTK+ és a Qt ugyanazokat a funkciókat kínálja a widgetek tekintetében. Az egyszerű interaktív elemek megkülönböztethetetlenek lehetnek, míg az összetett widgetek — például az alkalmazások által fájlok megnyitásához vagy mentéséhez használt párbeszédablak — azonban egészen másképp nézhetnek ki. Mindazonáltal a különböző eszközkészletekkel épített alkalmazások egyszerre is futtathatók, függetlenül attól, hogy a többi asztali komponens milyen widgeteszközkészletet használ.

Az alapvető asztali komponensek mellett, amelyek önmagukban is önálló programoknak tekinthetők, az asztali környezetek az asztali metaforát azáltal követik, hogy az azonos tervezési irányelvek szerint kifejlesztett kiegészítő alkalmazások minimális készletét biztosítják. A következő alkalmazások variációit általában minden nagyobb asztali környezet biztosítja:

Rendszerrel kapcsolatos alkalmazások

A rendszerhez kapcsolódó alkalmazások: terminál emulátor, fájlkezelő, csomagtelepítő menedzser, rendszerkonfigurációs eszközök.

Kommunikáció és Internet

Kapcsolatkezelő, email kliens, webböngésző.

Irodai alkalmazások

Naptár, számológép, szövegszerkesztő.

Az asztali környezetek számos más szolgáltatást és alkalmazást is tartalmazhatnak: a

bejelentkezési képernyő, a munkamenet-kezelő, a folyamatok közötti kommunikáció, keyring agent stb. Emellett harmadik féltől származó rendszerszolgáltatások által biztosított funkciókat is tartalmaznak, mint például a *PulseAudio* a hangkezeléshez és a *CUPS* a nyomtatáshoz. Ezeknek a funkcióknak nincs szükségük a grafikus környezetre a működéshez, de az asztali környezet grafikus frontendeket biztosít az ilyen erőforrások konfigurálásának és működtetésének megkönnyítésére.

Népszerű asztali környezetek

Sok szabadalmaztatott operációs rendszer csak egyetlen hivatalos asztali környezetet támogat, amely az adott kiadáshoz van kötve, és nem szabad megváltoztatni. Velük ellentétben a Linux-alapú operációs rendszerek különböző asztali környezeteket támogatnak, amelyek az X-szel együtt használhatók. Mindegyik asztali környezetnek megvannak a saját jellemzői, de általában van néhány közös tervezési koncepciójuk:

- Egy alkalmazásindító, amely felsorolja a rendszerben elérhető beépített és harmadik féltől származó alkalmazásokat.
- A fájltípusokhoz és protokollokhoz tartozó alapértelmezett alkalmazásokat meghatározó szabályok.
- Konfigurációs eszközök az asztali környezet megjelenésének és viselkedésének testre szabásához.

A *Gnome* az egyik legnépszerűbb asztali környezet, az első választás olyan disztribúciókban, mint a Fedora, Debian, Ubuntu, SUSE Linux Enterprise, Red Hat Enterprise Linux, CentOS stb. A Gnome a 3. verziójában jelentős változásokat hozott a megjelenésében és a struktúrájában, eltávolodott az asztali metaforától, és bevezette a *Gnome Shell*-t, mint új interfészt.

Figure 1. Gnome Shell tevékenységek

Az általános célú, teljes képernyős indítóprogram *Gnome Shell Activities* felváltotta a hagyományos alkalmazásindítót és a tálcát. A Gnome 3 azonban továbbra is használható a régi megjelenéssel, ha a bejelentkezési képernyőn a *Gnome Classic* opciót választjuk.

A *KDE* alkalmazások és fejlesztési platformok nagy ökoszisztémája. Legújabb asztali környezetének legújabb verzióját, a *KDE Plasma*-t alapértelmezés szerint az openSUSE, a Mageia, a Kubuntu stb. használja. A Qt könyvtár alkalmazása a KDE feltűnő jellemzője, amely összetéveszthetetlen megjelenést és eredeti alkalmazások sokaságát adja. A KDE még egy konfigurációs eszközt is biztosít a GTK+ alkalmazásokkal való vizuális kohézió biztosítására.

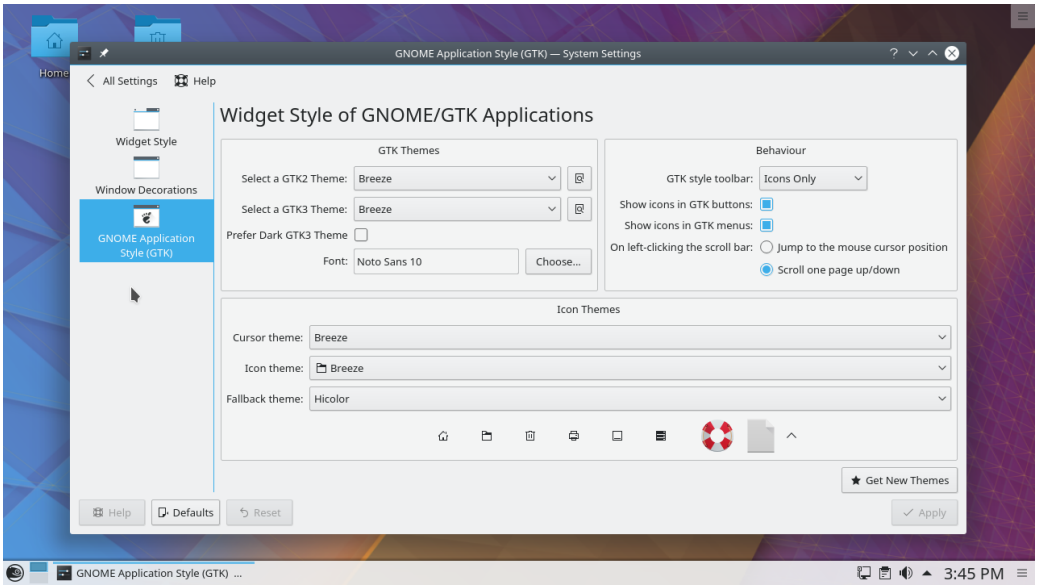


Figure 2. KDE GTK beállítások

Az *Xfce* egy olyan asztali környezet, amelynek célja, hogy esztétikus legyen, miközben nem fogyaszt sok gépi erőforrást. Szerkezete erősen moduláris, lehetővé téve a felhasználó számára, hogy igényei és preferenciái szerint aktiváljon és deaktiváljon komponenseket.

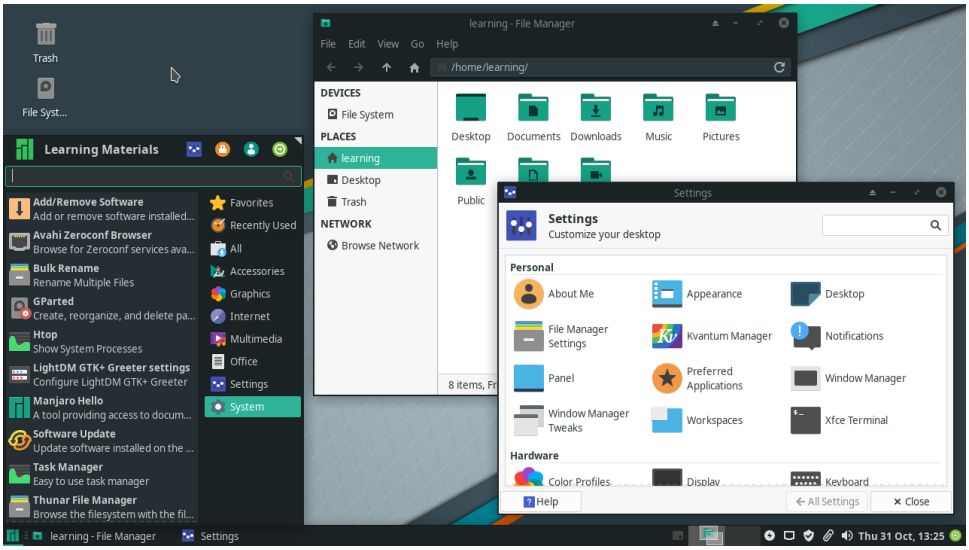


Figure 3. Az Xfce asztal

Sok más, Linuxhoz elérhető asztali környezet is létezik, amelyeket általában alternatív disztribúciók biztosítanak. A Linux Mint disztribúció például két eredeti asztali környezetet biztosít: *Cinnamon* (a Gnome 3 leágazása) és *MATE* (a Gnome 2 leágazása). Az *LXDE* egy alacsony erőforrás-fogyasztásra szabott asztali környezet, ami jó választás régebbi vagy single board számítógépek esetén. Bár nem kínálja a nehezebb asztali környezetek összes funkcióját, az LXDE minden alapvető, egy modern grafikus felhasználói felülettől elvárt funkciót biztosít.

TIP

A billentyűparancsok fontos szerepet játszanak az asztali környezettel való interakcióban. Egyes billentyűparancsok — például a `Alt` + `Tab` az ablakok közötti váltáshoz vagy a `Ctrl` + `c` a szöveg másolásához — univerzálisak az asztali környezetekben, de minden asztali környezetnek megvannak a saját billentyűparancsai is. A billentyűparancsok az asztali környezet által biztosított billentyűzetkonfigurációs eszközben találhatók, ahol a kombinációk hozzáadhatók vagy módosíthatók.

Asztali interoperabilitás

A Linux-alapú operációs rendszerekben az asztali környezetek sokfélesége kihívást jelent: hogyan lehet elérni, hogy ezek megfelelően működjenek harmadik féltől származó grafikus alkalmazásokkal vagy rendszerszolgáltatásokkal anélkül, hogy mindegyikhez külön támogatást kelljen nyújtani. Az asztali környezetek közötti közös módszerek és specifikációk nagyban javítják a felhasználói élményt, és számos fejlesztési kérdést megoldanak, mivel a grafikus

alkalmazásoknak kölcsönhatásba kell lépniük az aktuális asztali környezettel, függetlenül attól, hogy eredetileg milyen asztali környezetre tervezték őket. Emellett fontos az általános asztali beállítások megtartása, ha a felhasználó végül megváltoztatja az asztali környezetet.

A *freedesktop.org* szervezet számos, az asztali számítógépek közötti átjárhatóságra vonatkozó specifikációt tart fenn. A teljes specifikáció elfogadása nem kötelező, de sok közülük széles körben használatos:

Mappa lokációk

A személyes beállítások és egyéb felhasználó-specifikus fájlok helye.

Asztali bejegyzések

A parancssoros alkalmazások bármely terminálemulátoron keresztül futtathatók az asztali környezetben, de túl zavaró lenne, ha az összeset elérhetővé tennénk az alkalmazásindítóban. Az asztali bejegyzések `.desktop` végződésű szöveges fájlok, amelyeket az asztali környezet arra használ, hogy információt gyűjtsön az elérhető asztali alkalmazásokról és azok használatáról.

Alkalmazás automatikus indítása

Asztali bejegyzések, amelyek azt az alkalmazást jelzik, amelynek automatikusan el kell indulnia a felhasználó bejelentkezése után.

Drag and drop

Hogyan kezeljék az alkalmazások a drag and drop eseményeket.

Lomtár

A fájlkezelő által törölt fájlok közös helye, valamint a fájlok tárolásának és eltávolításának módszere.

Ikon témák

A cserélhető ikonkönyvtárak közös formátuma.

Az asztali környezetek által biztosított könnyű használatnak van egy hátránya az olyan szöveges felületekkel szemben, mint a shell: a távoli hozzáférés biztosítása. Míg egy távoli gép parancssoros shell környezetét könnyen elérhetjük olyan eszközökkel, mint az `ssh`, a grafikus környezetek távoli elérése más módszereket igényel, és lassabb kapcsolatokon nem biztos, hogy kielégítő teljesítményt nyújt.

Nem-helyi hozzáférés

Az X Window Systems az autonóm *megjelenítőkön* (display) alapuló felépítést alkalmazza, ahol

ugyanaz az *X display manager* egyszerre több grafikus asztali munkamenetet is vezérelhet. Lényegében a kijelző analóg a szöveges terminállal: mindkettő egy gépre vagy szoftveralkalmazásra utal, amelyet belépési pontként használnak egy független operációs rendszer munkamenet létrehozásához. Bár a leggyakoribb beállítás a helyi gépen futó egyetlen grafikus munkamenet, más, kevésbé hagyományos beállítások is lehetségesek:

- Váltás az aktív grafikus asztali munkamenetek között ugyanazon a gépen.
- Egynél több megjelenítő eszköz (pl. képernyő, billentyűzet, egér) csatlakoztatása ugyanahhoz a géphez, mindegyik a saját grafikus asztali munkamenetét vezérli.
- Távoli grafikus asztali munkamenetek, ahol a grafikus felületet a hálózaton keresztül egy távoli kijelzőre küldik.

A távoli asztali munkameneteket az X natívan támogatja, amely a távoli kijelzőkkel való kommunikációhoz az *X Display Manager Control Protocol* (XDMCP) protokollt használja. A nagy sávszélesség-használat miatt az XDMCP-t ritkán használják az interneten keresztül vagy alacsony sebességű LAN-okban. Az XDMCP-vel kapcsolatban biztonsági problémák is felmerülnek: a helyi kijelző egy privilegizált távoli X kijelzőkezelővel kommunikál a távoli eljárások végrehajtásához, így egy esetleges sebezhetőség lehetővé teheti tetszőleges privilegizált parancsok végrehajtását a távoli gépen.

Továbbá az XDMCP-hez a kapcsolat mindkét végén futó X-példányokra van szükség, ami megvalósíthatatlanná teheti, ha az X Windows rendszer nem áll rendelkezésre az összes érintett gépen. A gyakorlatban más, hatékonyabb és kevésbé invazív módszereket használnak a távoli grafikus asztali munkamenetek létrehozására.

A *Virtual Network Computing* (VNC) egy platformfüggetlen eszköz a távoli asztali környezetek megtekintésére és vezérlésére a *Remote Frame Buffer* protokoll (RFB) segítségével. Ezen keresztül a helyi billentyűzet és egér által létrehozott események továbbításra kerülnek a távoli asztali gépre, amely viszont visszaküldi a helyben megjelenítendő képernyőfrissítéseket. Lehetőség van több VNC-szerver futtatására ugyanazon a gépen, de minden VNC-szervernek szüksége van egy kizárólagos TCP-portra a bejövő munkamenet-kérelmeket fogadó hálózati interfészen. A konvenció szerint az első VNC-szervernek az 5900-as TCP-portot kell használnia, a másodiknak az 5901-est, és így tovább.

A VNC-szerver futtatásához nincs szükség különleges jogosultságokra. Egy közönséges felhasználó például bejelentkezhet a távoli fiókjába, és onnan elindíthatja a saját VNC-szerverét. Ezután a helyi gépen bármilyen VNC kliensalkalmazás használható a távoli asztal elérésére (feltéve, hogy a megfelelő hálózati portok elérhetők). A `~/.vnc/xstartup` fájl egy shellscript, amelyet a VNC-szerver indításkor futtat, és amellyel meghatározható, hogy a VNC-szerver milyen asztali környezetet tegyen elérhetővé a VNC-kliens számára. Fontos megjegyezni, hogy a VNC natívan

nem biztosít modern titkosítási és hitelesítési módszereket, ezért olyan harmadik féltől származó alkalmazással együtt kell használni, amely biztosítja ezeket a funkciókat. A VPN és SSH-tunneleket (alagutakat) tartalmazó módszereket gyakran használják a VNC-kapcsolatok biztosítására.

A *Remote Desktop Protocol* (RDP) elsősorban a *Microsoft Windows* operációs rendszer asztali gépének távoli elérésére szolgál a TCP 3389 hálózati porton keresztül. Bár a Microsoft saját fejlesztésű RDP protokollját használja, a Linux rendszerekben használt kliensimplementáció nyílt forráskódú program, amely a *GNU General Public License* (GPL) licenc alatt áll, és használatának nincs jogi korlátozása.

A *Simple Protocol for Independent Computing Environments* (Spice) egy olyan eszközkészletet tartalmaz, amelynek célja a *virtualizált* rendszerek asztali környezetének elérése, akár a helyi gépen, akár egy távoli helyen. Ezen túlmenően a Spice protokoll natív funkciókat kínál a helyi és a távoli rendszerek integrálásához, mint a helyi eszközök (például a hangszórók és a csatlakoztatott USB-eszközök) elérésének lehetőségét a távoli gépről, valamint a két rendszer közötti fájlmegosztást.

Mindegyik távoli asztali protokollhoz külön kliensparancsok állnak rendelkezésre a csatlakozáshoz, de a *Remmina* távoli asztali kliens integrált grafikus felületet biztosít, amely megkönnyíti a csatlakozási folyamatot, és opcionálisan tárolja a csatlakozási beállításokat későbbi használatra. A Remmina minden egyes protokollhoz rendelkezik bővítményekkel, és vannak bővítmények az XDMCP, a VNC, az RDP és a Spice protokollokhoz. A megfelelő eszköz kiválasztása az érintett operációs rendszerektől, a hálózati kapcsolat minőségétől és attól függ, hogy a távoli asztali környezetnek milyen funkciókkal kell rendelkeznie.

Gyakorló feladatok

1. Milyen típusú alkalmazás biztosít ablakos shell munkameneteket az asztali környezetben?

2. A Linux asztali környezetek sokfélesége miatt ugyanannak az alkalmazásnak több változata is lehet, amelyek mindegyike egy adott widget eszközkészlethez a legjobban illeszkedik. Például a *Transmission* bittorrent kliensnek két változata van: *transmission-gtk* és *transmission-qt*. Melyiket kell telepíteni a kettő közül a KDE-vel való maximális integráció érdekében?

3. Milyen Linux asztali környezetek ajánlottak alacsony költségű, kis feldolgozási teljesítményű single board számítógépekhez?

 | |

Gondolkodtató feladatok

1. Az X Window Systemben kétféleképpen lehet szöveget másolni és beilleszteni: a hagyományos `Ctrl` + `c` és `Ctrl` + `v` billentyűkombinációval (az ablak menüben is elérhető), vagy az egér középső gombjának kattintásával beilleszteni az éppen kijelölt szöveget. Mi a megfelelő módja a szöveg másolásának és beillesztésének terminálemulátorból?

2. A legtöbb asztali környezet a `Alt` + `F2` billentyűparancsot a *Run program* ablakhoz rendeli, ahol a programok parancssori módon futtathatók. A KDE-ben melyik parancs futtatná az alapértelmezett terminálemulátort?

3. Milyen protokoll a legalkalmasabb egy távoli Windows-asztal elérésére Linux asztali környezetből?

Összefoglalás

Ez a lecke egy áttekintés a Linux rendszereken elérhető grafikus asztalokról. Az X Window System önmagában csak egyszerű felületi funkciókat biztosít, ezért az asztali környezetek a grafikus ablakos felületen bővítik a felhasználói élményt. A lecke a következő témákat járja körül:

- Grafikus interfész és X Window System koncepciók.
- Linuxhoz elérhető asztali környezetek.
- Asztali környezetek közötti hasonlóságok és különbségek.
- Hogyan érhetünk el egy távoli asztali környezetet.

A tárgyalt koncepciók és programok az alábbiak voltak:

- X Window System.
- Népszerű asztali környezetek: KDE, Gnome, Xfce.
- Távoli elérési protokollok: XDMCP, VNC, RDP, Spice.

Válaszok a gyakorló feladatokra

1. Milyen típusú alkalmazás biztosít ablakos shell munkameneteket az asztali környezetben?

   Bármely terminál emulátor, mint a Konsole, Gnome terminál, xterm, stb., hozzáférést biztosít egy helyi interaktív shell munkamenethez.

2. A Linux asztali környezetek sokfélesége miatt ugyanannak az alkalmazásnak több változata is lehet, amelyek mindegyike egy adott widget eszközkészlethez a legjobban illeszkedik. Például a *Transmission* bittorrent kliensnek két változata van: *transmission-gtk* és *transmission-qt*. Melyiket kell telepíteni a kettő közül a KDE-vel való maximális integráció érdekében?

   A KDE a Qt könyvtárra épült, így a Qt verziót — *transmission-qt* — kell telepítenünk.

3. Milyen Linux asztali környezetek ajánlottak alacsony költségű, kis feldolgozási teljesítményű single board számítógépekhez?

   Olyan alap asztali környezetek, amelyek nem használnak túl sok vizuális effektet, mint az Xfce és az LXDE.

Válaszok a gondolkodtató feladatokra

1. Az X Window Systemben kétféleképpen lehet szöveget másolni és beilleszteni: a hagyományos `Ctrl` + `c` és `Ctrl` + `v` billentyűkombinációval (az ablak menüben is elérhető), vagy az egér középső gombjának kattintásával beilleszteni az éppen kijelölt szöveget. Mi a megfelelő módja a szöveg másolásának és beillesztésének terminálemulátorból?

   Az interaktív shell munkamenetek a `Ctrl` + `c` billentyűparancsot rendelik a programfuttatás leállításához, ezért a középső gombos módszer használata javasolt.

2. A legtöbb asztali környezet a `Alt` + `F2` billentyűparancsot a *Run program* ablakhoz rendeli, ahol a programok parancssori módon futtathatók. A KDE-ben melyik parancs hajtaná végre az alapértelmezett terminálemulátort?

   A `konsole` parancs futtatja a KDE terminál emulátorát, de az olyan általános kifejezések, mint a *terminal* is működnek.

3. Milyen protokoll a legalkalmasabb egy távoli Windows-asztal elérésére egy Linux asztali környezetből?

   A Remote Desktop Protocol (RDP), mivel azt natívan támogatja mind a Windows, mind a Linux.

106.3 Akadálymentesség

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 102, Objective 106.3

Súlyozás

1

Kulcsfontosságú ismeretek

- Alapvető ismeretek a vizuális beállításokról és témákról
- Alapvető ismeretek a segédtechnológiákról

A használt fájlok, kifejezések és segédprogramok listája

- Nagy kontrasztú/nagyméretű asztali témák
- Képernyőolvasó
- Braille-kijelző
- Képernyőnagyító
- Képernyőn megjelenő billentyűzet
- Sticky/repeat billentyűk
- Slow/bounce/toggle billentyűk
- Egérgombok
- Gesztusok
- Hangfelismerés

106.3 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	106 Felhasználói felületek és asztalok
Fejezet:	106.3 Akadálymentesség
Lecke:	1/1

Bevezetés

A Linux asztali környezete számos beállítással és eszközzel rendelkezik a felhasználói felület fogyatékkal élők számára történő átalakításához. Az általános emberi interfész eszközök — képernyő, billentyűzet és egér/érintőpad — egyénileg átkonfigurálhatók a látáskárosodásból vagy a mozgáskorlátozottságból fakadó akadályok leküzdésére.

Lehetőség van például az asztali színséma beállítására, hogy jobban használható legyen a színvak emberek számára. Az ismétlődő megterhelés okozta sérülésben szenvedők is kihasználhatják az alternatív gépelési és mutatási módszerek előnyeit.

Az akadálymentességi funkciók egy részét maga az asztali környezet biztosítja, mint például a Gnome vagy a KDE, másik részét pedig kiegészítő programok. Ez utóbbi esetben fontos, hogy olyan eszközt válasszunk, amelyik a legjobban integrálódik az asztali környezetbe, így a segítség jobb minőségű lesz.

Akadálymentesség beállításai

Minden nagyobb Linux-disztribúció nagyjából ugyanazokat az akadálymentességi funkciókat

biztosítja, amelyeket az asztali környezethez tartozó beállításkezelőben található konfigurációs modullal lehet testre szabni. Az akadálymentesítési beállítási modul neve a Gnome esetén *Universal Access*, míg a KDE-ben a *System Settings*, *Personalization*, *Accessibility* alatt található. Más asztali környezetek, mint például az *Xfce*, szintén *Accessibility*-nek hívják a grafikus beállításkezelőjükben. Ezek azonban a Gnome-hoz és a KDE-hez képest csökkentett funkcionalitást kínálnak.

A Gnome például beállítható úgy, hogy a képernyő jobb felső sarkában állandóan megjelenjen az Universal Access menü, ahol gyorsan bekapcsolhatók az akadálymentesítési lehetőségek. A hangjelzés például vizuálisra cserélhető, így a hallássérült felhasználók könnyebben érzékelik a rendszer figyelmeztetéseit. Bár a KDE-ben nincs hasonló gyorselérési menü, a vizuális figyelmeztetés funkció is elérhető, de *visual bell* néven.

A billentyűzet és az egér támogatása

Az alapértelmezett billentyűzet és egér viselkedése módosítható a speciális mobilitási nehézségek kiküszöbölése érdekében. A billentyűkombinációk, a billentyűk automatikus ismétlési aránya és a nem szándékos billentyűleütések jelentős akadályokat jelenthetnek a csökkent mozgásképességű felhasználók számára. Ezeket a gépelési nehézségeket három, a billentyűzethez kapcsolódó akadálymentesítési funkció kezeli: *Sticky keys*, *Bounce keys* és *Slow keys*.

A Gnome Universal Access konfigurációjának *Typing Assist* szakaszában található *Sticky keys* funkció lehetővé teszi a felhasználó számára, hogy egyszerre egy billentyűvel írja be a billentyűparancsokat. Ha engedélyezve van, az olyan billentyűkombinációkat, mint a `Ctrl` + `C` nem kell egyszerre lenyomva tartani. A felhasználó először megnyomhatja a `Ctrl` billentyűt, majd elengedheti, és csak ezután nyomhatja meg a `C` billentyűt. A KDE-ben ez a beállítás az Accessibility settings ablak *Modifier Keys* lapján található. A KDE a *Locking Keys* opciót is felkínálja: ha engedélyezve van, a `Alt`, `Ctrl` és a Shift billentyűk “lent” maradnak, ha a felhasználó kétszer megnyomja őket, hasonlóan a Caps Lock billentyű viselkedéséhez. A Caps Lock funkcióhoz hasonlóan a felhasználónak újra meg kell nyomnia a megfelelő billentyűt a feloldáshoz.

A *Bounce keys* funkció úgy próbálja megakadályozni a nem szándékos billentyűleütéseket, hogy késleltetést helyez közéjük, vagyis az új billentyűleütést csak akkor fogadja el, ha az utolsó billentyűleütés óta eltelt egy meghatározott idő. A kézremegéssel küzdő felhasználók hasznosnak találhatják a Bounce keys funkciót, mert így elkerülhetik egy billentyű többszöri lenyomását akkor, amikor csak egyszer akarják megnyomni. A Gnome-ban ez a funkció csak az azonos billentyűk ismétlésére vonatkozik, míg a KDE-ben bármilyen más billentyű lenyomására is, és a *Keyboard Filters* fülön található meg.

A *Slow keys* funkció szintén segít elkerülni a véletlen billentyűleütéseket. Ha engedélyezve van, a felhasználónak egy meghatározott ideig nyomva kell tartania a billentyűt, mielőtt az elfogadásra

kerülne. A felhasználó igényeitől függően hasznos lehet a billentyűzet beállításaiban elérhető automatikus ismétlés beállítása a billentyű lenyomva tartása közben.

A Sticky keys és a Slow keys akadálymentességi funkciók be- és kikapcsolhatók a billentyűzeten végrehajtott *aktiváló mozdulatokkal* (Activation Gestures). A KDE-ben az *Use gestures for activating sticky keys and slow keys* opciót kell bejelölni az Activation Gestures engedélyezéséhez, míg a Gnome-ban ez a funkció a *Enable by Keyboard* nevet viseli a *Typing Assist* konfigurációs ablakban. Ha az Activation Gestures engedélyezve van, a Sticky keys funkció a Shift billentyű öt egymást követő alkalommal történő lenyomása után aktiválódik. A Slow keys funkció aktiválásához a Shift billentyűt nyolc másodpercig folyamatosan lenyomva kell tartani.

Azok a felhasználók, akiknek kényelmesebb a billentyűzet használata az egér vagy az érintőpad helyett, az asztali környezetben való eligazodáshoz a billentyűzet gyorsbillentyűit használhatják. Továbbá a *Mouse Keys* nevű funkció lehetővé teszi a felhasználó számára, hogy magát az egérmutatót a numerikus billentyűzettel vezérelje, amely teljes méretű billentyűzeteken és a nagyobb laptopokon is megtalálható.

A numerikus billentyűzet négyzetrácsba van rendezve, így minden szám egy iránynak felel meg: `2` lefelé mozgatja a kurzort, `4` balra mozgatja a kurzort, `7` észak-nyugatra mozgatja a kurzort, stb. Alapértelmezés szerint a `5` szám a bal egérkattintásnak felel meg.

Míg a Gnome-ban csak egy kapcsoló van a Mouse Keys opció engedélyezésére az Universal Access ablakban, addig a KDE-ben a Mouse Keys beállításai a *System Settings*, *Mouse*, *Keyboard Navigation* alatt találhatók, és az olyan opciók, mint a sebesség és a gyorsítás testre szabhatók.

TIP

A Slow keys, Stick keys, Bounce keys és Mouse keys az AccessX, az X Window System X billentyűzetbővítményének egyik erőforrása által biztosított akadálymentességi funkciók. Az AccessX beállításai a parancssorból is módosíthatók az `xkbset` paranccsal.

Az egér vagy az érintőpad használható billentyűbevitel generálására, ha a billentyűzet használata túl kényelmetlen vagy nem lehetséges. Ha a Gnome Universal Access beállításaiban a *Screen Keyboard* kapcsoló be van kapcsolva, akkor egy képernyőbillentyűzet jelenik meg minden alkalommal, amikor a kurzor egy szövegmezőben van, és az új szöveg bevitele az egérrel vagy az érintőképernyővel történő billentyűzetre kattintással történik, hasonlóan az okostelefonok virtuális billentyűzetéhez.

A KDE és más asztali környezetek alapértelmezés szerint nem biztosítják a képernyőbillentyűzetet, de az *onboard* csomag manuálisan telepíthető, ami egy egyszerű képernyőbillentyűzetet biztosít, amely bármelyik asztali környezetben használható. Telepítés után az alkalmazásindítóban normál alkalmazásként lesz elérhető.

A mutató viselkedése akkor is módosítható, ha az egérrel való kattintás és húzás fájdalmat okoz, vagy más okból nem kivitelezhető. Ha a felhasználó nem képes elég gyorsan kattintani az egérgombra ahhoz, hogy például dupla kattintás eseményt váltson ki, akkor a rendszer konfigurációs ablakában a *Mouse Preferences* alatt növelhető az az időintervallum, amíg az egérgombot másodszor is meg kell nyomni a dupla kattintáshoz.

Ha a felhasználó nem képes megnyomni az egér egyik gombját vagy egyiket sem, akkor az egérkattintás különböző technikákkal szimulálható. A *Gnome Universal Access Click Assist* szakaszában a *Simulate a right mouse click* opció jobb egérkattintást generál, ha a felhasználó lenyomja és lenyomva tartja a bal egérgombot. A *Simulate click by hovering* opció engedélyezésével kattintási eseményt vált ki, ha a felhasználó az egeret mozdulatlanul tartja. A KDE-ben a *KMouseTool* alkalmazás ugyanezeket a funkciókat biztosítja az egérrel végzett műveletek segítésére.

Látássérültek

A csökkent látású felhasználók továbbra is használhatják a monitor képernyőjét a számítógéppel való interakcióra. A felhasználó igényeitől függően számos vizuális módosítás végezhető el a szabványos grafikus asztal egyébként nehezen látható részleteinek javítása érdekében.

A Gnome *Universal Access* beállítások *Seeing* szekciója olyan opciókat biztosít, amelyek segíthetnek a csökkent látással élőknek:

High Contrast
: az ablakok és gombok könnyebben láthatóvá válnak, mivel élesebb színekkel jeleníti meg őket.

Large Text
: megnöveli a képernyő szabványos betűméretét.

Cursor Size
: lehetővé teszi a nagyobb egérkurzor kiválasztását, így az könnyebben megtalálható a képernyőn.

Ezen beállítások némelyike nem kapcsolódik szorosan az akadálymentességi funkciókhoz, ezért más asztali környezetek által biztosított konfigurációs segédprogramok megjelenés szakaszában találhatóak. Egy olyan felhasználó, akinek nehézséget okoz a vizuális elemek közötti megkülönböztetés, választhat egy nagy kontrasztú témát, hogy könnyebben felismerje a gombokat, az egymást átfedő ablakokat stb.

Ha a megjelenési beállítások önmagukban nem elegendőek a képernyő olvashatóságának javításához, akkor egy képernyőnagyító programmal a képernyő egyes részeinek nagyítása is

lehetséges. Ezt a funkciót a Gnome *Universal Access* beállításaiban *Zoom*-nak hívják, ahol olyan opciókat lehet testre szabni, mint a nagyítás aránya, a nagyító pozíciója és a színbeállítások.

A KDE-ben a *KMagnifier* program ugyanezeket a funkciókat biztosítja, de az alkalmazásindítón keresztül közönséges alkalmazásként érhető el. Más asztali környezetek saját képernyőnagyítót is biztosíthatnak. Az Xfce például a `Alt` billentyű lenyomva tartása közben az egérgörgő forgatásával nagyít ki és be a képernyőbe.

Végül, azok a felhasználók, akik számára a grafikus felület nem jelent lehetőséget, a számítógéppel való interakcióhoz használhatnak egy *képernyőolvasót*. A választott asztali környezettől függetlenül a Linux rendszerek legnépszerűbb képernyőolvasója az *Orca*, amely a legtöbb disztribúcióban alapértelmezés szerint telepítve van. Az Orca szintetizált hangot generál a képernyő eseményeinek jelentésére és az egérkurzor alatti szöveg felolvasására. Az Orca a *frissíthető Braille-kijelzőkkel* is működik, olyan speciális eszközökkel, amelyek a Braille-karaktereket az ujjbegyekkel tapintható kis tűk felemelésével jelenítik meg. Nem minden asztali alkalmazás igazodik teljes mértékben a képernyőolvasókhoz, és nem minden felhasználó számára lesz könnyű a használatuk, ezért fontos, hogy minél több képernyőolvasási stratégiát biztosítsunk a felhasználók számára, amelyek közül választhatnak.

Gyakorló feladatok

1. Milyen akadálymentességi funkció segíthet a felhasználónak váltogatni a megnyitott ablakok között a billentyűzet segítségével, ha a felhasználó nem tudja egyszerre lenyomni a `Alt` és a `Tab` billentyűket?

2. Hogyan segíthet a *Bounce keys* akadálymentességi funkció azoknak a felhasználóknak, akiknél a kézremegés megzavarja a gépelést?

3. Melyik a leggyakoribb Activation Gestures a *Sticky keys* akadálymentességi funkció aktiválásához?

Gondolkodtató feladatok

1. Az akadálymentességi funkciókat nem feltétlenül egyetlen alkalmazás biztosítja, és ezek egyes asztali környezetekben eltérőek lehetnek. A KDE-ben melyik alkalmazás segíti az ismétlődő megterheléses sérülésekkel küzdőket azzal, hogy mindig akkor kattint az egérrel, amikor az egérkurzor rövid szünetet tart?

2. A grafikus környezet milyen megjelenési szempontjai módosíthatók annak érdekében, hogy az emberek könnyebben el tudják olvasni a képernyőn megjelenő szöveget?

3. Milyen módon segíthet az *Orca* alkalmazás a látássérült felhasználóknak az asztali környezettel való interakcióban?

Összefoglalás

Ez a lecke a Linux rendszerekben elérhető általános akadálymentességi funkciókat tárgyalja. Minden nagyobb asztali környezet, különösen a Gnome és a KDE, számos beépített és harmadik féltől származó alkalmazást biztosít a látássérült vagy mozgáskorlátozott emberek számára. A lecke a következő témákat járja körül:

- Az akadálymentességi beállítások módosítása.
- A billentyűzet és az egér használatának alternatív módjai.
- Asztali fejlesztések látássérültek számára.

A tárgyalt parancsok és eljárások az alábbiak voltak:

- A billentyűzet akadálymentessé tétele: Sticky keys, Slow keys, Bounce keys.
- Egérrel kapcsolatos események mesterséges generálása.
- Képernyőn megjelenő billentyűzet.
- Vizuális beállítások az olvashatóság javítása érdekében.
- Nagy kontrasztú/nagy betűméretes asztali témák.
- Képernyőnagyítók.
- Az Orca képernyőolvasó.

Válaszok a gyakorló feladatokra

1. Milyen akadálymentességi funkció segíthet a felhasználónak váltogatni a megnyitott ablakok között a billentyűzet segítségével, ha a felhasználó nem tudja egyszerre lenyomni a `Alt` és a `Tab` billentyűket?

   A Sticky keys funkció, amely lehetővé teszi a felhasználó számára, hogy egyszerre egy billentyű lenyomásával írja be a billentyűparancsokat.

2. Hogyan segíthet a *Bounce keys* akadálymentességi funkció azoknak a felhasználóknak, akiknél a kézremegés megzavarja a gépelést?

   A Bounce keys engedélyezésével az új billentyű lenyomása csak az utolsó billentyű lenyomása óta eltelt meghatározott idő elteltével fogadható el.

3. Melyik a leggyakoribb Activation Gestures a *Sticky keys* akadálymentességi funkció aktiválásához?

   Ha az Activation Gestures engedélyezve van, a Sticky keys funkció a `Shift` billentyű ötszöri lenyomása után aktiválódik.

Válaszok a gondolkodtató feladatokra

1. Az akadálymentességi funkciókat nem feltétlenül egyetlen alkalmazás biztosítja, és ezek egyes asztali környezetekben eltérőek lehetnek. A KDE-ben melyik alkalmazás segíti az ismétlődő megterheléses sérülésekkel küzdőket azzal, hogy mindig akkor kattint az egérrel, amikor az egérkurzor rövid szünetet tart?

   A *KMouseTool* alkalmazás.

2. A grafikus környezet milyen megjelenési szempontjai módosíthatók annak érdekében, hogy az emberek könnyebben el tudják olvasni a képernyőn megjelenő szöveget?

   A képernyő nagyobb betűméretének beállítása az asztali konfigurációban megkönnyíti a képernyőn megjelenő szövegek olvashatóságát.

3. Milyen módon segíthet az *Orca* alkalmazás a látássérült felhasználóknak az asztali környezettel való interakcióban?

   Az Orca egy képernyőfelolvasó, amely szintetizált hangot generál a képernyő eseményeinek jelentéséhez és az egérkurzor alatti szöveg olvasásához. Működik a *frissíthető Braille-kijelző* nevű eszközökkel is, így a felhasználó tapintható mintákkal azonosíthatja a szöveget.

Linux Professional Institute

Témakör 107: Adminisztrációs feladatok

107.1 Felhasználói- és csoport-fiókok, valamint a kapcsolódó rendszerfájlok kezelése

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 102, Objective 107.1

Súlyozás

5

Kulcsfontosságú ismeretek

- Felhasználók és csoportok hozzáadása, módosítása és eltávolítása
- Felhasználói/csoportinformációk kezelése jelszó/csoport adatbázisokban
- Speciális célú és korlátozott fiókok létrehozása és kezelése

A használt fájlok, kifejezések és segédprogramok listája

- `/etc/passwd`
- `/etc/shadow`
- `/etc/group`
- `/etc/skel/`
- `chage`
- `getent`
- `groupadd`
- `groupdel`
- `groupmod`
- `passwd`

- `useradd`
- `userdel`
- `usermod`

107.1 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	107 Adminisztrációs feladatok
Fejezet:	107.1 Felhasználói- és csoport-fiókok, valamint a kapcsolódó rendszerfájlok kezelése
Lecke:	1/2

Bevezetés

A felhasználók és csoportok kezelése nagyon fontos része minden rendszergazda munkájának. A modern Linux-disztribúciók olyan grafikus felületeket implementálnak, amelyek lehetővé teszik az ezzel a kulcsfontosságú szemponttal kapcsolatos összes tevékenység gyors és egyszerű kezelését. Ezek a felületek a grafikus elrendezés tekintetében különböznek egymástól, de a funkciók ugyanazok. Ezekkel az eszközökkel megtekinthetjük, szerkeszthetjük, hozzáadhatjuk és törölhetjük a helyi felhasználókat és csoportokat. A fejlettebb munkavégzéshez azonban a parancssort kell használnunk.

Felhasználói fiókok hozzáadása

Linuxban új felhasználói fiókot a `useradd` paranccsal hozhatunk létre. Root jogosultságok birtokában például létrehozhatunk egy új felhasználói fiókot `michael` néven, alapértelmezett beállításokkal, a következő módon:

```
# useradd michael
```

A `useradd` parancs futtatásakor a jelszó- és csoportadatbázisokban tárolt felhasználói és csoportinformációk frissülnek az újonnan létrehozott felhasználói fiókkal, és ha megadtuk, az új felhasználó home mappája is létrejön. Az új felhasználói fiókkal azonos nevű csoport is létrejön.

Miután létrehoztuk az új felhasználót, a `passwd` paranccsal beállíthatjuk a jelszavát. A felhasználói azonosítóját (UID), a csoport azonosítóját (GID) és a hozzá tartozó csoportokat az `id` és a `groups` parancsok segítségével tekinthetjük meg.

```
# passwd michael
Changing password for user michael.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
# id michael
uid=1000(michael) gid=100(michael) groups=100(michael)
# groups michael
michael : michael
```

NOTE

Ne feledjük, hogy bármelyik felhasználó megtekintheti az UID-jét, GID-jét és a csoportokat, amelyekhez tartozik, ha egyszerűen, argumentumok nélkül használja az `id` és a `groups` parancsokat! Bármelyik felhasználó megváltoztathatja jelszavát a `passwd` paranccsal, azonban csak a root jogosultsággal rendelkező felhasználók változtathatják meg *bármely* felhasználó jelszavát.

Az `useradd` parancs legfontosabb kapcsolói a következők

`-c`

Új felhasználói fiók létrehozása egyedi megjegyzésekkel (például a felhasználó teljes neve).

`-d`

Új felhasználói fiók létrehozása egyedi home mappával.

`-e`

Új felhasználói fiók létrehozása egy adott dátum megadásával, amelyen a fiók letiltásra kerül.

`-f`

Új felhasználói fiók létrehozása a jelszó lejárta utáni napok számának megadásával, amely alatt a felhasználónak frissítenie kell a jelszavát (különben a fiók letiltásra kerül).

`-g`

Új felhasználói fiók létrehozása egy adott GID-del.

`-G`
Új felhasználói fiók létrehozása több másodlagos csoporthoz való hozzáadással.

`-k`
Új felhasználói fiók létrehozása a skeleton fájlok másolásával egy adott egyéni mappából (ez a kapcsoló csak akkor érvényes, ha a `-m` vagy a `--create-home` opciót adtuk meg).

`-m`
Új felhasználói fiók létrehozása a saját home mappájával (ha az nem létezik).

`-M`
Új felhasználói fiók létrehozása home mappa nélkül.

`-s`
Új felhasználói fiók létrehozása egy adott login shellel.

`-u`
Új felhasználói fiók létrehozása egy adott UID-val.

Az `useradd` parancs man oldalán a kapcsolók teljes listáját megtalálhatjuk.

Felhasználói fiókok módosítása

Időnként meg kell változtatnunk egy meglévő felhasználói fiók valamely attribútumát, például a bejelentkezési nevet, a login shellt, a jelszó lejárati dátumát stb. Ilyen esetekben a `usermod` parancsot kell használni.

```
# usermod -s /bin/tcsh michael
# usermod -c "Michael User Account" michael
```

A `useradd` parancshoz hasonlóan a `usermod` parancshoz is root jogosultságok szükségesek.

A fenti példákban először a `michael` felhasználó login shelljét változtattuk meg, majd egy rövid leírást adtunk a felhasználói fiókhoz. Ne feledjük, hogy egyszerre több attribútumot is módosíthatunk, egyetlen parancsban megadva őket!

A `usermod` parancs legfontosabb kapcsolói az alábbiak:

`-c`
Hozzáad egy rövid megjegyzést a megadott felhasználói fiókhoz.

`-d`

A megadott felhasználói fiók home mappájának módosítása. Az `-m` opcióval együtt használva az aktuális home mappa tartalma átkerül az új home mappába, amely, ha még nem létezik, létre is jön.

`-e`

A megadott felhasználói fiók lejárati dátumának beállítása.

`-f`

Beállítja a jelszó lejárta utáni napok számát, amely alatt a felhasználónak frissítenie kell a jelszót (különben a fiók letiltásra kerül).

`-g`

A megadott felhasználói fiók elsődleges csoportjának módosítása (a csoportnak léteznie kell).

`-G`

Másodlagos csoportok hozzáadása a megadott felhasználói fiókhoz. Minden csoportnak léteznie kell és vesszővel kell elválasztani őket egymástól, szóközök nélkül. Ha egyedül használjuk, ez a kapcsoló eltávolítja az összes meglévő csoportot, amelyhez a felhasználó tartozik, míg ha az `-a` opcióval együtt használjuk, akkor egyszerűen új másodlagos csoportokat csatol a meglévőkhöz.

`-l`

A megadott felhasználói fiók bejelentkezési nevének módosítása.

`-L`

A megadott felhasználói fiók zárolása. Ez egy felkiáltójelet tesz az `/etc/shadow` fájlban a titkosított jelszó elé, így letiltja az adott felhasználó jelszóval való hozzáférését.

`-s`

A megadott felhasználói fiók login shelljének módosítása.

`-u`

A megadott felhasználói fiók UID-jének módosítása.

`-U`

A megadott felhasználói fiók feloldása. Ez eltávolítja a felkiáltójelet a titkosított jelszó előtt a `/etc/shadow` fájlban.

Az opciók teljes listája a `usermod` parancs manual oldalain megtalálható.

TIP

Ne feledjük, hogy amikor megváltoztatjuk egy felhasználói fiók bejelentkezési nevét, valószínűleg át kell neveznünk a felhasználó home mappáját és más, a felhasználóval kapcsolatos elemeket, például a levelezési spoolfájlokat! Ne feledjük azt sem, hogy amikor megváltoztatjuk egy felhasználói fiók UID azonosítóját, valószínűleg a felhasználó home mappáján kívüli fájlok és mappák tulajdonjogát is meg kell változtatni (a felhasználói azonosító automatikusan megváltozik a felhasználó postafiókja és a felhasználó tulajdonában lévő, a felhasználó home mappájában található összes fájl esetén)!

Felhasználói fiókok törlése

Ha törölni szeretnénk egy felhasználói fiókot, használhatjuk a `userdel` parancsot. Ez a parancs frissíti a fiókadatbázisokban tárolt információkat, és törli a megadott felhasználóra utaló összes bejegyzést. A `-r` kapcsolóval a felhasználó home mappáját és annak minden tartalmát is eltávolítjuk, a felhasználó mail spooljával együtt. A máshol található egyéb fájlokat manuálisan kell megkeresni és törölni.

```
# userdel -r michael
```

Ahogy a `useradd` és a `usermod` esetén, a felhasználói fiókok törléséhez is root jogosultság szükséges.

Csoportok létrehozása, módosítása és törlése

A felhasználók menedzseléséhez hasonlóan hozhatunk létre, módosíthatunk vagy törölhetünk csoportokat a `groupadd`, `groupmod` és `groupdel` parancsokkal, ha van root jogosultságunk. Ha szeretnénk egy `developer` nevű csoportot létrehozni, az alábbi parancsot kell futtatnunk:

```
# groupadd -g 1090 developer
```

A parancs `-g` kapcsolója a csoportot egy specifikus GID-del hozza létre.

WARNING

Ne feledjük, hogy amikor új felhasználói fiókot adunk hozzá, az elsődleges csoportnak és a másodlagos csoportoknak, amelyekhez tartozik, léteznie kell a `useradd` parancs elindítása előtt.

Később, ha át akarjuk nevezni a csoportot `developer`-ről `web-developer`-re úgy, hogy a GID-et is megváltoztatjuk, az alábbit kell lefuttatnunk:

```
# groupmod -n web-developer -g 1050 developer
```

TIP Ne feledjük, hogy ha a `-g` kapcsolóval megváltoztatjuk a GID-et, minden olyan fájl és mappa GID-jét meg kell változtatnunk, amelyeknek továbbra is a csoporthoz kell tartozniuk.

Végül, ha törölni szeretnénk a `web-developer` csoportot, az alábbit kell futtatnunk:

```
# groupdel web-developer
```

Nem törölhető a csoport, ha az egy felhasználói fiók elsődleges csoportja. Ezért a csoport eltávolítása előtt el kell távolítanunk a felhasználót. A felhasználókhoz hasonlóan, ha törlünk egy csoportot, a csoporthoz tartozó fájlok a fájlrendszerben maradnak, és nem törlődnek, illetve nem rendelhetők másik csoporthoz.

A skeleton mappa

Amikor új felhasználói fiókot adunk hozzá úgy, hogy még a home mappáját is létrehozzuk, az újonnan létrehozott home mappát a skeleton (csontváz) mappából (alapértelmezés szerint `/etc/skel`) másolt fájlokkal és mappákkal töltjük fel. A mögöttes ötlet egyszerű: a rendszergazda olyan új felhasználókat akar hozzáadni, amelyeknek a home mappájában ugyanazok a fájlok és mappák vannak. Ezért, ha testre szeretnénk szabni az új felhasználói fiókok home mappájában automatikusan létrejövő fájlokat és mappákat, akkor ezeket az új fájlokat és mappákat a skeleton mappához kell hozzáadni.

TIP Vegyük figyelembe, hogy ha a skeleton mappában lévő összes fájlt és mappát listázni akarjuk, akkor az `ls -al` parancsot kell használnunk!

Az `/etc/login.defs` fájl

A Linuxban az `/etc/login.defs` fájl határozza meg a felhasználók és csoportok létrehozását vezérlő konfigurációs paramétereket. Ezenkívül az előző szakaszokban bemutatott parancsok ebből a fájlból veszik át az alapértelmezett értékeket.

A legfontosabb irányelvek a következők:

`UID_MIN` és `UID_MAX`

A felhasználói azonosítók azon tartománya, amelyet új közönséges felhasználókhoz lehet rendelni.

`GID_MIN` és `GID_MAX`

Az új közönséges csoportokhoz rendelhető csoportazonosítók tartománya.

`CREATE_HOME`

Annak megadása, hogy az új felhasználók számára alapértelmezés szerint létre kell-e hozni egy home mappát.

`USERGROUPS_ENAB`

Annak megadása, hogy a rendszer alapértelmezés szerint hozzon-e létre egy új csoportot minden új felhasználói fiókhoz, amelynek neve megegyezik a felhasználó nevével, és hogy a felhasználói fiók törlése a felhasználó elsődleges csoportját is törölje-e, ha az már nem tartalmaz tagokat.

`MAIL_DIR`

A mail spool mappa.

`PASS_MAX_DAYS`

A jelszó maximálisan használható napjainak száma.

`PASS_MIN_DAYS`

A jelszóváltoztatások között engedélyezett minimális napok száma.

`PASS_MIN_LEN`

A jelszó minimálisan elfogadható hossza.

`PASS_WARN_AGE`

A jelszó lejárta előtti figyelmeztető napok száma.

TIP A felhasználók és csoportok kezelése során mindig ellenőrizzük ezt a fájlt, hogy megtekinthessük és szükség esetén megváltoztathassuk a rendszer alapértelmezett viselkedését!

A `passwd` parancs

Ez a parancs elsősorban a felhasználó jelszavának megváltoztatására szolgál. Mint korábban leírtuk, bármelyik felhasználó megváltoztathatja a saját jelszavát, de csak a root módosíthatja *bármelyik* felhasználó jelszavát. Ez azért történik, mert a `passwd` parancsban be van állítva a SUID bit (egy `s` a tulajdonost jelző futtatható flag helyén), ami azt jelenti, hogy a fájl tulajdonosának (tehát a root-nak) a jogosultságaival hajtódik végre.

```
# ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 42096 mag 17  2015 /usr/bin/passwd
```

A használt `passwd` kapcsolóktól függően szabályozhatja a jelszó öregedésének bizonyos aspektusait:

`-d`

Törli a felhasználói fiók jelszavát (ezzel letiltja a felhasználót).

`-e`

Kényszeríti a felhasználói fiókot a jelszó megváltoztatására.

`-i`

Beállítja, hogy a jelszó lejárta után hány nap inaktivitás után frissítse a felhasználó a jelszavát (különben a fiók letiltásra kerül).

`-l`

Zárolja a felhasználói fiókot (a titkosított jelszó elé felkiáltójel kerül a `/etc/shadow` fájlban).

`-n`

A jelszó minimális élettartamának beállítása.

`-S`

Információk kiadása egy adott felhasználói fiók jelszóstátuszáról.

`-u`

A felhasználói fiók feloldása (a `/etc/shadow` fájl jelszó mezőjéből a felkiáltójel eltávolításra kerül).

`-x`

A jelszó maximális élettartamának beállítása.

`-w`

Beállítja a jelszó lejárta előtti figyelmeztető napok számát, amely alatt a felhasználó figyelmeztetést kap, hogy a jelszót meg kell változtatni.

NOTE

A csoportoknak is lehet jelszava, amelyet a `gpasswd` paranccsal lehet beállítani. Azok a felhasználók, akik nem tagjai egy csoportnak, de ismerik a jelszavát, a `newgrp` paranccsal ideiglenesen csatlakozhatnak a csoporthoz. Ne feledjük, hogy a `gpasswd` parancs arra is használható, hogy felhasználókat adjunk hozzá és

távolítsunk el egy csoportból, valamint hogy beállítsuk a csoport rendszergazdáinak és közönséges tagjainak listáját!

A `chage` parancs

Ez a parancs, amely a “change age” rövidítése, egy felhasználó jelszó-öregedési információinak megváltoztatására szolgál. A `chage` parancs a root-ra korlátozódik, kivéve a `-l` kapcsolót, amelyet a közönséges felhasználók is használhatnak a saját fiókjuk jelszó-öregedési információinak listázására.

A `chage` parancs további kapcsolói az alábbiak:

`-d`

A felhasználói fiók utolsó jelszóváltoztatásának beállítása.

`-E`

A felhasználói fiók lejárati dátumának beállítása.

`-I`

Beállítja, hogy a jelszó lejárta után hány nap inaktivitás után frissítse a felhasználó a jelszavát (különben a fiók letiltásra kerül).

`-m`

A felhasználói fiók minimális jelszó-élettartamának beállítása.

`-M`

A felhasználói fiók maximális jelszó-élettartamának beállítása.

`-W`

Beállítja a jelszó lejárta előtti figyelmeztető napok számát, amely alatt a felhasználó figyelmeztetést kap, hogy a jelszót meg kell változtatni.

Gyakorló feladatok

1. Adjuk meg az alábbi parancsok használatának célját:

`usermod -L`	
`passwd -u`	
`chage -E`	
`groupdel`	
`useradd -s`	
`groupadd -g`	
`userdel -r`	
`usermod -l`	
`groupmod -n`	
`useradd -m`	

2. Az alábbi `passwd` parancsokhoz adjuk meg a megfelelő `chage` parancsot:

`passwd -n`	
`passwd -x`	
`passwd -w`	
`passwd -i`	
`passwd -S`	

3. Fejtsük ki részletesen az előző kérdésben szereplő parancsok célját:

4. Milyen parancsokkal zárolható egy felhasználói fiók? És milyen parancsokkal lehet feloldani?

Gondolkodtató feladatok

1. A `groupadd` paranccsal hozzuk létre az `administrators` és `developers` csoportot! Tegyük fel, hogy root jogokkal rendelkezünk.

2. Most, hogy létrehoztuk ezeket a csoportokat, futtassuk a következő parancsot: `useradd -G administrators,developers kevin`. Milyen műveleteket hajt végre ez a parancs? Tegyük fel, hogy a `CREATE_HOME` és a `USERGROUPS_ENAB` az `/etc/login.defs' fájlban 'yes'-re van állítva.

3. Hozzunk létre egy új csoportot `designers` néven, majd nevezzük át `web-designers`-re és adjuk hozzá ezt az új csoportot a `kevin` felhasználói fiók másodlagos csoportjaihoz! Azonosítsuk az összes csoportot és az ID-jukat, ami `kevin`-hez tartozik.

4. Távolítsuk el csak a `developers` csoportot `kevin` másodlagos csoportjai közül!

5. Állítsuk be a `kevin` felhasználói fiók jelszavát!

6. A `chage` paranccsal először ellenőrizzük a `kevin` felhasználói fiók lejárati dátumát, majd módosítsuk azt 2022. december 31-re! Milyen más paranccsal módosíthatjuk a felhasználói fiók lejárati dátumát?

7. Hozzunk létre egy új felhasználói fiókot `emma` néven, az UID legyen 1050 és állítsuk be az `administrators`-t elsődleges csoportként, a `developers` és a `web-designers` csoportokat pedig másodlagosként!

8. Változtassuk meg `emma` login shelljét `/bin/sh`-ra!

9. Töröljük az `emma` és `kevin` felhasználói fiókokat, valamint az `administrators`, `developers` és `web-designers` csoportokat!

Összefoglalás

Ebben a leckében megtanultuk:

- A felhasználó- és csoportkezelés alapjai Linuxban.
- Felhasználói fiókok hozzáadása, módosítása és eltávolítása.
- Csoportfiókok hozzáadása, módosítása és eltávolítása.
- A skeleton mappa karbantartása.
- A felhasználók és csoportok létrehozását vezérlő fájl szerkesztése.
- A felhasználói fiókok jelszavainak módosítása.
- A felhasználói fiókok jelszóidőzítési információinak módosítása.

A leckében az alábbi fájlokat és parancsokat tárgyaltuk:

`useradd`
: Új felhasználói fiók létrehozása.

`usermod`
: Felhasználói fiók módosítása.

`userdel`
: Felhasználói fiók törlése.

`groupadd`
: Új csoport létrehozása.

`groupmod`
: Csoport módosítása.

`groupdel`
: Csoport törlése.

`passwd`
: Felhasználói fiókok jelszavának módosítása és a jelszó öregedésének minden aspektusának kontrollja.

`chage`
: A felhasználói jelszó lejárati idejének módosítása.

`/etc/skel`

A skeleton mappa alapértelmezett lokációja.

`/etc/login.defs`

A felhasználók és csoportok létrehozását vezérlő fájl, amely számos felhasználói fiókparaméter alapértelmezett értékét adja meg.

Válaszok a gyakorló feladatokra

1. Adjuk meg az alábbi parancsok használatának célját:

`usermod -L`	A felhasználói fiók zárolása
`passwd -u`	A felhasználói fiók feloldása
`chage -E`	A felhasználói fiók lejárati dátumának beállítása
`groupdel`	A csoport törlése
`useradd -s`	Új felhasználói fiók létrehozása egy adott login shellel
`groupadd -g`	Új csoport létrehozása specifikus GID-del
`userdel -r`	A felhasználói fiók és a home mappájában lévő össze fájl, valamint magának a home mappának és a felhasználó mail spooljának törlése
`usermod -l`	A felhasználói fiók bejelentkezési nevének megváltoztatása
`groupmod -n`	A csoport nevének a megváltoztatása
`useradd -m`	Új felhasználói fiók és a hozzá tartozó home mappa létrehozása

2. Az alábbi `passwd` parancsokhoz adjuk meg a megfelelő `chage` parancsot:

`passwd -n`	`chage -m`
`passwd -x`	`chage -M`
`passwd -w`	`chage -W`
`passwd -i`	`chage -I`
`passwd -S`	`chage -l`

3. Fejtsük ki részletesen az előző kérdésben szereplő parancsok célját:

   Linuxban a `passwd -n` paranccsal (vagy `chage -m`) beállíthatjuk a jelszóváltoztatások közötti minimális napok számát, a `passwd -x` paranccsal (vagy `chage -M`) a jelszó érvényességi idejének maximális számát, a `passwd -w` paranccsal (vagy `chage -W`) a jelszó lejárta előtti

figyelmeztető napok számát, a `passwd -i` paranccsal (vagy `chage -I`) beállítható, hogy a felhasználónak hány nap inaktivitás után kell megváltoztatnia a jelszavát, és a `passwd -S` paranccsal (vagy `chage -l`) megjeleníthetünk rövid információkat a felhasználói fiók jelszaváról.

4. Milyen parancsokkal zárolható egy felhasználói fiók? És milyen parancsokkal lehet feloldani?

   A fiók zárolásához az alábbi parancsok valamelyikét használhatjuk: `usermod -L`, `usermod --lock` és `passwd -l`. A feloldásához pedig az `usermod -U`, `usermod --unlock` és `passwd -u` parancsokat.

Válaszok a gondolkodtató feladatokra

1. A `groupadd` paranccsal hozzuk létre az `administrators` és `developers` csoportot! Tegyük fel, hogy root jogokkal rendelkezünk.

```
# groupadd administrators
# groupadd developers
```

2. Most, hogy létrehoztuk ezeket a csoportokat, futtassuk a következő parancsot: `useradd -G administrators,developers kevin`. Milyen műveleteket hajt végre ez a parancs? Tegyük fel, hogy a `CREATE_HOME` és a `USERGROUPS_ENAB` az `/etc/login.defs' fájlban 'yes'-re van állítva.

   A parancs egy új, `kevin` nevű felhasználót ad hozzá a rendszerben lévő felhasználók listájához, létrehozza a felhasználói fiókját (a `CREATE_HOME` igenre van állítva, ezért elhagyható a -m kapcsoló), és létrehoz egy új, `kevin` nevű csoportot, mint a felhasználói fiók elsődleges csoportját (a `USERGROUPS_ENAB` igenre van állítva). Végül a skeleton mappában található fájlok és mappák átmásolódnak a `kevin` home mappájába.

3. Hozzunk létre egy új csoportot `designers` néven, majd nevezzük át `web-designers`-re és adjuk hozzá ezt az új csoportot a `kevin` felhasználói fiók másodlagos csoportjaihoz! Azonosítsuk az összes csoportot és az ID-jukat, ami `kevin`-hez tartozik.

```
# groupadd designers
# groupmod -n web-designers designers
# usermod -a -G web-designers kevin
# id kevin
uid=1010(kevin) gid=1030(kevin)
groups=1030(kevin),1028(administrators),1029(developers),1031(web-designers)
```

4. Távolítsuk el csak a `developers` csoportot `kevin` másodlagos csoportjai közül!

```
# usermod -G administrators,web-designers kevin
# id kevin
uid=1010(kevin) gid=1030(kevin) groups=1030(kevin),1028(administrators),1031(web-
designers)
```

   A `usermod` parancsban nincs olyan lehetőség, hogy csak egy csoportot távolítson el; ezért meg kell adni az összes másodlagos csoportot, amelyhez a felhasználó tartozik.

5. Állítsuk be a `kevin` felhasználói fiók jelszavát!

```
# passwd kevin
Changing password for user kevin.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

6. A `chage` paranccsal először ellenőrizzük a `kevin` felhasználói fiók lejárati dátumát, majd módosítsuk azt 2022. december 31-re! Milyen más paranccsal módosíthatjuk a felhasználói fiók lejárati dátumát?

```
# chage -l kevin | grep "Account expires"
Account expires     : never
# chage -E 2022-12-31 kevin
# chage -l kevin | grep "Account expires"
Account expires     : dec 31, 2022
```

   A `usermod` parancs a `-e` kapcsolóval megfelel a `chage -E`-nek.

7. Hozzunk létre egy új felhasználói fiókot `emma` néven, az UID legyen 1050 és állítsuk be az `administrators`-t elsődleges csoportként, a `developers` és a `web-designers` csoportokat pedig másodlagosként!

```
# useradd -u 1050 -g administrators -G developers,web-designers emma
# id emma
uid=1050(emma) gid=1028(administrators)
groups=1028(administrators),1029(developers),1031(web-designers)
```

8. Változtassuk meg `emma` login shelljét `/bin/sh`-ra!

```
# usermod -s /bin/sh emma
```

9. Töröljük az `emma` és `kevin` felhasználói fiókokat, valamint az `administrators`, `developers` és `web-designers` csoportokat!

```
# userdel -r emma
# userdel -r kevin
# groupdel administrators
# groupdel developers
# groupdel web-designers
```

Linux Professional Institute

107.1 Lecke 2

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	107 Adminisztrációs feladatok
Fejezet:	107.1 Felhasználói- és csoport-fiókok, valamint a kapcsolódó rendszerfájlok kezelése
Lecke:	2/2

Bevezetés

Az előző leckében tárgyalt parancssori eszközök és az egyes disztribúciók által biztosított grafikus alkalmazások, amelyek ugyanazokat a feladatokat végzik el, frissítik a felhasználókra és csoportokra vonatkozó információkat tároló fájlokat.

Ezek a fájlok az `/etc/` mappában találhatók és az alábbiak:

`/etc/passwd`

A fájl hét, kettősponttal elválasztott mezőből áll, amelyek a felhasználók alapvető adatait tartalmazzák.

`/etc/group`

Négy, kettőspontokkal elválasztott mezőből álló fájl, amely a csoportok alapvető adatait tartalmazza.

`/etc/shadow`

Kilenc, kettőspontokkal elválasztott mezőből álló fájl, amely titkosított felhasználói jelszavakat

tartalmaz.

`/etc/gshadow`

Négy, kettőspontokkal elválasztott mezőből álló fájl, amely titkosított csoportjelszavakat tartalmaz.

Bár ez a négy fájl egyszerű szöveg, nem szabad közvetlenül szerkeszteni őket, hanem csak a használt disztribúció által biztosított eszközökkel.

/etc/passwd

Ez egy világosan olvasható fájl, amely a felhasználók listáját tartalmazza, minden egyes felhasználót külön sorban. Minden sor hét, kettősponttal elválasztott mezőből áll:

Username

A név, amivel a felhasználó belép a rendszerbe.

Password

A titkosított jelszó (vagy egy `x`, ha shadow jelszavak vannak használatban).

User ID (UID)

A rendszerben a felhasználóhoz tartozó ID.

Group ID (GID)

A felhasználó elsődleges csoportjának száma a rendszerben.

GECOS

Egy opcionális megjegyzés mező, amely a felhasználóval kapcsolatos további információk (például a teljes név) hozzáadására szolgál. A mező több, vesszővel elválasztott bejegyzést is tartalmazhat.

Home Directory

A felhasználó home mappájának abszolút elérési útja.

Shell

Annak a programnak az abszolút elérési útja, amely automatikusan elindul, amikor a felhasználó bejelentkezik a rendszerbe (általában egy interaktív shell, például `/bin/bash`).

/etc/group

Ez egy világosan olvasható fájl, amely a csoportok listáját tartalmazza, mindegyiket külön sorban. Minden sor négy, kettősponttal elválasztott mezőből áll:

Group Name

A csoport neve.

Group Password

A csoport jelszava titkosítva (vagy egy `x`, ha shadow jelszavak vannak használatban).

Group ID (GID)

A rendszerben a csoporthoz tartozó ID.

Member List

A csoporthoz tartozó felhasználók vesszővel elválasztott listája, kivéve azoké, akik számára ez az elsődleges csoport.

/etc/shadow

Ez egy csak a root és a root jogosultsággal rendelkező felhasználók által olvasható fájl, amely titkosított felhasználói jelszavakat tartalmaz, mindegyik külön sorban. Minden sor kilenc, kettősponttal elválasztott mezőből áll:

Username

A felhasználó rendszerbe való bejelentkezésekor használt név.

Encrypted Password

A felhasználó jelszava titkosítva (ha az érték elején van egy `!`, a fiók zárolva van).

Date of Last Password Change

A legutóbbi jelszóváltoztatás dátuma, az 1970.01.01. óta eltelt napok számaként (a 0 érték azt jelenti, hogy a felhasználónak a következő bejelentkezéskor meg kell változtatnia a jelszót).

Minimum Password Age

A jelszóváltoztatás után legalább ennyi napnak kell eltelnie, amíg a felhasználó újra megváltoztathatja a jelszavát.

Maximum Password Age

A jelszóváltoztatásig hátralévő napok maximális száma.

Password Warning Period

A jelszó lejárta előtti napok száma, ameddig a felhasználó figyelmeztetést kap a jelszó megváltoztatására.

Password Inactivity Period

A jelszó lejártát követő napok száma, amely alatt a felhasználónak frissítenie kell a jelszót. Ezen időszak letelte után, ha a felhasználó nem változtatja meg a jelszavát, a fiók letiltásra kerül.

Account Expiration Date

Az 1970. 01. 01. óta eltelt napok számában kifejezett dátum, amikor a felhasználói fiók letiltásra kerül (az üres mező azt jelenti, hogy a felhasználói fiók soha nem jár le).

A reserved field

Egy jövőbeni használatra fenntartott mező.

/etc/gshadow

Ez egy csak a root és a root jogosultsággal rendelkező felhasználók által olvasható fájl, amely titkosított csoportjelszavakat tartalmaz, mindegyik külön sorban. Minden sor négy, kettősponttal elválasztott mezőből áll:

Group Name

A csoport neve.

Encrypted Password

A csoport titkosított jelszava (ezt akkor használjuk, ha egy olyan felhasználó, aki nem tagja a csoportnak, a `newgrp` paranccsal szeretne csatlakozni a csoporthoz — ha a jelszó `!`-vel kezdődik, senki sem léphet be a csoportba a `newgrp` paranccsal).

Group Administrators

A csoport rendszergazdáinak vesszővel elválasztott listája (ők módosíthatják a csoport jelszavát, és a `gpasswd` paranccsal hozzáadhatnak vagy eltávolíthatnak csoporttagokat).

Group Members

A csoport tagjainak vesszővel elválasztott listája.

A jelszó- és csoportadatbázisok szűrése

Nagyon gyakran szükség lehet az e négy fájlban tárolt, felhasználókra és csoportokra vonatkozó információk áttekintésére és bizonyos rekordok keresésére. Ehhez a feladathoz használhatjuk a

`grep` parancsot, vagy alternatívaként a `cat` és a `grep` parancsok összekapcsolását.

```
# grep emma /etc/passwd
emma:x:1020:1020:User Emma:/home/emma:/bin/bash
# cat /etc/group | grep db-admin
db-admin:x:1050:grace,frank
```

Az adatbázisok elérésének másik módja a `getent` parancs használata. Ez a parancs általában a *Name Service Switch* (NSS) könyvtárak által támogatott adatbázisok bejegyzéseit jeleníti meg és az adatbázis nevét és egy keresőkulcsot igényel. Ha nincs kulcs-argumentum megadva, akkor a megadott adatbázis összes bejegyzése megjelenik (kivéve, ha az adatbázis nem támogatja a felsorolást). Ellenkező esetben, ha egy vagy több kulcsargumentumot adunk meg, az adatbázis szűrése ennek megfelelően történik.

```
# getent passwd emma
emma:x:1020:1020:User Emma:/home/emma:/bin/bash
# getent group db-admin
db-admin:x:1050:grace,frank
```

A `getent` parancs nem igényel root jogosultságot; csak arra van szükség, hogy képesek legyünk olvasni az adatbázist, amelyből a rekordokat le akarjuk kérni.

NOTE Ne feledjük, hogy a `getent` csak azokat az adatbázisokat éri el, amelyek a `/etc/nsswitch.conf` fájlban vannak konfigurálva!

Gyakorló feladatok

1. Nézzük meg az alábbi kimenetet és válaszoljuk meg az alábbi kérdéseket:

```
# cat /etc/passwd | grep '\(root\|mail\|catherine\|kevin\)'
root:x:0:0:root:/root:/bin/bash
mail:x:8:8:mail:/var/spool/mail:/sbin/nologin
catherine:x:1030:1025:User Chaterine:/home/catherine:/bin/bash
kevin:x:1040:1015:User Kevin:/home/kevin:/bin/bash
# cat /etc/group | grep '\(root\|mail\|db-admin\|app-developer\)'
root:x:0:
mail:x:8:
db-admin:x:1015:emma,grace
app-developer:x:1016:catherine,dave,christian
# cat /etc/shadow | grep '\(root\|mail\|catherine\|kevin\)'
root:$6$1u36Ipok$ljt8ooPMLewAhkQPf.lYgGopAB.jClTO6ljsdczxvkLPkpi/amgp.zyfAN680zrLLp2avvpd
KA0llpssdfcPppOp:18015:0:99999:7:::
mail:*:18015:0:99999:7:::
catherine:$6$ABCD25jlld14hpPthEFGnnssEWw1234yioMpliABCdef1f3478kAfhhAfgbAMjY1/BAeeAsl/FeE
dddKd12345g6kPACcik:18015:20:90:5:::
kevin:$6$DEFGabc123WrLp223fsvp0ddx3dbA7pPPc4LMaa123u6Lp02Lpvm123456pyphhh5ps012vbArL245.P
R1345kkA3Gas12P:18015:0:60:7:2::
# cat /etc/gshadow | grep '\(root\|mail\|db-admin\|app-developer\)'
root:*::
mail:*::
db-admin:!:emma:emma,grace
app-developer:!::catherine,dave,christian
```

   - Mi a felhasználói azonosítója (UID) és a csoportazonosítója (GID) a `root` és `catherine` felhasználóknak?

   - Mi a neve a `kevin` elsődleges csoportjának? Vannak más tagok is ebben a csoportban?

   - Melyik shell van beállítva a `mail`-nek? Ez mit jelent?

   - Kik az `app-developer` csoport tagjai? A tagok közül kik a csoport adminisztrátorai és kik az egyszerű tagok?

- Mennyi a minimális jelszó élettartama a `catherine` felhasználónak? És mennyi a maximális jelszó élettartama?

- Mennyi a jelszó inaktivitási ideje a `kevin` felhasználó esetén?

2. A konvenció szerint mely azonosítókat rendeljük a rendszerfiókokhoz és melyeket az egyszerű felhasználókhoz?

3. Hogyan lehet megtudni, hogy egy felhasználói fiók, amely korábban hozzáférhetett a rendszerhez, most zárolt-e? Tegyük fel, hogy a rendszer shadow jelszavakat használ.

Gondolkodtató feladatok

1. Hozzunk létre egy `christian` nevű felhasználói fiókot a `useradd -m` paranccsal, és határozzuk meg a felhasználói azonosítóját (UID), a csoport azonosítóját (GID) és a shell-t!

2. Határozzuk meg `christian` elsődleges csoportjának nevét! Mire következtethetünk?

3. A `getent` parancs segítségével nézzük meg a `christian` felhasználói fiók jelszavának öregedési információit!

4. Adjuk hozzá az `editor` csoportot a `christian` fiók másodlagos csoportjaihoz. Tegyük fel, hogy ez a csoport már tartalmazza az `emma`, `dave` és `frank` fiókokat rendes tagként. Hogyan tudjuk ellenőrizni, hogy nincsenek adminisztrátorok ebben a csoportban?

5. Futtassuk le az `ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow` parancsot és írjuk le a kimenetet, amelyet a jogosultságok tekintetében ad ki! Melyik négy fájl van biztonsági okokból árnyékolva? Tegyük fel, hogy a rendszer shadow jelszavakat használ!

Összefoglalás

Ebben a leckében megtanultuk:

- A felhasználókra és csoportokra vonatkozó információkat tároló fájlok helye.
- Jelszó- és csoportadatbázisokban tárolt felhasználói és csoportinformációk kezelése.
- Információk kinyerése jelszó- és csoportadatbázisokból.

A leckében használt fájlok és parancsok:

`/etc/passwd`
: A felhasználókra vonatkozó alapvető információkat tartalmazó fájl.

`/etc/group`
: A csoportokra vonatkozó alapvető információkat tartalmazó fájl.

`/etc/shadow`
: A titkosított felhasználói jelszavakat tartalmazó fájl.

`/etc/gshadow`
: A titkosított csoportjelszavakat tartalmazó fájl.

`getent`
: A jelszó- és csoportadatbázisok szűrése.

Válaszok a gyakorló feladatokra

1. Nézzük meg az alábbi kimenetet és válaszoljuk meg az alábbi kérdéseket:

```
# cat /etc/passwd | grep '\(root\|mail\|catherine\|kevin\)'
root:x:0:0:root:/root:/bin/bash
mail:x:8:8:mail:/var/spool/mail:/sbin/nologin
catherine:x:1030:1025:User Chaterine:/home/catherine:/bin/bash
kevin:x:1040:1015:User Kevin:/home/kevin:/bin/bash
# cat /etc/group | grep '\(root\|mail\|db-admin\|app-developer\)'
root:x:0:
mail:x:8:
db-admin:x:1015:emma,grace
app-developer:x:1016:catherine,dave,christian
# cat /etc/shadow | grep '\(root\|mail\|catherine\|kevin\)'
root:$6$1u36Ipok$ljt8ooPMLewAhkQPf.lYgGopAB.jClTO6ljsdczxvkLPkpi/amgp.zyfAN680zrLLp2avvpd
KA0llpssdfcPppOp:18015:0:99999:7:::
mail:*:18015:0:99999:7:::
catherine:$6$ABCD25jlld14hpPthEFGnnssEWw1234yioMpliABCdef1f3478kAfhhAfgbAMjY1/BAeeAsl/FeE
dddKd12345g6kPACcik:18015:20:90:5:::
kevin:$6$DEFGabc123WrLp223fsvp0ddx3dbA7pPPc4LMaa123u6Lp02Lpvm123456pyphhh5ps012vbArL245.P
R1345kkA3Gas12P:18015:0:60:7:2::
# cat /etc/gshadow | grep '\(root\|mail\|db-admin\|app-developer\)'
root:*::
mail:*::
db-admin:!:emma:emma,grace
app-developer:!::catherine,dave,christian
```

   - Mi a felhasználói azonosítója (UID) és a csoportazonosítója (GID) a `root` és `catherine` felhasználóknak?

     A `root` UID-ja és GID-je 0 és 0, míg `catherine` UID-ja és GID-je 1030 és 1025.

   - Mi a neve a `kevin` elsődleges csoportjának? Vannak más tagok is ebben a csoportban?

     A csoport neve `db-admin`. `emma` és `grace` vannak még a csoportban.

   - Melyik shell van beállítva a `mail`-nek? Ez mit jelent?

     A `mail` egy egy rendszerfelhasználói fiók, és a shellje a `/sbin/nologin`. Valójában a rendszerfelhasználói fiókok, mint például az `mail`, `ftp`, `news` és `daemon` adminisztrációs feladatok elvégzésére szolgálnak, ezért a normál bejelentkezést meg kell akadályozni ezen

fiókok esetén. Ezért a shell általában a `/sbin/nologin` vagy a `/bin/false` értékre van állítva.

- Kik az `app-developer` csoport tagjai? A tagok közül kik a csoport adminisztrátorai és kik az közönséges tagok?

  A tagok `catherine`, `dave` és `christian` és mindannyian közönséges felhasználók.

- Mennyi a minimális jelszó élettartama a `catherine` felhasználónak? És mennyi a maximális jelszó élettartama?

  A minimális jelszó élettartam 20 nap, a maximális pedig 90 nap.

- Mennyi a jelszó inaktivitási ideje a `kevin` felhasználó esetén?

  A jelszó inaktivitási ideje 2 nap. Ez alatt az időszak alatt `kevin` felhasználónak frissítenie kell a jelszót, különben a fiók letiltásra kerül.

2. A konvenció szerint mely azonosítókat rendeljük a rendszerfiókokhoz és melyeket az egyszerű felhasználókhoz?

   A rendszerfiókok általában 100-nál kisebb vagy 500 és 1000 közötti UID-vel rendelkeznek, míg a közönséges felhasználók UID-je 1000-től kezdődik, bár egyes régi rendszerek számozása 500-nál kezdődik. A `root` felhasználó UID-je 0. Ne feledjük, hogy az `/etc/login.defs` állományban található `UID_MIN` és `UID_MAX` értékek határozzák meg a közönséges felhasználók létrehozásához használt UID-k tartományát! Az LPI Linux Essentials és az LPIC-1 szempontjából a rendszerfiókok 1000-nél kisebb UID-vel rendelkeznek, a közönséges felhasználók pedig 1000-nél nagyobb UID-vel.

3. Hogyan lehet megtudni, hogy egy felhasználói fiók, amely korábban hozzáférhetett a rendszerhez, most zárolt-e? Tegyük fel, hogy a rendszer shadow jelszavakat használ.

   Amikor shadow jelszavakat használunk, az `/etc/passwd` fájl második mezőjében az `x` karakter tartozik minden felhasználói fiókhoz, mivel a titkosított felhasználói jelszavak az `/etc/shadow` állományban vannak tárolva. Egy felhasználói fiók titkosított jelszava tárolódik ennek a fájlnak a második mezőjében, és ha felkiáltójellel kezdődik, akkor a fiók zárolva van.

Válaszok a gondolkodtató feladatokra

1. Hozzunk létre egy `christian` nevű felhasználói fiókot a `useradd -m` paranccsal, és határozzuk meg a felhasználói azonosítóját (UID), a csoport azonosítóját (GID) és a shell-t!

```
# useradd -m christian
# cat /etc/passwd | grep christian
christian:x:1050:1060::/home/christian:/bin/bash
```

   `christian` UID-ja és GID-je 1050 és 1060 (a harmadik és negyedik mező az `/etc/passwd` fájlban). A `/bin/bash` shell van beállítva ehhez a felhasználói fiókhoz (a hetedik mező az `/etc/passwd` fájlban).

2. Határozzuk meg `christian` elsődleges csoportjának nevét! Mire következtethetünk?

```
# cat /etc/group | grep 1060
christian:x:1060:
```

   A `christian` elsődleges csoportjának neve `christian` (az `/etc/group` első mezője). Ezért az `/etc/login.defs` állományban az `USERGROUPS_ENAB` értéke yes, így a `useradd` alapértelmezés szerint létrehoz egy csoportot a felhasználói fiók nevével megegyező néven.

3. A `getent` parancs segítségével nézzük meg a `christian` felhasználói fiók jelszavának öregedési információit!

```
# getent shadow christian
christian:!:18015:0:99999:7:::
```

   A `christian` felhasználói fióknak nincs jelszava, és most zárolva van (az `/etc/shadow` második mezője felkiáltójelet tartalmaz). Ehhez a felhasználói fiókhoz nincs minimális és maximális jelszóéletkor (az `/etc/shadow` negyedik és ötödik mezője 0 és 99999 napra van beállítva), míg a jelszó figyelmeztetési időszaka 7 napra van beállítva (az `/etc/shadow` hatodik mezője). Végül pedig nincs inaktivitási időszak (az `/etc/shadow` hetedik mezője), és a fiók soha nem jár le (az `/etc/shadow` nyolcadik mezője).

4. Adjuk hozzá az `editor` csoportot a `christian` fiók másodlagos csoportjaihoz. Tegyük fel, hogy ez a csoport már tartalmazza az `emma`, `dave` és `frank` fiókokat rendes tagként. Hogyan tudjuk ellenőrini, hogy nincsenek adminisztrátorok ebben a csoportban?

```
# cat /etc/group | grep editor
editor:x:1100:emma,dave,frank
# usermod -a -G editor christian
# cat /etc/group | grep editor
editor:x:1100:emma,dave,frank,christian
# cat /etc/gshadow | grep editor
editor:!::emma,dave,frank,christian
```

Az `/etc/ghadow` harmadik és negyedik mezője tartalmazza a megadott csoport adminisztrátorait és közönséges tagjait. Mivel tehát a harmadik mező üres az `editor` esetében, nincsenek adminisztrátorok ebben a csoportban (`emma`, `dave`, `frank` és `christian` mind közönséges tagok).

5. Futtassuk le az `ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow` parancsot és írjuk le a kimenetet, amelyet a jogosultságokról ad meg! Melyik négy fájl van biztonsági okokból árnyékolva? Tegyük fel, hogy a rendszer shadow jelszavakat használ!

```
# ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow
-rw-r--r-- 1 root root    853 mag  1 08:00 /etc/group
-rw-r----- 1 root shadow 1203 mag  1 08:00 /etc/gshadow
-rw-r--r-- 1 root root   1354 mag  1 08:00 /etc/passwd
-rw-r----- 1 root shadow 1563 mag  1 08:00 /etc/shadow
```

Az `/etc/passwd` és az `/etc/group` fájlok a világ számára olvashatóak, és biztonsági okokból árnyékolva vannak. Ha shadow jelszavakat használunk, akkor ezeknek a fájloknak a második mezőjében egy `x` látható, mert a felhasználók és csoportok titkosított jelszavai a `/etc/shadow` és `/etc/gshadow` állományokban vannak tárolva, amelyek csak a root számára olvashatók, és ezen rendszer esetében még a `shadow` csoporthoz tartozó tagok számára is.

107.2 A rendszerfelügyeleti feladatok automatizálása jobok ütemezésével

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 102, Objective 107.2

Súlyozás

4

Kulcsfontosságú ismeretek

- Cron és at jobok kezelése
- A felhasználók hozzáférésének beállítása a cron és at szolgáltatásokhoz
- A systemd időzítő egységek megértése

A használt fájlok, kifejezések és segédprogramok listája

- `/etc/cron.{d,daily,hourly,monthly,weekly}/`
- `/etc/at.deny`
- `/etc/at.allow`
- `/etc/crontab`
- `/etc/cron.allow`
- `/etc/cron.deny`
- `/var/spool/cron/`
- `crontab`
- `at`
- `atq`
- `atrm`

- `systemctl`
- `systemd-run`

107.2 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	107 Adminsztrációs feladatok
Fejezet:	107.2 A rendszerfelügyeleti feladatok automatizálása jobok ütemezésével
Lecke:	1/2

Bevezetés

Egy jó rendszergazda egyik legfontosabb feladata a rendszeresen végrehajtandó jobok ütemezése. A rendszergazda például létrehozhat és automatizálhat jobokat biztonsági mentésekhez, rendszerfrissítésekhez és sok más ismétlődő tevékenység elvégzéséhez. Ehhez használhatja a `cron` lehetőséget, amely hasznos az időszakos jobok ütemezésének automatizálására.

Jobok ütemezése a Cron segítségével

A Linuxban a `cron` egy olyan daemon, amely folyamatosan fut és percenként felébred, hogy ellenőrizzen egy sor táblázatot, hogy megtalálja a végrehajtandó feladatokat. Ezek a táblázatok *crontabs* néven ismertek, és az úgynevezett *cronjobokat* tartalmazzák. A cron olyan szerverekhez és rendszerekhez alkalmas, amelyek folyamatosan be vannak kapcsolva, mert minden egyes cronjob csak akkor hajtódik végre, ha a rendszer a tervezett időpontban fut. Használhatják a közönséges felhasználók, akiknek mindegyike rendelkezik saját `crontab'-bal, valamint a root felhasználó, aki a rendszer crontabjait kezeli.

NOTE A Linuxban létezik az `anacron` lehetőség is, amely a kikapcsolható rendszerek

(például asztali számítógépek vagy laptopok) esetén megfelelő. Csak root felhasználók használhatják. Ha a gép ki van kapcsolva, amikor az `anacron` jobokat végre kell hajtani, akkor azok a gép következő bekapcsolásakor fognak lefutni. Az `anacron` nem tartozik az LPIC-1 tanúsítvány témaköreihez.

Felhasználói szintű crontabok

A *felhasználói szintű crontabok* (user crontab) olyan szöveges fájlok, amelyek a felhasználó által meghatározott cronjobok ütemezését kezelik. Nevüket mindig az őket létrehozó felhasználói fiók után kapják, de a fájlok helye a használt disztribúciótól függ (általában a `/var/spool/cron` almappájában).

A felhasználói crontab minden sora hat, szóközzel elválasztott mezőt tartalmaz:

- Az óra perce (0-59).
- A nap órája (0-23).
- A hónap napja (1-31).
- Az év hónapja (1-12).
- A hét napja (0-7, vasárnap=0 vagy vasárnap=7).
- A futtatandó parancs.

Az év hónapjának és a hét napjának megadásához a név első három betűjét is használhatjuk a megfelelő szám helyett.

Az első öt mező azt jelzi, hogy mikor kell végrehajtani a hatodik mezőben megadott parancsot, és egy vagy több értéket tartalmazhat. Több értéket is megadhatunk a következőkkel:

`*` (csillag)
: Bármilyen értékre vonatkozik.

`,` (vessző)
: A lehetséges értékek listáját adja meg.

`-` (kötőjel)
: Lehetséges értékek tartományát adja meg.

`/` (perjel)
: Fokozatos értékek megadása.

Sok disztribúció tartalmazza az `/etc/crontab` fájlt, amely referenciaként használható a `cron` fájl elrendezéséhez. Íme egy példa a `/etc/crontab` fájlra egy Debian telepítésből:

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
```

Rendszerszintű crontabok

A *rendszerszintű crontabok* (system crontab) olyan szöveges fájlok, amelyek a rendszer cronjobjainak ütemezését kezelik, és csak a root felhasználó szerkesztheti őket. Az `/etc/crontab` és az `/etc/cron.d` mappában található összes fájl rendszer crontab.

A legtöbb disztribúció tartalmazza a `/etc/cron.hourly`, `/etc/cron.daily`, `/etc/cron.weekly` és `/etc/cron.monthly` mappákat is, amelyek a megfelelő gyakorisággal futtatandó scripteket tartalmazzák. Ha például naponta akarunk futtatni egy scriptet, akkor azt az `/etc/cron.daily` mappában helyezhetjük el.

WARNING

Egyes disztribúciók a `/etc/cron.d/hourly`, `/etc/cron.d/daily`, `/etc/cron.d/weekly` és `/etc/cron.d/monthly` állományokat használják. Ne feledjük mindig ellenőrizni a megfelelő mappákat, ahová a cron által futtatni kívánt scripteket helyeznénk el!

A rendszerszintű crontabok szintaxisa hasonló a felhasználói crontabok szintaxisához, azonban egy további kötelező mezőre is szükség van, amely azt adja meg, hogy melyik felhasználó fogja futtatni a cronjobot. Ezért a rendszerszintű crontabok minden sora hét, szóközzel elválasztott mezőt tartalmaz:

- Az óra perce (0-59).
- A nap órája (0-23).
- A hónap napja (1-31).
- Az év hónapja (1-12).

- A hét napja (0-7, vasárnap=0 vagy vasárnap=7).
- A parancs végrehajtásakor használandó felhasználói fiók neve.
- A futtatandó parancs.

A felhasználói crontabok esetében is több értéket adhatunk meg az időmezőkhöz a `*`, `,` , `-` és `/` operátorok használatával. Az év hónapját és a hét napját a név első három betűjével is megadhatjuk a megfelelő szám helyett.

Különleges időspecifikációk

A crontab-fájlok szerkesztésekor az első öt oszlopban az időadatok helyett speciális gyorsbillentyűket is használhatunk:

`@reboot`
: A megadott feladat futtatása újraindítás után, egyszer.

`@hourly`
: A megadott feladat futtatása óránként egyszer, az óra elején.

`@daily` (vagy `@midnight`)
: A megadott feladat futtatása naponta egyszer, éjfélkor.

`@weekly`
: A megadott feladat futtatása hetente egyszer, vasárnap éjfélkor.

`@monthly`
: A megadott feladat futtatása havonta egyszer, a hónap első napján éjfélkor.

`@yearly` (vagy `@annually`)
: A megadott feladat futtatása évente egyszer, január 1-jén éjfélkor.

Crontab változók

A crontab fájlban lehetnek változó-hozzárendelések, amelyeket az ütemezett feladatok deklarálása előtt definiálunk. Az általában beállított környezeti változók a következők:

`HOME`
: A mappa, ahonnan a `cron` a parancsokat hívja (alapértelmezés szerint a felhasználó home mappája).

`MAILTO`

Annak a felhasználónak a neve vagy címe, akinek a standard kimenetet és a hibákat postázza (alapértelmezés szerint a crontab tulajdonosa). Több, vesszővel elválasztott érték is megengedett, az üres érték pedig azt jelzi, hogy nem kell levelet küldeni.

`PATH`

Az az útvonal, ahol a parancsok megtalálhatók.

`SHELL`

A használandó shell (alapértelmezés szerint `/bin/sh`).

Felhasználói szintű cronjobok létrehozása

A `crontab` parancs az egyes felhasználók crontab fájljainak karbantartására szolgál. A `crontab -e` parancsot használhatjuk a saját crontab fájlunk szerkesztésére, vagy létrehozására, ha még nem létezik.

```
$ crontab -e
no crontab for frank - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/ed
  2. /bin/nano        < ---- easiest
  3. /usr/bin/emacs24
  4. /usr/bin/vim.tiny

Choose 1-4 [2]:
```

Alapértelmezés szerint a `crontab` parancs megnyitja a `VISUAL` vagy `EDITOR` környezeti változók által megadott szerkesztőt, így a crontab fájl szerkesztését a kívánt szerkesztővel kezdhetjük el. Egyes disztribúciók, mint a fenti példában is látható, lehetővé teszik, hogy a `crontab` első futtatásakor egy listából kiválaszthassuk a szerkesztőt.

Ha minden nap 10:00-kor szeretnénk futtatni a home mappánkban található `foo.sh` scriptet, akkor a következő sort kell hozzáadnunk a crontab fájlhoz:

```
0 10 * * * /home/frank/foo.sh
```

Nézzük meg az alábbi crontab bejegyzéseket:

```
0,15,30,45 08 * * 2 /home/frank/bar.sh
30 20 1-15 1,6 1-5 /home/frank/foobar.sh
```

Az első sorban a `bar.sh` script minden kedden 08:00-kor, 08:15-kor, 08:30-kor és 08:45-kor kerül végrehajtásra. A második sorban a `foobar.sh` script hétfőtől péntekig, január és június első tizenöt napján 08:30-kor kerül végrehajtásra.

WARNING Bár a crontab fájlok kézzel is szerkeszthetők, mindig ajánlott a `crontab` parancs használata. A crontab fájlok jogosultságai általában csak a `crontab` paranccsal szerkeszthetők.

A fent említett `-e` kapcsoló mellett a `crontab` parancsnak más hasznos kapcsolói is vannak:

`-l`
: Az aktuális crontab megjelenítése a standard kimeneten.

`-r`
: Az aktuális crontab eltávolítása.

`-u`
: Annak a felhasználónak a neve, akinek a crontab-ját módosítani kell. Ez az opció root jogosultságokat igényel, és lehetővé teszi a root felhasználó számára a felhasználói crontab fájlok szerkesztését.

Rendszerszintű cronjobok létrehozása

A felhasználói crontabokkal ellentétben a rendszerszintű crontabok frissítése egy szerkesztőprogrammal történik: ezért nem kell a `crontab` parancsot futtatni az `/etc/crontab` és az `/etc/cron.d` állományok szerkesztéséhez. Ne feledjük, hogy a rendszerszintű crontab szerkesztésekor meg kell adni azt a fiókot, amely a cronjob futtatására szolgál (ez általában a root felhasználó)!

Ha például a `/root` mappában található `barfoo.sh` scriptet minden nap 01:30-kor szeretnénk futtatni, akkor nyissuk meg a `/etc/crontab` állományt a kívánt szerkesztőprogrammal, és adjuk hozzá a következő sort:

```
30 01 * * * root /root/barfoo.sh >>/root/output.log 2>>/root/error.log
```

A fenti példában a job kimenete a `/root/output.log`, míg a hibák a `/root/error.log` állományba kerülnek.

WARNING

Hacsak a kimenet nincs átirányítva egy fájlba, mint a fenti példában (vagy a `MAILTO` változó üres értékre van állítva), a cronjob minden kimenete e-mailben kerül elküldésre a felhasználónak. Általános gyakorlat, hogy a szabványos kimenetet átirányítjuk a `/dev/null'-ba (vagy egy fájlba a későbbi felülvizsgálathoz, ha szükséges), de nem irányítjuk át a szabványos hibát. Így a felhasználó azonnal értesítést kap e-mailben az esetleges hibákról.

A jobok ütemezéséhez való hozzáférés konfigurálása

Linuxban a `/etc/cron.allow` és `/etc/cron.deny` fájlokat használjuk a `crontab` korlátozások beállítására. Különösen arra szolgálnak, hogy engedélyezzék vagy megtiltsák a cronjobok ütemezését különböző felhasználók számára. Ha az `/etc/cron.allow` fájl létezik, akkor csak a benne felsorolt nem root felhasználók ütemezhetnek cronjobokat a `crontab` paranccsal. Ha az `/etc/cron.allow` nem létezik, de létezik az `/etc/cron.deny`, akkor csak az ebben a fájlban felsorolt nem root felhasználók nem ütemezhetnek cronjobokat a `crontab` paranccsal (ebben az esetben az üres `/etc/cron.deny` azt jelenti, hogy minden felhasználónak engedélyezett a cronjobok ütemezése a `crontab` paranccsal). Ha egyik fájl sem létezik, akkor a felhasználó hozzáférése a cronjobok ütemezéséhez a használt disztribúciótól függ.

NOTE

Az `/etc/cron.allow` és `/etc/cron.deny` fájlok a felhasználónevek listáját tartalmazzák, mindegyiket külön sorban.

A cron egy alternatívája

A systemd rendszer- és szolgáltatáskezelőt használva a *timers* a `cron` alternatívájaként állítható be a feladatok ütemezéséhez. Az időzítők a `.timer` utótaggal azonosított systemd egységfájlok, és mindegyikhez kell egy megfelelő egységfájl, amely leírja az időzítő lejártakor aktiválandó egységet. Alapértelmezés szerint egy `timer` aktiválja az azonos nevű szolgáltatást, kivéve a suffixet.

Az időzítő tartalmaz egy `[Timer]` szakaszt, amely meghatározza, hogy az ütemezett jobok mikor fussanak. Az `OnCalendar=` opciót használhatjuk a *real-time timers* (valósidejű időzítők) definiálására, amelyek ugyanúgy működnek, mint a cronjobok (a naptári eseménykifejezéseken alapulnak). Az `OnCalendar=` kapcsoló esetén a következő szintaxisra van szükség:

```
DayOfWeek Year-Month-Day Hour:Minute:Second
```

A `DayOfWeek` opcionális. A `*`, `/` és `,` operátoroknak ugyanaz a jelentésük, mint a cronjobok esetén, míg a két érték közötti `..` egy összefüggő tartományt jelöl. A `DayOfWeek` megadásához használhatjuk a nap nevének első három betűjét vagy akár a teljes nevét.

NOTE Meghatározhatunk *monotonikus időzítőket* (monotonic timers) is, amelyek egy adott kezdőponttól számított bizonyos idő elteltével aktiválódnak (például amikor a gép elindult, vagy amikor maga az időzítő aktiválódik).

Ha például a `/etc/systemd/system/foobar.service` nevű szolgáltatást minden hónap első hétfőjén 05:30-kor szeretnénk futtatni, akkor a következő sorokat adhatjuk hozzá a megfelelő `/etc/systemd/system/foobar.timer` egységfájlhoz.

```
[Unit]
Description=Run the foobar service

[Timer]
OnCalendar=Mon *-*-1..7 05:30:00
Persistent=true

[Install]
WantedBy=timers.target
```

Miután létrehoztuk az új időzítőt, engedélyezhetjük és elindíthatjuk a következő parancsok root felhasználóként történő futtatásával:

```
# systemctl enable foobar.timer
# systemctl start foobar.timer
```

Az ütemezett job gyakoriságát megváltoztathatjuk az `OnCalendar` érték módosításával, majd a `systemctl daemon-reload` parancs beírásával.

Végül, ha az aktív időzítők listáját szeretnénk megtekinteni a következő idő szerint rendezve, akkor használhatjuk a `systemctl list-timers` parancsot. Az `--all` kapcsolót is hozzáadhatjuk, hogy az inaktív időzítőegységeket is lássuk.

NOTE Ne feledjük, hogy az időzítők naplózva vannak a systemd journal fájlban és a különböző egységek logjait a `journalctl` paranccsal tekinthetjük meg! Ne feledjük el azt sem, hogy ha közönséges felhasználók vagyunk, akkor a `systemctl` és a `journalctl` parancsok `--user` kapcsolóját kell használnunk!

A fent említett hosszabb normalizált formátum helyett használhatunk néhány speciális kifejezést, amelyek a jobok végrehajtásának bizonyos gyakoriságát írják le:

`hourly`

A megadott feladatot futtatása óránként egyszer, az óra elején.

`daily`

A megadott feladat futtatása naponta egyszer, éjfélkor.

`weekly`

A megadott feladat futtatása hetente egyszer, hétfőn éjfélkor.

`monthly`

A megadott feladat futtatása havonta egyszer, a hónap első napján éjfélkor.

`yearly`

A megadott feladat futtatása évente egyszer, január első napján éjfélkor.

Az idő- és dátum-specifikációk teljes listáját a `systemd.timer(5)` man oldalain találhatjuk meg.

Gyakorló feladatok

1. Az alábbi `crontab` rövidítésekhez adjuk meg a megfelelő időmeghatározást (például a felhasználói `crontab` fájl első öt oszlopát):

`@hourly`	
`@daily`	
`@weekly`	
`@monthly`	
`@annually`	

2. Az alábbi `OnCalendar` rövidítésekhez adjuk meg a megfelelő időmeghatározást (a hosszabb normalizált formátumot):

`hourly`	
`daily`	
`weekly`	
`monthly`	
`yearly`	

3. Magyarázzuk meg az alábbi, `crontab` fájlban található időadatok jelentését:

`30 13 * * 1-5`	
`00 09-18 * * *`	
`30 08 1 1 *`	
`0,20,40 11 * * Sun`	
`00 09 10-20 1-3 *`	
`*/20 * * * *`	

4. Magyarázzuk meg az alábbi, timer fájl `OnCalendar` opciójában használt következő időmeghatározások jelentését:

`*-*-* 08:30:00`	
`Sat,Sun *-*-* 05:00:00`	

`*-*-01 13:15,30,45:00`	
`Fri *-09..12-* 16:20:00`	
`Mon,Tue *-*-1,15 08:30:00`	
`*-*-* *:00/05:00`	

Gondolkodtató feladatok

1. Feltételezve, hogy közönséges felhasználóként jogosultak vagyunk a `cron` programmal jobokat ütemezni, milyen parancsot használhatunk a saját crontab fájlunk létrehozásához?

2. Hozzunk létre egy egyszerű ütemezett jobot, amely minden pénteken 01:00-kor végrehajtja a `date` parancsot! Hol láthatjuk a job kimenetét?

3. Hozzunk létre egy másik ütemezett jobot, amely percenként végrehajtja a `foobar.sh` scriptet, átirányítva a kimenetet a home mappában lévő `output.log` fájlba, így e-mailben csak a standard hiba kerüljön elküldésre!

4. Nézzük meg az újonnan létrehozott ütemezett job `crontab` bejegyzését! Miért nem szükséges megadni annak a fájlnak az abszolút elérési útját, amelybe a szabványos kimenetet menti? És miért használhatjuk a `./foobar.sh` parancsot a script futtatására?

5. Szerkesszük az előző `crontab` bejegyzést úgy, hogy eltávolítjuk a kimenet átirányítását, és tiltsuk le az első létrehozott cron jobot!

6. Hogyan küldhetjük el e-mailben az ütemezett job kimenetét és hibáit az `emma` felhasználói fióknak? És hogyan lehet elkerülni, hogy a standard kimenetet és a hibákat e-mailben küldjük el?

7. Futtassuk az `ls -l /usr/bin/crontab` parancsot! Melyik speciális bit van beállítva, és mi a jelentése?

Összefoglalás

Ebben a leckében megtanultuk:

- A `cron` használatát a jobok rendszeres időközönkénti futtatásához.
- A cronjobok menedzselését.
- A felhasználói hozzáférés beállítását a cronjobok ütemezéséhez.
- A systemd időzítő egységek szerepének megértését a `cron` alternatívájaként.

A leckében használt fájlok és parancsok:

`crontab`

Az egyes felhasználók `crontab` fájljainak karbantartása.

`/etc/cron.allow` és **`/etc/cron.deny`**

A `crontab` korlátozások beállításához használt speciális fájlok.

`/etc/crontab`

Rendszerszintű crontab fájl.

`/etc/cron.d`

A mappa, ami a rendszerszintű crontab fájokat tartalmazza.

`systemctl`

A systemd rendszer- és szolgáltatáskezelő vezérlése. Az időzítőkkel kapcsolatban azok engedélyezésére és indítására használható.

Válaszok a gyakorló feladatokra

1. Az alábbi `crontab` rövidítésekhez adjuk meg a megfelelő időmeghatározást (azaz a felhasználói `crontab` fájl első öt oszlopát):

`@hourly`	`0 * * * *`
`@daily`	`0 0 * * *`
`@weekly`	`0 0 * * 0`
`@monthly`	`0 0 1 * *`
`@annually`	`0 0 1 1 *`

2. Az alábbi `OnCalendar` rövidítésekhez adjuk meg a megfelelő időmeghatározást (a hosszabb normalizált formátumot):

`hourly`	`*-*-* *:00:00`
`daily`	`*-*-* 00:00:00`
`weekly`	`Mon *-*-* 00:00:00`
`monthly`	`*-*-01 00:00:00`
`yearly`	`*-01-01 00:00:00`

3. Magyarázzuk meg az alábbi, `crontab` fájlban található időadatok jelentését:

`30 13 * * 1-5`	13:30-kor a hét minden napján, hétfőtől péntekig
`00 09-18 * * *`	Minden nap és minden órában 09 órától 06 óráig
`30 08 1 1 *`	Január első napján 08:30-kor
`0,20,40 11 * * Sun`	Minden vasárnap 11:00, 11:20 és 11:40 órakor
`00 09 10-20 1-3 *`	Január, február és március 10. és 20. között reggel 9:00 órakor
`*/20 * * * *`	Húsz percenként

4. Magyarázzuk meg az alábbi, timer fájl `OnCalendar` opciójában használt következő időmeghatározások jelentését:

`*-*-* 08:30:00`	Minden nap 08:30-kor
`Sat,Sun *-*-* 05:00:00`	Szombaton és vasárnap 05:00 órakor
`*-*-01 13:15,30,45:00`	A hónap első napján 13:15-kor, 13:30-kor és 13:45-kor
`Fri *-09..12-* 16:20:00`	Szeptemberben, októberben, novemberben és decemberben minden pénteken 16:20-kor
`Mon,Tue *-*-1,15 08:30:00`	Minden hónap első vagy tizenötödik napján reggel 8:30-kor, de csak akkor, ha ez a nap hétfőre vagy keddre esik
`*-*-* *:00/05:00`	Öt percenként

Válaszok a gondolkodtató feladatokra

1. Feltételezve, hogy egyszerű felhasználóként jogosultak vagyunk a `cron` programmal jobokat ütemezni, milyen parancsot használhatunk a saját crontab fájlunk létrehozásához?

```
dave@hostname ~ $ crontab -e
no crontab for dave - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/ed
  2. /bin/nano        < ---- easiest
  3. /usr/bin/emacs24
  4. /usr/bin/vim.tiny

Choose 1-4 [2]:
```

2. Hozzunk létre egy egyszerű ütemezett jobot, amely minden pénteken 01:00-kor végrehajtja a `date` parancsot! Hol láthatjuk a job kimenetét?

```
00 13 * * 5 date
```

   A kimenetet a felhasználó e-mailben kapja meg, ahhoz, hogy ezt megnézzük, használhatjuk a `mail` parancsot.

3. Hozzunk létre egy másik ütemezett jobot, amely percenként végrehajtja a `foobar.sh` scriptet, átirányítva a kimenetet a home mappában lévő `output.log` fájlba, így e-mailben csak a standard hiba kerüljön elküldésre!

```
*/1 * * * * ./foobar.sh >> output.log
```

4. Nézzük meg az újonnan létrehozott ütemezett job `crontab` bejegyzését! Miért nem szükséges megadni annak a fájlnak az abszolút elérési útját, amelybe a szabványos kimenetet menti? És miért használhatjuk a `./foobar.sh` parancsot a script futtatására?

   A `cron` a parancsok a felhasználó home mappájából hívódnak, hacsak a `crontab` fájlban a `HOME` környezeti változóval nem adunk meg más helyet. Emiatt használhatjuk a kimeneti fájl relatív elérési útvonalát, és a scriptet a `./foobar.sh` paranccsal futtathatjuk.

5. Szerkesszük az előző `crontab` bejegyzést úgy, hogy eltávolítjuk a kimenet átirányítását, és tiltsuk le az első létrehozott cron jobot!

```
#00 13 * * 5 date
*/1 * * * * ./foobar.sh
```

Egy cron job letiltásához egyszerűen kommentezzük ki a megfelelő sort a `crontab` fájlban.

6. Hogyan küldhetjük el e-mailben az ütemezett munka kimenetét és hibáit az `emma` felhasználói fióknak? És hogyan lehet elkerülni, hogy a standard kimenetet és a hibákat e-mailben küldjük el?

   Ahhoz, hogy a standard kimenetet és hibát `emma` címére küldjük, a `MAILTO` környezeti változót kell beállítanunk a `crontab` fájlban az alábbiak szerint:

   ```
   MAILTO="emma"
   ```

   Ahhoz, hogy megadjuk a `cron`-nak, hogy nem kell e-mailt küldeni, a `MAILTO` környezeti változóhoz üres értéket kell rendelnünk!

   ```
   MAILTO=""
   ```

7. Futtassuk az `ls -l /usr/bin/crontab` parancsot! Melyik speciális bit van beállítva, és mi a jelentése?

   ```
   $ ls -l /usr/bin/crontab
   -rwxr-sr-x 1 root crontab 25104 feb 10  2015 /usr/bin/crontab
   ```

   A `crontab` parancsban az SGID bit be van állítva (az `s` karakter a csoport futtatható flagje helyén), ami azt jelenti, hogy a parancs a csoport (tehát a `crontab`) jogosultságaival kerül végrehajtásra. Ezért van az, hogy a közönséges felhasználók a `crontab` paranccsal szerkeszthetik a `crontab` fájlt. Vegyük figyelembe, hogy sok disztribúcióban a fájljogosultságok úgy vannak beállítva, hogy a `crontab` fájlok csak a `crontab` paranccsal szerkeszthetők!

Linux Professional Institute

107.2 Lecke 2

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	107 Adminisztratív feladatok
Fejezet:	107.2 A rendszerfelügyeleti feladatok automatizálása jobok ütemezésével
Lecke:	2/2

Bevezetés

Amint azt az előző leckében megtanultuk, a cron vagy systemd időzítők segítségével ütemezhetünk rendszeres jobokat, de néha előfordulhat, hogy egy jobot a jövőben csak egyszer kell lefuttatnunk egy adott időpontban. Ehhez egy másik hatékony segédprogramot használhatunk: az `at` parancsot.

Jobok ütemezése az `at` segítségével

Az `at` parancs egyszeri feladatütemezéshez használható, és csak azt kell megadnunk, hogy a jobot mikor kell a jövőben futtatni. Miután a parancssorba beírtuk az `at` parancsot, majd az időpontot, az `at` promptba lépünk, ahol meghatározhatjuk a végrehajtandó parancsokat. A promptból a `Ctrl` + `D` billentyűkombinációval léphetünk ki.

```
$ at now +5 minutes
warning: commands will be executed using /bin/sh
at> date
at> Ctrl+D
```



```
job 12 at Sat Sep 14 09:15:00 2019
```

A fenti példában szereplő `at` job egyszerűen öt perc múlva végrehajtja a `date` parancsot. A `cron` -hoz hasonlóan a standard kimenetet és a hibát e-mailben küldi el. Vegyük figyelembe, hogy az `atd` daemonnak futnia kell ahhoz, hogy használhassuk az `at` ütemezőt.

NOTE A Linuxban a `batch` parancs hasonló az `at` parancshoz, azonban a `batch` jobok csak akkor kerülnek végrehajtásra, ha a rendszer terhelése elég alacsony ahhoz, hogy ezt lehetővé tegye.

Az `at` paranccsal kapcsolatos legfontosabb kapcsolók:

`-c`
Egy adott job azonosítójához tartozó parancsokat ír ki a standard kimenetre.

`-d`
Jobok törlése az azonosítójuk alapján. Ez az `atrm` aliasa.

`-f`
A szabványos bemenet helyett egy fájlból olvassa be a jobot.

`-l`
A felhasználó függőben lévő jobjainak listázása. Ha a felhasználó a root, akkor az összes felhasználó összes jobját listázza. Ez az `atq` aliasa.

`-m`
E-mail küldése a felhasználónak a job végén, még akkor is, ha nem volt kimenet.

`-q`
Egyetlen betűvel adjuk meg a várólistát `a`-tól `z`-ig és `A`-tól `Z`-ig (alapértelmezés szerint `a` az `at` és `b` a `batch` esetén). A legmagasabb betűjelű sorokban lévő jobok fokozott prioritással kerülnek végrehajtásra. A nagybetűs sorba beküldött jobokat `batch` jobként kezeljük.

`-v`
Megjeleníti azt az időpontot, amikor a job lefut a job olvasása előtt.

Ütemezett jobok listázása az `atq` segítségével

Most ütemezzünk be két további `at` jobot: az első a `foo.sh` scriptet hajtja végre 09:30-kor, míg a második a `bar.sh` scriptet egy óra múlva!

```
$ at 09:30 AM
warning: commands will be executed using /bin/sh
at> ./foo.sh
at> Ctrl+D
job 13 at Sat Sep 14 09:30:00 2019
$ at now +2 hours
warning: commands will be executed using /bin/sh
at> ./bar.sh
at> Ctrl+D
job 14 at Sat Sep 14 11:10:00 2019
```

A függőben lévő jobok listázásához használhatjuk az `atq` parancsot, amely a következő információkat mutatja minden egyes jobról: a job azonosítója, a job végrehajtásának dátuma, a job végrehajtásának ideje, a várólista és a felhasználónév.

```
$ atq
14      Sat Sep  14 11:10:00 2019 a frank
13      Sat Sep  14 09:30:00 2019 a frank
12      Sat Sep  14 09:15:00 2019 a frank
```

Ne feledjük, hogy az `at -l` parancs az `atq` aliasa.

NOTE Ha rootként futtatjuk az `atq`-t, akkor az összes felhasználó ütemezett jobját megjeleníti.

Jobok törlése az `atrm` segítségével

Ha egy `at` jobot szeretnénk törölni, akkor használhatjuk az `atrm` parancsot, amelyet a job azonosítója (ID) követ. Például a 14-es azonosítójú job törléséhez futtassuk a következőt:

```
$ atrm 14
```

Az `atrm` segítségével több jobot is törölhetünk, ha több azonosítót adunk meg, szóközzel elválasztva. Ne feledjük, hogy az `at -d` parancs az `atrm` aliasa!

NOTE Ha rootként futtatjuk az `atrm` parancsot, bármelyik felhasználó jobjait törölhetjük.

A jobok ütemezéséhez való hozzáférés konfigurálása

Az átlagos felhasználók `at` job-ütemezésének engedélyezését az `/etc/at.allow` és `/etc/at.deny` fájlok határozzák meg. Ha létezik az `/etc/at.allow` fájl, akkor csak a benne felsorolt nem root felhasználók ütemezhetnek `at` jobokat. Ha az `/etc/at.allow` fájl nem létezik, de az `/etc/at.deny` létezik, akkor csak a benne felsorolt nem root felhasználók nem ütemezhetnek `at` feladatokat (ebben az esetben az üres `/etc/at.deny` fájl azt jelenti, hogy minden felhasználónak engedélyezett az `at` feladatok ütemezése). Ha egyik fájl sem létezik, akkor a felhasználó hozzáférése az `at` jobok ütemezéséhez a használt disztribúciótól függ.

Időspecifikációk

A `HH:MM` formátumot használva megadhatjuk, hogy egy adott `at` job mikor kerüljön végrehajtásra, amelyet 12 órás formátum esetén választhatóan AM vagy PM követ. Ha a megadott időpont már eltelt, akkor a következő nap lesz feltételezve. Ha egy adott dátumot szeretnénk beütemezni, amelyen a job futni fog, akkor a dátuminformációt az időpont után kell hozzáadni a következő formátumok valamelyikével: `month-name day-of-month`, `month-name day-of-month year`, `MMDDYY`, `MM/DD/YY`, `DD.MM.YY` és `YYYYYY-MM-DD`).

Az alábbi kulcsszavak is elfogadottak: `midnight`, `noon`, `teatime` (16:00) és `now`, amelyet egy pluszjel (`+`) és időperiódusok (percek, órák, napok és hetek) követnek. Végül, az `at`-t a mai vagy a holnapi job futtatására is utasíthatjuk, ha az időpontot a `today` vagy `tomorrow` szavakkal egészítjük ki. Például az `at 07:15 AM Jan 01 Jan 01` használatával január 01-én 07:15-kor hajthatunk végre egy feladatot, az `at now +5 minutes` használatával pedig öt perc múlva. Az `/usr/share` alatt található `timespec` fájlban további információkat találhatunk az időspecifikációk pontos meghatározásáról.

Az `at` egy alternatívája

A systemd rendszer- és szolgáltatáskezelőt használva egyszeri feladatokat is ütemezhetünk a `systemd-run` paranccsal. Általában átmeneti időzítőegység létrehozására használják, hogy egy parancs egy adott időpontban végrehajtásra kerüljön anélkül, hogy szolgáltatásfájlt kellene létrehozni. Például root felhasználóként eljárva a `date` parancsot 2019.10.06-án 11:30-kor futtathatjuk a következőkkel:

```
# systemd-run --on-calendar='2019-10-06 11:30' date
```

Ha két perc múlva futtatni szeretnénk a `foo.sh` scriptet, ami az aktuális mappánkban van, az alábbit használhatjuk:

```
# systemd-run --on-active="2m" ./foo.sh
```

A `systemd-run` és a `systemd-run(1)` összes lehetséges felhasználási módját megismerhetjük a man oldalaikon.

NOTE

Ne feledjük, hogy az időzítők naplózva vannak a systemd journal fájljában, és a különböző egységek logjait a `journalctl` paranccsal tekinthetjük meg! Ne feledjük azt sem, hogy ha közönséges felhasználóként járunk el, akkor a `systemd-run` és a `journalctl` parancsok `--user` kapcsolóját kell használnunk!

Gyakorló feladatok

1. A következő időmeghatározások mindegyikénél jelöljük, hogy melyik érvényes és melyik érvénytelen az `at` esetében:

`at 08:30 AM next week`	
`at midday`	
`at 01-01-2020 07:30 PM`	
`at 21:50 01.01.20`	
`at now +4 days`	
`at 10:15 PM 31/03/2021`	
`at tomorrow 08:30 AM`	

2. Ha egyszer már ütemeztünk egy jobot az `at` segítségével, hogyan tudjuk felülvizsgálni a parancsokat?

3. Mely parancsokat használhatjuk a függőben lévő `at` jobok áttekintésére? Mely parancsokkal törölhetjük őket?

4. A systemd esetében melyik parancsot használjuk a `at` alternatívájaként?

Gondolkodtató feladatok

1. Hozzunk létre egy `at` jobot, amely október 31-én 10:30-kor lefuttatja a home mappában található `foo.sh` scriptet! Tegyük fel, hogy közönséges felhasználók vagyunk!

2. Jelentkezzünk be a rendszerbe egy másik közönséges felhasználóként, és hozzunk létre egy másik `at` jobot, amely holnap 10:00-kor futtatja a `bar.sh` scriptet! Tegyük fel, hogy a script a felhasználó home mappájában található!

3. Jelentkezzünk be a rendszerbe egy másik közönséges felhasználóként és hozzunk létre egy újabb `at` jobot, ami futtatja a `foobar.sh` scriptet 30 perc múlva! Tegyük fel, hogy a script a felhasználó home mappájában található!

4. Most root felhasználóként futtassuk az `atq` parancsot az összes felhasználó ütemezett `at` jobjainak áttekintéséhez! Mi történik, ha egy közönséges felhasználó hajtja végre ezt a parancsot?

5. Root felhasználóként töröljük az összes függőben lévő `at` jobot egyetlen paranccsal!

6. Futtassuk az `ls -l /usr/bin/at` parancsot és nézzük meg a jogosultságokat!

Összefoglalás

Ebben a leckében megtanultuk:

- Az `at` használatát egyszeri jobok futtatásához egy adott időpontban.
- Az `at` jobok menedzselését.
- Felhasználói hozzáférés beállítását az `at` jobok ütemezéséhez.
- A `systemd-run` használata az `at` alternatívájaként.

Az alábbi parancsokat és fájlokat tárgyaltuk a leckében:

`at`
: Parancsok végrehajtása egy megadott időpontban.

`atq`
: A felhasználó függőben lévő `at` jobjainak listázása, kivéve, ha a felhasználó a root.

`atrm`
: Az `at` jobok törlése, az azonosítószámuk alapján azonosítva.

`/etc/at.allow` and `/etc/at.deny`
: Az `at` korlátozások beállítására használt fájlok.

`systemd-run`
: Tranziens `timer` egység létrehozása az `at` alternatívájaként az egyszeri ütemezéshez.

Válaszok a gyakorló feladatokra

1. A következő időmeghatározások mindegyikénél jelöljük, hogy melyik érvényes és melyik érvénytelen az `at` esetében:

`at 08:30 AM next week`	Valid
`at midday`	Invalid
`at 01-01-2020 07:30 PM`	Invalid
`at 21:50 01.01.20`	Valid
`at now +4 days`	Valid
`at 10:15 PM 31/03/2021`	Invalid
`at tomorrow 08:30 AM monotonic`	Invalid

2. Ha egyszer már ütemeztünk egy jobot az `at` segítségével, hogyan tudjuk felülvizsgálni a parancsokat?

   Használhatjuk az `at -c` parancsot, amelyet annak a jobnak az azonosítója követ, amelynek a parancsait szeretnénk áttekinteni. Vegyük figyelembe, hogy a kimenet tartalmazza a job ütemezésének időpontjában aktív környezet nagy részét is! Ne feledjük, hogy a root minden felhasználó feladatait megtekintheti!

3. Mely parancsokat használhatjuk a függőben lévő `at` jobok áttekintésére? Mely parancsokkal törölhetjük őket?

   A `at -l` paranccsal áttekinthetjük a függőben lévő jobjainkat, az `at -d` paranccsal pedig törölhetjük őket. Az `at -l` az `atq`, az `at -d` pedig az `atrm` aliasa. Ne feledjük, hogy a root minden felhasználó munkáit listázhatja és törölheti!

4. A systemd esetében melyik parancsot használjuk a `at` alternatívájaként?

   A `systemd-run` parancs az `at` parancs alternatívájaként használható egyszeri jobok ütemezésére. Használhatjuk például parancsok egy adott időpontban történő futtatására, különböző kezdőpontokhoz viszonyított *naptári időzítő* vagy *monotonikus időzítő* meghatározásával.

Válaszok a gondolkodtató feladatokra

1. Hozzunk létre egy `at` jobot, amely október 31-én 10:30-kor lefuttatja a home mappában található `foo.sh` scriptet! Tegyük fel, hogy egyszerű felhasználók vagyunk!

```
$ at 10:30 AM October 31
warning: commands will be executed using /bin/sh
at> ./foo.sh
at> Ctrl+D
job 50 at Thu Oct 31 10:30:00 2019
```

2. Jelentkezzünk be a rendszerbe egy másik közönséges felhasználóként, és hozzunk létre egy másik `at` jobot, amely holnap 10:00-kor futtatja a `bar.sh` scriptet! Tegyük fel, hogy a script a felhasználó home mappájában található!

```
$ at 10:00 AM tomorrow
warning: commands will be executed using /bin/sh
at> ./bar.sh
at> Ctrl+D
job 51 at Sun Oct 6 10:00:00 2019
```

3. Jelentkezzünk be a rendszerbe egy másik közönséges felhasználóként és hozzunk létre egy újabb `at` jobot, ami futtatja a `foobar.sh` scriptet 30 perc múlva! Tegyük fel, hogy a script a felhasználó home mappájában található!

```
$ at now +30 minutes
warning: commands will be executed using /bin/sh
at> ./foobar.sh
at> Ctrl+D
job 52 at Sat Oct 5 10:19:00 2019
```

4. Most root felhasználóként futtassuk az `atq` parancsot az összes felhasználó ütemezett `at` jobjainak áttekintéséhez! Mi történik, ha egy közönséges felhasználó hajtja végre ezt a parancsot?

```
# atq
52      Sat Oct  5 10:19:00 2019 a dave
50      Thu Oct 31 10:30:00 2019 a frank
51      Sun Oct  6 10:00:00 2019 a emma
```

Ha rootként futtatjuk le az `atq` parancsot, minden függőben lévő `at` job kilistázódik, bárki is legyen a tulajdonosa. Ha egyszerű felhasználóként futtatjuk, csak a saját függőben lévő `at` feladataink kerülnek kilistázásra.

5. Root felhasználóként töröljük az összes függőben lévő `at` jobot egyetlen paranccsal!

```
# atrm 50 51 52
```

6. Futtassuk az `ls -l /usr/bin/at` parancsot és nézzük meg a jogosultságokat!

```
# ls -l /usr/bin/at
-rwsr-sr-x 1 daemon daemon 43762 Dec  1  2015 /usr/bin/at
```

Ebben a disztribúcióban az `at` parancsnak mind a SUID (az `s` a tulajdonost jelző futtatható flag helyén), mind az SGID (az `s` karakter a csoport futtatható flagje helyén) bitjei be vannak állítva, ami azt jelenti, hogy a fájl tulajdonosának és csoportjának jogosultságaival hajtódik végre (mindkettőnél a `daemon`). Ez az oka annak, hogy a hétköznapi felhasználók a `at` segítségével ütemezhetnek be jobokat.

107.3 Lokalizáció és internacionalizáció

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 102, Objective 107.3

Súlyozás

3

Kulcsfontosságú ismeretek

- Lokalizációs beállítások és a környezeti változók konfigurálása
- Időzóna beállítások és környezeti változók konfigurálása

A használt fájlok, kifejezések és segédprogramok listája

- `/etc/timezone`
- `/etc/localtime`
- `/usr/share/zoneinfo/`
- `LC_*`
- `LC_ALL`
- `LANG`
- `TZ`
- `/usr/bin/locale`
- `tzselect`
- `timedatectl`
- `date`
- `iconv`

- `UTF-8`
- `ISO-8859`
- ASCII
- Unicode

107.3 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	107 Adminisztrációs feladatok
Fejezet:	107.3 Lokalizáció és internacionalizáció
Lecke:	1/1

Bevezetés

Minden nagyobb Linux-disztribúció konfigurálható az egyéni lokalizációs beállítások használatára. Ezek a beállítások tartalmazzák a régióval és nyelvvel kapcsolatos meghatározásokat, mint például az időzóna, az interfész nyelve, valamint a karakterkódolás, amelyek az operációs rendszer telepítése során vagy azt követően bármikor módosíthatók.

Az alkalmazások a környezeti változókra, a rendszer konfigurációs fájljaira és a parancsokra támaszkodnak a megfelelő idő és nyelv kiválasztásában; ezért a legtöbb Linux-disztribúcióban szabványosított módon lehet beállítani az idő- és lokalizációs beállításokat. Ezek a beállítások nem csak a felhasználói élmény javítása miatt fontosak, hanem azért is, hogy a rendszeresemények időzítése - ami például a biztonsággal kapcsolatos problémák jelentéséhez fontos - helyesen legyen kiszámítva.

Ahhoz, hogy bármilyen írott szöveget képesek legyenek megjeleníteni, függetlenül a beszélt nyelvtől, a modern operációs rendszereknek szükségük van egy referencia *karakterkódolási szabványra*, és a Linux rendszerek sem különböznek ettől. Mivel a számítógépek csak számokkal tudnak bánni, a szöveges karakter nem más, mint egy grafikus szimbólummal társított szám. A különböző számítógépes platformok különböző számértékeket társíthatnak ugyanahhoz a

karakterhez, ezért a kompatibilitásukhoz szükség van egy közös karakterkódolási szabványra. Az egyik rendszerben létrehozott szöveges dokumentum csak akkor lesz olvasható egy másik rendszerben, ha mindkettő egyetért a kódolási formátumban és abban, hogy melyik karakterhez milyen szám tartozik, vagy legalábbis ha tudják, hogyan kell a két szabvány között konvertálni.

A Linux-alapú rendszerek lokalizációs beállításainak heterogén jellege finom különbségeket eredményez a disztribúciók között. E különbségek ellenére minden disztribúció ugyanazokat az alapvető eszközöket és koncepciókat használja a rendszer internacionalizálási szempontjainak beállításához.

Időzónák

Az időzónák a Föld felszínének nagyjából arányos, egy órával egyenértékű, különálló sávjai, azaz a világ azon régiói, amelyek a nap bármelyik órájában ugyanazt az időpontot élik meg. Mivel nem létezik egyetlen olyan hosszúsági fok, amely az egész világ számára a nap kezdetének tekinthető, az időzónákat a *nullmeridiánhoz* (kezdő hosszúsági kör) viszonyítják, ahol a Föld hosszúsági szöge 0. A nullmeridiánon mért időt nevezzük koordinált világidőnek (*Coordinated Universal Time*), amelyet egyezményesen UTC-nek rövidítenek. Gyakorlati okokból az időzónák nem követik pontosan a referenciaponttól (a főmeridiántól) mért hosszanti távolságot. Ehelyett az időzónákat mesterségesen úgy alakítják ki, hogy azok az országok vagy más jelentős alegységek határait kövessék.

A politikai felosztás annyira lényeges, hogy az időzónákat az adott terület valamelyik fontos földrajzi tényezőjéről nevezik el, általában az adott zónán belüli nagy ország vagy város neve alapján. Az időzónákat azonban az UTC-hez viszonyított időeltolódásuk szerint is felosztják, és ez az eltolódás az adott zóna megjelölésére is használható. Az *GMT-5* időzóna például olyan régiót jelöl, amelynek UTC-ideje öt órával előrébb van, azaz az adott régió 5 órával van az UTC mögött. Hasonlóképpen, a *GMT+3* időzóna olyan régiót jelöl, amelynek UTC ideje három órával van lemaradva. A GMT kifejezést — a *Greenwich Mean Time* kifejezésből — az UTC szinonimájaként használják az eltoláson alapuló zónanevekben.

Egy csatlakoztatott gépet a világ különböző pontjairól lehet elérni, ezért jó gyakorlat, ha a hardver óráját UTC-re (a GMT+0 időzónára) állítjuk, és az időzóna kiválasztását mindig az egyes esetekre hagyjuk. A felhőszolgáltatásokat például általában úgy konfigurálják, hogy UTC-t használjanak, mivel ez segíthet enyhíteni a helyi idő és az ügyfelek vagy más szerverek időzítése közötti alkalmi ellentmondásokat. Ezzel szemben azok a felhasználók, akik távoli munkamenetet nyitnak a szerveren, talán a saját helyi időzónájukat akarják használni. Ennek megoldása az operációs rendszeren múlik, hogy az egyes eseteknek megfelelően állítsa be a megfelelő időzónát.

Az aktuális dátum és idő mellett a `date` parancs az aktuálisan beállított időzónát is kiírja:

```
$ date
Mon Oct 21 10:45:21 -03 2019
```

A UTC-hez viszonyított eltolódást a `-03` érték adja meg, ami azt jelenti, hogy a megjelenített idő három órával kevesebb, mint a UTC. Ezért az UTC-idő három órával előrébb van, így az adott időpontnak megfelelő időzóna a *GMT-3*. A `timedatectl` parancs, amely a systemd-t használó disztribúciókban elérhető, további részleteket jelenít meg a rendszer idejéről és dátumáról:

```
$ timedatectl
                      Local time: Sat 2019-10-19 17:53:18 -03
                  Universal time: Sat 2019-10-19 20:53:18 UTC
                        RTC time: Sat 2019-10-19 20:53:18
                       Time zone: America/Sao_Paulo (-03, -0300)
       System clock synchronized: yes
systemd-timesyncd.service active: yes
                 RTC in local TZ: no
```

Amint az az `Time zone` bejegyzésben látható, a helységeken alapuló időzóna nevek — mint például az `Amerika/Sao_Paulo` — szintén elfogadottak. A rendszer alapértelmezett időzónáját az `/etc/timezone` fájlban tartjuk, a zóna teljes leírónevével vagy az időeltolódással. Az UTC-től való időeltolódással megadott általános időzóna neveknek a név első részeként tartalmazniuk kell az `Etc` szót. Tehát, ha az alapértelmezett időzónát GMT+3-ra állítjuk, az időzóna nevének `Etc/GMT+3` -nak kell lennie:

```
$ cat /etc/timezone
Etc/GMT+3
```

Bár a helységeken alapuló időzónanevek nem igénylik az időzónától való eltolódást a működéshez, nem olyan egyszerű kiválasztani a megfelelőt. Ugyanannak a zónának több neve is lehet, ami megnehezíti a megjegyzését. Ennek a problémának az enyhítésére a `tzselect` parancs egy interaktív módszert kínál, amely a felhasználót a megfelelő időzóna meghatározásához vezeti. A `tzselect` parancsnak alapértelmezés szerint minden Linux disztribúcióban elérhetőnek kell lennie, mivel a GNU C könyvtárhoz kapcsolódó szükséges segédprogramokat tartalmazó csomag tartalmazza.

A `tzselect` parancs hasznos lehet például egy olyan felhasználó számára, aki a “Brazil” területen lévő “São Paulo City” időzónáját szeretné meghatározni. A `tzselect` a kívánt hely makrorégiójának megkérdezésével kezdődik:

```
$ tzselect
Please identify a location so that time zone rules can be set correctly.
Please select a continent, ocean, "coord", or "TZ".
 1) Africa
 2) Americas
 3) Antarctica
 4) Asia
 5) Atlantic Ocean
 6) Australia
 7) Europe
 8) Indian Ocean
 9) Pacific Ocean
10) coord - I want to use geographical coordinates.
11) TZ - I want to specify the time zone using the Posix TZ format.
#? 2
```

A 2-es opció (észak- és dél-) amerikai helyszínekre vonatkozik, nem feltétlenül ugyanabban az időzónában. Lehetőség van az időzónát földrajzi koordinátákkal vagy az eltolásos jelöléssel, más néven *Posix TZ formátummal* is megadni. A következő lépés az ország kiválasztása:

```
Please select a country whose clocks agree with yours.
 1) Anguilla              19) Dominican Republic    37) Peru
 2) Antigua & Barbuda     20) Ecuador               38) Puerto Rico
 3) Argentina             21) El Salvador           39) St Barthelemy
 4) Aruba                 22) French Guiana         40) St Kitts & Nevis
 5) Bahamas               23) Greenland             41) St Lucia
 6) Barbados              24) Grenada               42) St Maarten (Dutch)
 7) Belize                25) Guadeloupe            43) St Martin (French)
 8) Bolivia               26) Guatemala             44) St Pierre & Miquelon
 9) Brazil                27) Guyana                45) St Vincent
10) Canada                28) Haiti                 46) Suriname
11) Caribbean NL          29) Honduras              47) Trinidad & Tobago
12) Cayman Islands        30) Jamaica               48) Turks & Caicos Is
13) Chile                 31) Martinique            49) United States
14) Colombia              32) Mexico                50) Uruguay
15) Costa Rica            33) Montserrat            51) Venezuela
16) Cuba                  34) Nicaragua             52) Virgin Islands (UK)
17) Curaçao               35) Panama                53) Virgin Islands (US)
18) Dominica              36) Paraguay
#? 9
```

Brazília területe négy időzónán átível, így az országinformáció önmagában nem elegendő az

időzóna beállításához. A következő lépésben a `tzselect` a helyi régió megadására kéri a felhasználót:

```
Please select one of the following time zone regions.
 1) Atlantic islands
 2) Pará (east); Amapá
 3) Brazil (northeast: MA, PI, CE, RN, PB)
 4) Pernambuco
 5) Tocantins
 6) Alagoas, Sergipe
 7) Bahia
 8) Brazil (southeast: GO, DF, MG, ES, RJ, SP, PR, SC, RS)
 9) Mato Grosso do Sul
10) Mato Grosso
11) Pará (west)
12) Rondônia
13) Roraima
14) Amazonas (east)
15) Amazonas (west)
16) Acre
#? 8
```

Nem minden helységnév áll rendelkezésre, de a legközelebbi régió kiválasztása is elegendő. A megadott információt a `tzselect` a megfelelő időzóna megjelenítéséhez használja fel:

```
The following information has been given:

        Brazil
        Brazil (southeast: GO, DF, MG, ES, RJ, SP, PR, SC, RS)

Therefore TZ='America/Sao_Paulo' will be used.
Selected time is now:   sex out 18 18:47:07 -03 2019.
Universal Time is now:  sex out 18 21:47:07 UTC 2019.
Is the above information OK?
1) Yes
2) No
#? 1

You can make this change permanent for yourself by appending the line
        TZ='America/Sao_Paulo'; export TZ
to the file '.profile' in your home directory; then log out and log in again.
```

```
Here is that TZ value again, this time on standard output so that you
can use the /usr/bin/tzselect command in shell scripts:
America/Sao_Paulo
```

Az így kapott időzóna neve, `America/Sao_Paulo`, a `/etc/timezone` állomány tartalmaként is használható a rendszer alapértelmezett időzónájának megadására:

```
$ cat /etc/timezone
America/Sao_Paulo
```

Ahogy a `tzselect` kimenete is mutatja, a `TZ` környezeti változó határozza meg a shell munkamenet időzónáját, függetlenül attól, hogy a rendszer alapértelmezett időzónája milyen. Ha a `~/.profile` fájlhoz hozzáadjuk a `TZ='America/Sao_Paulo'; export TZ` sort, akkor az `America/Sao_Paulo` lesz a felhasználó jövőbeli munkameneteinek időzónája. A `TZ` változót az aktuális munkamenet alatt ideiglenesen is módosíthatjuk, hogy egy másik időzóna idejét jelenítsük meg:

```
$ env TZ='Africa/Cairo' date
Mon Oct 21 15:45:21 EET 2019
```

A példában az `env` parancs az adott parancsot egy új sub-shell munkamenetben futtatja, az aktuális munkamenet környezeti változóival, kivéve a `TZ` változót, amelyet a `TZ='Africa/Cairo'` argumentummal módosít.

Nyári időszámítás

Sok régióban az év egy részében nyári időszámítást alkalmaznak — amikor az órákat tipikusan egy órával állítják át --, ami ahhoz vezethet, hogy egy rosszul konfigurált rendszer az évnek ebben az időszakában rossz időt jelent.

Az `/etc/localtime` fájl tartalmazza azokat az adatokat, amelyeket az operációs rendszer az óra megfelelő beállításához használ. A szabványos Linux rendszerekben az összes időzóna fájlja az `/usr/share/zoneinfo/` mappában található, így az `/etc/localtime` csak egy szimbolikus hivatkozás az ebben a mappában található tényleges adatfájlra. Az `/usr/share/zoneinfo/` állományok a megfelelő időzóna neve szerint vannak rendezve, így az `Amerika/Sao_Paulo` időzóna adatfájlja a `/usr/share/zoneinfo/Amerika/Sao_Paulo` lesz.

Mivel a nyári időszámítás definíciói változhatnak, fontos, hogy a `/usr/share/zoneinfo/` fájlokat naprakészen tartsuk. A disztribúció által biztosított csomagkezelő eszköz upgrade parancsának

frissítenie kell őket minden alkalommal, amikor egy új verzió elérhetővé válik.

Nyelv és karakterkódolás

A Linux-rendszerek számos nyelvet és nem nyugati karakterkódolást is tudnak használni, az úgynevezett *locales* meghatározásokkal. A legalapvetőbb helyi beállítás a `LANG` környezeti változó definiálása, amelyből a legtöbb shell program azonosítja a használandó nyelvet.

A `LANG` változó tartalma az `ab_CD` formátumot követi, ahol `ab` a nyelv kódja és `CD` a régió kódja. A nyelvkódnak az ISO-639 szabványt, a régió kódjának pedig az ISO-3166 szabványt kell követnie. Egy brazil portugál nyelv használatára konfigurált rendszer esetén például a `LANG` változót `pt_BR.UTF-8`-ra kell beállítanunk:

```
$ echo $LANG
pt_BR.UTF-8
```

Amint a fenti példa kimenetben látható, a `LANG` változó tartalmazza a rendszerhez tervezett karakterkódolást is. Az ASCII, az *American Standard Code for Information Interchange* rövidítése, az első széles körben használt karakterkódolási szabvány volt az elektronikus kommunikációban. Mivel azonban az ASCII az elérhető számértékek nagyon korlátozott tartományával rendelkezik, és az angol ábécén alapult, nem tartalmazza a más nyelvek által használt karaktereket, illetve a nem ábécés szimbólumok kibővített készletét. Az UTF-8 kódolás egy *Unicode szabvány* a szokásos nyugati karakterek, valamint számos más, nem hagyományos szimbólum számára. Az *Unicode Consortium*, az *Unicode szabvány* fenntartója szerint a számítógépes platformok közötti kompatibilitás biztosítása érdekében alapértelmezés szerint ezt kell alkalmazni:

> The Unicode Standard provides a unique number for every character, no matter what platform, device, application or language. It has been adopted by all modern software providers and now allows data to be transported through many different platforms, devices and applications without corruption. Support of Unicode forms the foundation for the representation of languages and symbols in all major operating systems, search engines, browsers, laptops, and smart phones — plus the Internet and World Wide Web (URLs, HTML, XML, CSS, JSON, etc.). (…) the Unicode Standard and the availability of tools supporting it are among the most significant recent global software technology trends. (Az Unicode szabvány minden karakterhez egyedi számot biztosít, függetlenül a platformtól, eszköztől, alkalmazástól vagy nyelvtől. Ezt a szabványt minden modern szoftvergyártó elfogadta, és most már lehetővé teszi, hogy az adatokat számos különböző platformon, eszközön és alkalmazáson keresztül lehessen továbbítani, anélkül, hogy azok sérülnének. Az Unicode támogatása képezi az alapját a nyelvek és szimbólumok megjelenítésének az összes főbb operációs rendszeren, keresőmotorban, böngészőben, laptopon és okostelefonon — valamint

> az interneten és a világhálón (URL-ek, HTML, XML, CSS, JSON stb.). (...) az Unicode-szabvány és az azt támogató eszközök elérhetősége a legjelentősebb legújabb globális szoftvertechnológiai trendek közé tartozik.)
>
> — The Unicode Consortium, What is Unicode?

Egyes rendszerek továbbra is használhatják az ISO által meghatározott szabványokat — például az ISO-8859-1 szabványt — a nem ASCII karakterek kódolására. Az ilyen karakterkódolási szabványokat azonban az Unicode kódolási szabványok javára el kell törölni. Minden nagyobb operációs rendszer alapértelmezés szerint az Unicode szabványt alkalmazza.

A rendszerszintű nyelvi beállítások a `/etc/locale.conf` fájlban vannak megadva. A `LANG` változót és más, a nyelvjáráshoz kapcsolódó változókat ebben a fájlban rendeljük hozzá úgy, mint egy egyszerű shell változót:

```
$ cat /etc/locale.conf
LANG=pt_BR.UTF-8
```

A felhasználók egyéni nyelvi beállításokat használhatnak a `LANG` környezeti változó átdefiniálásával. Ez alkalmazható csak az aktuális munkamenetre vagy a jövőbeli munkamenetekre is, ha az új definíciót hozzáadjuk a felhasználó Bash-profiljához a `~/.bash_profile` vagy a `~/.profile` állományban. Amíg azonban a felhasználó be nem jelentkezik, az alapértelmezett rendszer locale-t továbbra is a felhasználó független programok fogják használni, mint például a kijelzőkezelő bejelentkezési képernyője.

TIP

A *systemd* rendszermenedzsert használó rendszereken elérhető `localectl` parancs is használható a rendszer nyelvi beállításainak lekérdezésére és módosítására. Például: `localectl set-locale LANG=en_US.UTF-8`.

A `LANG` változó mellett más környezeti változók is befolyásolnak bizonyos helyi vonatkozásokat, például azt, hogy milyen pénznem szimbólumot használjunk, vagy a számok helyes ezres elválasztójelét:

`LC_COLLATE`

Az ábécé sorrend beállítása. Egyik célja, hogy meghatározza a fájlok és mappák sorrendjét.

`LC_CTYPE`

Bizonyos karakterkészletek rendszer által történő használatának beállítása. Meghatározza például, hogy mely karaktereket tekintse *nagybetűs* (uppercase) vagy *kisbetűs* (lowercase) karaktereknek.

`LC_MESSAGES`

A programüzenetek megjelenítési nyelvének (többnyire GNU programok) beállítása.

`LC_MONETARY`

A pénzegység és a pénznem formátumának beállítása.

`LC_NUMERIC`

A nem monetáris értékek numerikus formátumának beállítása. Fő célja az ezres és a tizedesvessző elválasztójelek meghatározása.

`LC_TIME`

Az idő és dátum formátumának beállítása.

`LC_PAPER`

A szabványos papírméret beállítása.

`LC_ALL`

Minden más változót felülír, beleértve a `LANG` változót is.

A `locale` parancs megjeleníti az összes definiált változót az aktuális locale konfigurációban:

```
$ locale
LANG=pt_BR.UTF-8
LC_CTYPE="pt_BR.UTF-8"
LC_NUMERIC=pt_BR.UTF-8
LC_TIME=pt_BR.UTF-8
LC_COLLATE="pt_BR.UTF-8"
LC_MONETARY=pt_BR.UTF-8
LC_MESSAGES="pt_BR.UTF-8"
LC_PAPER=pt_BR.UTF-8
LC_NAME=pt_BR.UTF-8
LC_ADDRESS=pt_BR.UTF-8
LC_TELEPHONE=pt_BR.UTF-8
LC_MEASUREMENT=pt_BR.UTF-8
LC_IDENTIFICATION=pt_BR.UTF-8
LC_ALL=
```

Az egyetlen nem definiált változó az `LC_ALL`, amely az összes többi nyelvi beállítás ideiglenes felülírására használható. A következő példa azt mutatja be, hogy a `date` parancs — egy `pt_BR.UTF-8` nyelvi beállítású rendszerben futva — hogyan módosítja a kimenetét, hogy megfeleljen az új `LC_ALL` változónak:

```
$ date
seg out 21 10:45:21 -03 2019
$ env LC_ALL=en_US.UTF-8 date
Mon Oct 21 10:45:21 -03 2019
```

Az `LC_ALL` változó módosítása a hét napjának és a hónap nevének rövidítését amerikai angol nyelven (`en_US`) jelenítette meg. Nem kötelező azonban minden változóhoz ugyanazt a nyelvi tartományt beállítani. Lehetséges például, hogy a nyelvet `pt_BR`-re, a számformátumot (`LC_NUMERIC`) pedig az amerikai szabványra állítsuk be.

Egyes lokalizációs beállítások megváltoztatják, hogy a programok hogyan kezelik az ábécésorrendet és a számformátumokat. Míg a hagyományos programok általában felkészültek arra, hogy ilyen helyzetekben helyesen válasszák ki a közös nyelvterületet, előfordulhat, hogy a scriptek váratlanul viselkednek, például amikor megpróbálnak helyesen ábécérendbe rendezni egy elemlistát. Ezért ajánlott a `LANG` környezeti változót a közös `C` nyelvterületre állítani, mint például a `LANG=C`, így a script egyértelmű eredményeket ad, függetlenül attól, hogy milyen lokalizációs definíciókat használnak a rendszerben, ahol végrehajtják. A C lokalizáció csak egy egyszerű bytewise összehasonlítást végez, így ez jobban is fog teljesíteni, mint a többi lehetőség.

Kódolási konverzió

A szöveg érthetetlen karakterekkel kerülhet ábrázolásra, ha olyan rendszeren jelenik meg, amelynek karakterkódolási konfigurációja eltér attól a rendszertől, ahol a szöveget létrehozták. Az `iconv` parancs használható ennek a problémának a megoldására, a fájl eredeti karakterkódolásáról a kívánt karakterkódolásra való átkonvertálásával. Például egy `original.txt` nevű fájl ISO-8859-1 kódolásból az UTF-8 kódolást használó `converted.txt` nevű fájlba való átalakításához a következő parancs használható:

```
$ iconv -f ISO-8859-1 -t UTF-8 original.txt > converted.txt
```

Az `-f ISO-8859-1` (vagy `--from-code=ISO-8859-1`) kapcsoló az eredeti fájl kódolását, a `-t UTF-8` (vagy `--to-code=UTF-8`) kapcsoló pedig a konvertált fájl kódolását állítja be. Az `iconv` parancs által támogatott összes kódolást az `iconv -l` vagy `iconv --list` parancs listázza. A példában szereplő kimeneti átirányítás helyett a `-o converted.txt` vagy a `--output converted.txt` opció is használható.

Gyakorló feladatok

1. A `date` parancs alábbi kimenete alapján mi a rendszer időzónája GMT jelölésben?

```
$ date
Mon Oct 21 18:45:21 +05 2019
```

2. Milyen fájlra kell mutatnia a `/etc/localtime` szimbolikus linknek ahhoz, hogy a rendszer alapértelmezett helyi ideje `Európa/Brüsszel` legyen?

3. Előfordulhat, hogy a szövegfájlokban lévő karakterek nem jelennek meg helyesen egy olyan rendszerben, amelynek karakterkódolása eltér a szöveges dokumentumban használt karakterkódolástól. Hogyan lehet az `iconv` segítségével a WINDOWS-1252 kódolású `old.txt` fájlt az UTF-8 kódolású `new.txt` fájlba konvertálni?

Gondolkodtató feladatok

1. Melyik paranccsal lesz a `Pacific/Auckland` az alapértelmezett időzóna az aktuális shell munkamenethez?

2. Az `uptime` parancs többek között a rendszer *terhelésének átlagát* mutatja meg törtszámokban. Az aktuális helyi beállítások alapján dönti el, hogy a tizedesjel pont vagy vessző legyen-e. Ha például az aktuális nyelvi beállítási terület `de_DE.UTF-8` (a németországi szabványos nyelvi terület), akkor az `uptime` vesszőt használ elválasztóként. Mivel tudjuk, hogy az amerikai angol nyelvben a pont használatos elválasztóként, milyen parancs hatására választja el az `uptime` a törtrészeket vessző helyett ponttal az aktuális munkamenet hátralévő részében?

3. Az `iconv` parancs a célkarakterkészleten kívüli összes karaktert kérdőjellel helyettesíti. Ha a `//TRANSLIT` parancs a célkódoláshoz van csatolva, akkor a célkarakterkészletben nem szereplő karaktereket egy vagy több hasonló kinézetű karakterrel helyettesíti (átírja). Hogyan lehetne ezzel a módszerrel egy `readme.txt` nevű UTF-8 szövegfájlt egy `ascii.txt` nevű egyszerű ASCII fájlba konvertálni?

Összefoglalás

Ez a lecke azt mutatja be, hogyan állíthatunk be egy Linuxot úgy, hogy egyéni nyelvi és időbeállításokkal dolgozzon. A karakterkódolási fogalmakkal és beállításokkal is foglalkozunk, mivel ezek nagyon fontosak a szöveges tartalmak helyes megjelenítéséhez. A lecke a következő témákat járja körül:

- Hogyan választják ki a Linux rendszerek a nyelvet a shell üzenetek megjelenítéséhez.
- Annak megértése, hogy az időzónák hogyan befolyásolják a helyi időt.
- Hogyan lehet azonosítani a megfelelő időzónát és ennek megfelelően módosítani a rendszerbeállításokat.
- Melyek a karakterkódolások és hogyan kell konvertálni közöttük.

Az említett parancsok és procedúrák az alábbiak voltak:

- Lokációval és idővel kapcsolatos környezeti változók, mint például az `LC_ALL`, `LANG` és `TZ`.
- `/etc/timezone`
- `/etc/localtime`
- `/usr/share/zoneinfo/`
- `locale`
- `tzselect`
- `timedatectl`
- `date`
- `iconv`

Válaszok a gyakorló feladatokra

1. A `date` parancs alábbi kimenete alapján mi a rendszer időzónája GMT jelölésben?

   ```
   $ date
   Mon Oct 21 18:45:21 +05 2019
   ```

   Az `Etc/GMT+5` az időzóna.

2. Milyen fájlra kell mutatnia a `/etc/localtime` szimbolikus linknek ahhoz, hogy a rendszer alapértelmezett helyi ideje `Európa/Brüsszel` legyen?

   Az `/etc/localtime` linknek a `/usr/share/zoneinfo/Europe/Brussels` fájlra kell mutatnia.

3. Előfordulhat, hogy a szövegfájlokban lévő karakterek nem jelennek meg helyesen egy olyan rendszerben, amelynek karakterkódolása eltér a szöveges dokumentumban használt karakterkódolástól. Hogyan lehet az `iconv` segítségével a WINDOWS-1252 kódolású `old.txt` fájlt az UTF-8 kódolású `new.txt` fájlba konvertálni?

   Az `iconv -f WINDOWS-1252 -t UTF-8 -o new.txt old.txt` parancs fogja végrehajtani a kívánt konverziót.

Válaszok a gondolkodtató feladatokra

1. Melyik paranccsal lesz a `Pacific/Auckland` az alapértelmezett időzóna az aktuális shell munkamenethez?

   `export TZ=Pacific/Auckland`

2. Az `uptime` parancs többek között a rendszer *terhelésének átlagát* mutatja meg törtszámokban. Az aktuális helyi beállítások alapján dönti el, hogy a tizedesjel pont vagy vessző legyen-e. Ha például az aktuális nyelvi beállítási terület `de_DE.UTF-8` (a németországi szabványos nyelvi terület), akkor az `uptime` vesszőt használ elválasztóként. Mivel tudjuk, hogy az amerikai angol nyelvben a pont használatos elválasztóként, milyen parancs hatására választja el az `uptime` a törtrészeket vessző helyett ponttal az aktuális munkamenet hátralévő részében?

   Az `export LC_NUMERIC=en_US.UTF-8` vagy `export LC_ALL=en_US.UTF-8` parancsok.

3. Az `iconv` parancs a célkarakterkészleten kívüli összes karaktert kérdőjellel helyettesíti. Ha a `//TRANSLIT` parancs a célkódoláshoz van csatolva, akkor a célkarakterkészletben nem szereplő karaktereket egy vagy több hasonló kinézetű karakterrel helyettesíti (átírja). Hogyan lehetne ezzel a módszerrel egy `readme.txt` nevű UTF-8 szövegfájlt egy `ascii.txt` nevű egyszerű ASCII fájlba konvertálni?

   Az `iconv -f UTF-8 -t ASCII//TRANSLIT -o ascii.txt readme.txt` parancs hajtja végre a kívánt konverziót.

Linux Professional Institute

Témakör 108: Alapvető rendszerszolgáltatások

108.1 Rendszeridő karbantartása

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 102, Objective 108.1

Súlyozás

3

Kulcsfontosságú ismeretek

- A rendszer dátumának és idejének beállítása
- A hardver órájának beállítása a megfelelő UTC-időre
- A megfelelő időzóna beállítása
- Alapvető NTP konfiguráció az ntpd és a chrony használatával
- A pool.ntp.org szolgáltatás használatának ismerete
- Az ntpq parancs ismerete

A használt fájlok, kifejezések és segédprogramok listája

- `/usr/share/zoneinfo/`
- `/etc/timezone`
- `/etc/localtime`
- `/etc/ntp.conf`
- `/etc/chrony.conf`
- `date`
- `hwclock`
- `timedatectl`

- `ntpd`
- `ntpdate`
- `chronyc`
- `pool.ntp.org`

108.1 Lecke 1

Tanúsítvány:	LPIC-1 (102)
Verzió:	5.0
Témakör:	108 Alapvető rendszerszolgáltatások
Fejezet:	108.1 Rendszeridő karbantartása
Lecke:	1/2

Bevezetés

A pontos időmérés elengedhetetlenül fontos a modern számítástechnika számára, azonban ennek megvalósítása meglepően összetett. Az időmérés gyakorlata a végfelhasználó számára triviálisnak tűnik, a rendszernek azonban számos sajátosságot és szélsőséges tényezőt kell tudnia intelligensen kezelni. Gondoljunk arra, hogy az időzónák nem statikusak, hanem adminisztratív vagy politikai döntéssel megváltoztathatók. Egy ország dönthet úgy, hogy nem tartja be a nyári időszámítást. Minden programnak képesnek kell lennie arra, hogy ezeket a változásokat logikusan kezelje. A rendszergazdák szerencséjére a Linux operációs rendszer időmérési megoldásai kiforrottak, robusztusak és általában különösebb beavatkozás nélkül működnek.

Amikor egy Linux-számítógép elindul, elkezdi az időmérést. Ezt nevezzük *rendszerórának* (system clock), mivel az operációs rendszer frissíti. A modern számítógépek emellett rendelkeznek egy *hardveres* vagy *real time clock* órával is. Ez a hardveres óra gyakran az alaplap egyik eleme, és attól függetlenül követi az időt, hogy a számítógép fut-e vagy sem. A rendszerindítás során a rendszeridő a hardveres óráról kerül beállításra, de a legtöbb esetben ez a két óra egymástól függetlenül fut. Ebben a leckében a rendszer- és a hardverórával való interakció módszereit fogjuk megvitatni.

A legtöbb modern Linux rendszerben a rendszeridő és a hardveridő a *hálózati időhöz* (network time) van szinkronizálva, amelyet a *Network Time Protocol* (NTP) valósít meg. Az esetek túlnyomó többségében a normál felhasználónak csak az időzóna beállítását kell elvégeznie, a többiről pedig az NTP gondoskodik. Azonban az idővel való manuális munka néhány módjáról is lesz szó, a hálózati idő konfigurálásának sajátosságait pedig a következő leckében tárgyaljuk.

Helyi vs univerzális idő

A rendszer órája az egyezményes koordinált világidőre (UTC) van állítva, amely az Egyesült Királyságban, Greenwichben érvényes helyi idő. A felhasználó általában a *helyi időt* szeretné tudni.A helyi idő kiszámítása úgy történik, hogy az UTC-időből veszünk egy *eltolódást* (offset) az időzóna és a nyári időszámítás alapján. Így sok bonyolultság elkerülhető.

A rendszer óráját UTC-időre vagy helyi időre is be lehet állítani, de ajánlott, hogy UTC-időre is legyen beállítva.

Dátum

A `date` egy alapvető segédprogram, amely egyszerűen kiírja a helyi időt:

```
$ date
Sun Nov 17 12:55:06 EST 2019
```

A `date` parancs kapcsolói megváltoztatják a kimenet formátumát.

A felhasználó például a `date -u` paranccsal megtekintheti az aktuális UTC-időt.

```
$ date -u
Sun Nov 17 18:02:51 UTC 2019
```

Néhány más, gyakran használt kapcsoló a helyi időt olyan formátumban adja vissza, amely megfelel egy elfogadott RFC formátumnak:

`-I`

Dátum/idő ISO 8601 formátumban. A `date` (`-Idate`) hozzáadása csak a dátumra korlátozza a kimenetet. Egyéb formátumok: `hours`, `minutes`, `seconds` és `ns`, ami a nanoszekundumot jelenti.

`-R`

RFC 5322 formátumban adja vissza a dátumot és az időt.

`--rfc-3339`

RFC 3339 formátumban adja vissza a dátumot és az időt.

A `date` formátumát a felhasználó testre szabhatja a man oldalon megadott szekvenciákkal. Például az aktuális időt Unix időként így lehet formázni:

```
$ date +%s
1574014515
```

A `date` man oldaláról láthatjuk, hogy a `%s` a Unix időre utal.

A Unix-időt a legtöbb Unix-szerű rendszer belsőleg használja. Az UTC-időt az *Epoch* óta eltelt másodpercek számaként tárolja, amely 1970. január 1.

NOTE

A Unix-idő tárolásához jelenleg 32 bit szükséges. A jövőben a 32 bit nem lesz elegendő az aktuális idő Unix formátumban történő tárolására, ez pedig komoly problémákat fog okozni minden 32 bites Linux rendszerben. Szerencsére ez nem fog bekövetkezni 2038. január 19-ig.

Ezekkel a szekvenciákkal a dátumot és az időt szinte bármilyen alkalmazás által igényelt formátumúra tudjuk formázni. Természetesen a legtöbb esetben sokkal jobb, ha ragaszkodunk egy elfogadott szabványhoz.

Ezen kívül a `date --date` használható a nem aktuális idővel megegyező időpont formázására is. Az alábbi példában a felhasználó a rendszerre alkalmazandó dátumot Unix idővel adja meg:

```
$ date --date='@1564013011'
Wed Jul 24 20:03:31 EDT 2019
```

A `--debug` kapcsoló használata nagyon hasznos lehet annak biztosítására, hogy a dátumot sikeresen lehessen elemezni. Figyeljük meg, mi történik, ha egy érvényes dátumot adunk át a parancsnak:

```
$ date --debug --date="Fri, 03 Jan 2020 14:00:17 -0500"
date: parsed day part: Fri (day ordinal=0 number=5)
date: parsed date part: (Y-M-D) 2020-01-03
date: parsed time part: 14:00:17 UTC-05
```

```
date: input timezone: parsed date/time string (-05)
date: using specified time as starting value: '14:00:17'
date: warning: day (Fri) ignored when explicit dates are given
date: starting date/time: '(Y-M-D) 2020-01-03 14:00:17 TZ=-05'
date: '(Y-M-D) 2020-01-03 14:00:17 TZ=-05' = 1578078017 epoch-seconds
date: timezone: system default
date: final: 1578078017.000000000 (epoch-seconds)
date: final: (Y-M-D) 2020-01-03 19:00:17 (UTC)
date: final: (Y-M-D) 2020-01-03 14:00:17 (UTC-05)
```

Ez hasznos eszköz lehet egy dátumot generáló alkalmazás hibaelhárításakor.

Hardveres óra

A felhasználó futtathatja az `hwclock` parancsot a valós idejű óra által fenntartott idő megtekintéséhez. Ehhez a parancshoz emelt szintű jogosultságok szükségesek, ezért ebben az esetben a `sudo` parancsot fogjuk használni a parancs meghívására:

```
$ sudo hwclock
2019-11-20 11:31:29.217627-05:00
```

A `--verbose` kapcsoló használata több kimenetet ad vissza, ami hasznos lehet hibaelhárításkor:

```
$ sudo hwclock --verbose
hwclock from util-linux 2.34
System Time: 1578079387.976029
Trying to open: /dev/rtc0
Using the rtc interface to the clock.
Assuming hardware clock is kept in UTC time.
Waiting for clock tick...
...got clock tick
Time read from Hardware Clock: 2020/01/03 19:23:08
Hw clock time : 2020/01/03 19:23:08 = 1578079388 seconds since 1969
Time since last adjustment is 1578079388 seconds
Calculated Hardware Clock drift is 0.000000 seconds
2020-01-03 14:23:07.948436-05:00
```

Figyeljük meg a `Calculated Hardware Clock drift` értéket. Ebből a kimenetből megtudhatjuk, hogy a rendszeridő és a hardveridő eltér-e egymástól.

timedatectl

A `timedatectl` egy olyan parancs, amely az idő és a dátum általános állapotának ellenőrzésére használható, beleértve azt is, hogy a hálózati idő szinkronizálva van-e vagy sem (a Network Time Protocol a következő leckében lesz tárgyalva).

Alapértelmezés szerint a `timedatectl` a `date`-hez hasonló információkat ad vissza, de kiegészítve azt az RTC (hardveres) idővel, valamint az NTP szolgáltatás állapotával:

```
$ timedatectl
               Local time: Thu 2019-12-05 11:08:05 EST
           Universal time: Thu 2019-12-05 16:08:05 UTC
                 RTC time: Thu 2019-12-05 16:08:05
                Time zone: America/Toronto (EST, -0500)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

Az idő beállítása a timedatectl segítségével

Ha az NTP nem érhető el, akkor az idő beállításához a `date` vagy a `hwclock` helyett a `timedatectl` használata ajánlott:

```
# timedatectl set-time '2011-11-25 14:00:00'
```

A folyamat hasonló a `date`-hez. A felhasználó a dátumtól független időt is beállíthat HH:MM:SS formátumban.

Az időzóna beállítása a timedatectl segítségével

`systemd` alapú Linux rendszereken a `timedatectl` a helyi időzóna beállításának preferált módja, ha nincs elérhető GUI. A `timedatectl` felsorolja a lehetséges időzónákat, majd az időzóna beállítható ezek egyikének argumentumként való felhasználásával.

Először kilistázzuk a lehetséges időzónákat:

```
$ timedatectl list-timezones
Africa/Abidjan
Africa/Accra
Africa/Algiers
```

```
Africa/Bissau
Africa/Cairo
...
```

A lehetséges időzónák listája hosszú, ezért ebben az esetben a `grep` parancs használata ajánlott.

Ezután beállíthatjuk az időzónát a visszakapott lista egyik elemével:

```
$ timedatectl set-timezone Africa/Cairo
$ timedatectl
               Local time: Thu 2019-12-05 18:18:10 EET
           Universal time: Thu 2019-12-05 16:18:10 UTC
                 RTC time: Thu 2019-12-05 16:18:10
                Time zone: Africa/Cairo (EET, +0200)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

Ne feledjük, hogy az időzóna nevének pontosnak kell lennie! Az `Africa/Cairo` például megváltoztatja az időzónát, de a `Cairo` vagy az `africa/cairo` nem!

Az NTP kikapcsolása a timedatectl segítségével

Bizonyos esetekben szükség lehet az NTP letiltására. Ezt megtehetjük a `systemctl` segítségével, de most mi ezt a `timedatectl` segítségével fogjuk bemutatni:

```
# timedatectl set-ntp no
$ timedatectl
             Local time: Thu 2019-12-05 18:19:04 EET Universal time: Thu 2019-12-05 16:19:04
UTC
               RTC time: Thu 2019-12-05 16:19:04
              Time zone: Africa/Cairo (EET, +0200)
            NTP enabled: no
       NTP synchronized: no
        RTC in local TZ: no
             DST active: n/a
```

Az időzóna beállítása a timedatectl nélkül

Az időzóna-információk beállítása a Linux új gépre történő telepítésekor szokásos lépés. Ha van grafikus telepítési folyamat, akkor ez valószínűleg minden további felhasználói beavatkozás

nélkül megtörténik.

A `/usr/share/zoneinfo` mappa tartalmazza a lehetséges különböző időzónák adatait. A `zoneinfo` mappában vannak almappák, amelyek a kontinensek neveit, valamint egyéb szimbolikus linkeket tartalmaznak. Javasoljuk, hogy a saját régiónk `zoneinfo` mappáját a saját kontinensünktől kezdve keressük meg!

A `zoneinfo` fájlok tartalmazzák a helyi időeltolódás kiszámításához szükséges szabályokat az UTC-hez képest, és akkor is fontosak, ha az aktuális régiónkban van nyári időszámítás. Az `/etc/localtime` tartalmát olvassa be a Linux, amikor meg kell határoznia a helyi időzónát. Ahhoz, hogy az időzónát a felhasználói felület használata nélkül állítsa be, a felhasználónak szimbolikus linket kell létrehoznia a `/usr/share/zoneinfo`-ból az `/etc/localtime`-be. Például:

```
$ ln -s /usr/share/zoneinfo/Canada/Eastern /etc/localtime
```

A megfelelő időzóna beállítása után javasolt az alábbit futtatni:

```
# hwclock --systohc
```

Ezzel a *hardverórát* a *rendszerórához* igazítja (azaz a valós idejű óra a `date` időponttal azonos időre lesz beállítva). Ez a parancs csak root jogosultságokkal futtatható, a fenti esetben root-ként bejelentkezve.

Az `/etc/timezone` hasonló az `/etc/localtime`-hoz. Ez a helyi időzóna adatreprezentációja, és mint ilyen, a `cat` segítségével olvasható:

```
$ cat /etc/timezone
America/Toronto
```

Vegyük figyelembe, hogy ezt a fájlt nem minden Linux disztribúció használja!

A dátum és idő beállítása a timedatectl nélkül

NOTE

A legtöbb modern Linux rendszer a `systemd`-t használja a konfigurációhoz és a szolgáltatásokhoz, ezért nem ajánlott a `date` vagy a `hwclock` használata az idő beállításához! A `systemd` erre a `timedatectl`-t használja, azonban ennek ellenére fontos ismerni ezeket a régi parancsokat arra az esetre, ha egy régebbi rendszert kell adminisztrálnunk.

A date használata

A `date` rendelkezik egy kapcsolóval a rendszeridő beállításához, ez pedig a `--set` vagy `-s`. Használhatjuk a `--debug`-ot is, hogy ellenőrizzük a parancsot:

```
# date --set="11 Nov 2011 11:11:11"
```

Vegyük figyelembe, hogy a dátum beállításához root jogosultságok szükségesek! Az időt vagy a dátumot önállóan is megváltoztathatjuk:

```
# date +%Y%m%d -s "20111125"
```

Itt meg kell adnunk a szekvenciákat, hogy a stringünk megfelelően legyen elemezve. Például a `%Y` az évre utal, így az első négy számjegyet (2011) a 2011-es évnek fogja értelmezni. Hasonlóképpen, a `%T` az időre vonatkozó szekvencia, ezt az idő beállításával mutatjuk be:

```
# date +%T -s "13:11:00"
```

A rendszeridő módosítása után ajánlott a hardverórát is beállítani, hogy a rendszer- és a hardveróra szinkronizálódjon:

```
# hwclock --systohc
```

A `systohc` a “system clock to hardware clock” rövidítése.

A hwclock használata

A rendszeróra beállítása és a hardveróra frissítése helyett megfordíthatjuk a folyamatot. A hardveróra beállításával kezdjük:

```
# hwclock --set --date "4/12/2019 11:15:19"
# hwclock
Fri 12 Apr 2019 6:15:19 AM EST -0.562862 seconds
```

Vegyük észre, hogy az `hwclock` alapértelmezés szerint UTC időt vár, de alapértelmezés szerint a helyi időt adja vissza!

A hardveróra beállítása után frissítenünk kell a rendszerórát. Az `hctosys` azt jelenti, hogy

“hardware clock to system clock”.

```
# hwclock --hctosys
```

Gyakorló feladatok

1. Jelezzük, hogy a következő parancsok a *rendszeridőt* vagy a *hardveridőt* jelenítik meg vagy módosítják:

Parancs(ok)	Rendszer	Hardver	Mindkettő
`date -u`			
`hwclock --set --date "12:00:00"`			
`timedatectl`			
`timedatectl | grep RTC`			
`hwclock --hctosys`			
`date +%T -s "08:00:00"`			
`timedatectl set-time 1980-01-10`			

2. Figyeljük meg a következő kimenetet, majd javítsuk ki az argumentum formátumát, hogy a parancs sikeres legyen:

```
$ date --debug --date "20/20/12 0:10 -3"

date: warning: value 20 has less than 4 digits. Assuming MM/DD/YY[YY]
date: parsed date part: (Y-M-D) 0002-20-20
date: parsed time part: 00:10:00 UTC-03
date: input timezone: parsed date/time string (-03)
date: using specified time as starting value: '00:10:00'
date: error: invalid date/time value:
date:     user provided time: '(Y-M-D) 0002-20-20 00:10:00 TZ=-03'
date:        normalized time: '(Y-M-D) 0003-08-20 00:10:00 TZ=-03'
date:                                 ---- --
date:      possible reasons:
date:        numeric values overflow;
date:        incorrect timezone
date: invalid date ‘20/20/2 0:10 -3’
```

3. A `date` parancs és a szekvenciák használatával állítsuk be, hogy a rendszer hónapja február legyen! A dátum és az idő többi része maradjon változatlan!

4. Feltételezve, hogy a fenti parancs sikeres volt, használjuk az `hwclock` parancsot a hardveróra beállításához a rendszeróra alapján!

5. Van egy `eucla` nevű lokáció. Melyik kontinenshez tartozik? Derítsük ki a `grep` használatával!

6. Állítsuk be az aktuális időzónát `eucla`-ra!

Gondolkodtató feladatok

1. Melyik időbeállítási módszer az optimális? Milyen forgatókönyv esetén lenne lehetetlen az előnyben részesített módszer használata?

2. Mi lehet az oka annak, hogy ennyi módszer van ugyanannak a dolognak, például a rendszeridő beállításának az elérésére?

3. 2038. január 19. után a Linux System Time 64 bites számot igényel a tároláshoz. Lehetséges azonban, hogy egyszerűen csak egy “New Epoch” beállítását választjuk. Például 2038. január 1-jén éjfélkor `0` lenne a New Epoch Time. Vajon miért nem ez a megoldás a legközkedveltebb?

Összefoglalás

Ebben a leckében megtanultuk:

- Hogyan jeleníthetjük meg az időt különböző formátumokban a parancssorból.
- A különbséget a rendszeróra és a hardveróra között Linuxban.
- Hogyan állíthatjuk manuálisan a rendszerórát.
- Hogyan állíthatjuk be manuálisan a hardverórát.
- Hogyan lehet megváltoztatni a rendszer időzónáját.

A leckében használt parancsok:

`date`
A rendszeróra megjelenítése vagy módosítása. Egyéb lehetőségek:

`-u`
UTC-idő megjelenítése.

`+%s`
Szekvencia használata az Epoch idő megjelenítéséhez.

`--date=`
Egy adott idő megjelenítése a valós idő helyett.

`--debug`
Debug üzenetek megjelenítése a felhasználó által megadott dátum elemzéséhez.

`-s`
A rendszeridő manuális beállítása.

`hwclock`
A hardveróra megjelenítése vagy módosítása.

`--systohc`
A hardveróra beállítása a rendszeróra segítségével.

`--hctosys`
A rendszeróra beállítása a hardveróra segítségével.

`--set --date`

A hardveróra manuális beállítása.

`timedatectl`

Rendszer- és hardverórák, valamint az NTP konfiguráció megjelenítése systemd-alapú Linux rendszereken.

`set-time`

Az idő manuális beállítása.

`list-timezones`

Lehetséges időzónák listázása.

`set-timezone`

Időzóna manuális beállítása.

`set-ntp`

NTP engedélyezése/tiltása.

Válaszok a gyakorló feladatokra

1. Jelezzük, hogy a következő parancsok a *rendszeridőt* vagy a *hardveridőt* jelenítik meg vagy módosítják:

Parancs(ok)	Rendszer	Hardver	Mindkettő
`date -u`	X		
`hwclock --set --date "12:00:00"`		X	
`timedatectl`			X
`timedatectl | grep RTC`		X	
`hwclock --hctosys`	X		
`date +%T -s "08:00:00"`	X		
`timedatectl set-time 1980-01-10`			X

2. Figyeljük meg a következő kimenetet, majd javítsuk ki az argumentum formátumát, hogy a parancs sikeres legyen:

```
$ date --debug --date "20/20/12 0:10 -3"

date: warning: value 20 has less than 4 digits. Assuming MM/DD/YY[YY]
date: parsed date part: (Y-M-D) 0002-20-20
date: parsed time part: 00:10:00 UTC-03
date: input timezone: parsed date/time string (-03)
date: using specified time as starting value: '00:10:00'
date: error: invalid date/time value:
date:     user provided time: '(Y-M-D) 0002-20-20 00:10:00 TZ=-03'
date:        normalized time: '(Y-M-D) 0003-08-20 00:10:00 TZ=-03'
date:                                 ---- --
date:      possible reasons:
date:        numeric values overflow;
date:        incorrect timezone
date: invalid date ‘20/20/2 0:10 -3’
```

`date --debug --set "12/20/20 0:10 -3"`

3. A `date` parancs és a szekvenciák használatával állítsuk be, hogy a rendszer hónapja február legyen! A dátum és az idő többi része maradjon változatlan!

   ```
   date +%m -s "2"
   ```

4. Feltételezve, hogy a fenti parancs sikeres volt, használjuk az `hwclock` parancsot a hardveróra beállításához a rendszeróra alapján!

   ```
   hwclock -systohc
   ```

5. Van egy `eucla` nevű lokáció. Melyik kontinenshez tartozik? Derítsük ki a `grep` használatával!

   ```
   timedatectl list-timezones \| grep -i eucla
   ```

   VAGY

   ```
   grep -ri eucla /usr/share/zoneinfo
   ```

6. Állítsuk be az aktuális időzónát `eucla`-ra!.

   ```
   timedatectl set-timezone 'Australia/Eucla'
   ```

   VAGY

   ```
   ln -s /usr/share/zoneinfo/Australia/Eucla /etc/localtime
   ```

Válaszok a gondolkodtató feladatokra

1. Melyik időbeállítási módszer az optimális? Milyen forgatókönyv esetén lenne lehetetlen az előnyben részesített módszer használata?

   A legtöbb Linux-disztribúcióban az NTP alapértelmezés szerint engedélyezve van, és hagyni kell, hogy mindenféle beavatkozás nélkül állítsa be a rendszeridőt. Ha azonban van olyan Linux rendszer, amely nincs csatlakoztatva az internethez, az NTP nem lesz elérhető. Például egy ipari berendezésen futó beágyazott Linux rendszer nem biztos, hogy rendelkezik hálózati kapcsolattal.

2. Mi lehet az oka annak, hogy ennyi módszer van ugyanannak a dolognak, például a rendszeridő beállításának az elérésére?

   Mivel az idő beállítása évtizedek óta az összes *nix rendszer követelménye, az idő beállítására számos régi módszer létezik, amelyeket még mindig fenntartanak.

3. 2038. január 19. után a Linux System Time 64 bites számot igényel a tároláshoz. Lehetséges azonban, hogy egyszerűen csak egy “New Epoch” beállítását választjuk. Például 2038. január 1-jén éjfélkor `0` lenne a New Epoch Time. Vajon miért nem ez a megoldás a legközkedveltebb?

   2038-ra a számítógépek túlnyomó többsége már 64 bites CPU-t fog használni, és a 64 bites szám használata nem fogja jelentősen rontani a teljesítményt. Lehetetlen lenne azonban megbecsülni az Epoch time “nullázásának” kockázatát. Rengeteg olyan legacy szoftver van, amelyre ez hatással lehet. A bankok és nagyvállalatok például gyakran nagy mennyiségű régebbi programmal rendelkeznek, amelyek csak belső használatra vannak. Ez a forgatókönyv, sok máshoz hasonlóan, a kompromisszumok tanulmányozása. A 2038-ban még futó 32 bites rendszerekre hatással lenne az Epoch Time túlcsordulása, de az Epoch értékének megváltoztatásával a legacy szoftverekre is hatással lenne.

108.1 Lecke 2

Tanúsítvány:	LPIC-1 (102)
Verzió:	5.0
Témakör:	108 Alapvető rendszerszolgáltatások
Fejezet:	108.1 Rendszeridő karbantartása
Lecke:	2/2

Bevezetés

Míg a személyi számítógépek önmagukban is képesek viszonylag pontos időt tartani, a termelési számítógépek és a hálózati környezetek megkövetelik, hogy az idő nagyon pontos legyen. A legpontosabb időt *referenciaórák* mérik, amelyek jellemzően atomórák. A modern világ kidolgozott egy olyan rendszert, amelyben minden internetre csatlakoztatott számítógépes rendszer szinkronizálható ezekkel a referenciaórákkal az úgynevezett *Network Time Protocol* (NTP) segítségével. Az NTP-vel rendelkező számítógépes rendszer képes lesz szinkronizálni a rendszerórákat a referenciaórák által megadott időhöz. Ha a rendszeridő és a szervereken mért idő eltér egymástól, akkor a számítógép fokozatosan gyorsítja vagy lassítja a belső rendszeridejét, amíg a rendszeridő meg nem egyezik a hálózati idővel.

Az NTP hierarchikus struktúrát használ az idő terjesztésére. A referenciaórák a hierarchia tetején lévő szerverekhez kapcsolódnak. Ezek a szerverek *Stratum 1* gépek, és általában nem nyilvánosak. Az 1. rétegbeli gépek azonban elérhetők a *Stratum 2* gépek számára, amelyek elérhetők a *Stratum 3* gépek számára, és így tovább. A Stratum 2 szerverek a nyilvánosság számára hozzáférhetők, akárcsak a hierarchiában lejjebb elhelyezkedő gépek. Ha egy nagy hálózat NTP-jét állítjuk be, jó gyakorlat, ha kisszámú számítógép csatlakozik a Stratum 2+ szerverekhez, és ezek a gépek biztosítják az NTP-t az összes többi gép számára. Ily módon a Stratum 2-es gépekkel szemben

támasztott követelmények minimálisra csökkenthetők.

Van néhány fontos kifejezés, amely az NTP megvitatásakor felmerül. Ezek közül néhányat azokban a parancsokban használunk, amelyekkel nyomon követhetjük az NTP állapotát a gépeinken:

Offset

Ez a rendszeridő és az NTP-idő közötti abszolút különbségre utal. Ha például a rendszeróra 12:00:02-t mutat, az NTP-idő pedig 11:59:58-at, akkor a két óra közötti eltolódás (offset) négy másodperc.

Step

Ha az NTP-szolgáltató és a fogyasztó közötti időeltolódás nagyobb, mint 128 ms, akkor az NTP a rendszeridő lassítása vagy gyorsítása helyett egyetlen jelentős változást hajt végre a rendszeridőben. Ezt nevezzük *stepping*-nek.

Slew

A slewing a rendszeridőben bekövetkező változásokra utal, amikor a rendszeridő és az NTP közötti eltolódás kisebb, mint 128 ms. Ha ez a helyzet, akkor a módosítások fokozatosan történnek. Ezt nevezzük *slewing*-nek.

Insane Time

Ha a rendszeridő és az NTP-idő közötti eltolódás nagyobb, mint 17 perc, akkor a rendszeridő *insane*-nek (őrült) tekinthető, és az NTP-daemon nem fog változtatni a rendszeridőn. Különleges lépéseket kell tenni annak érdekében, hogy a rendszeridő 17 percen belül legyen a megfelelő időhöz képest.

Drift

A drift azt a jelenséget jelenti, amikor két óra idővel nem szinkronizálódik. Lényegében, ha két óra kezdetben szinkronban van, de az idő múlásával eltolódik, akkor a clock drift jelenséggel találkozunk.

Jitter

A jitter az órajel utolsó lekérdezése óta bekövetkezett eltérés mértékére utal. Tehát ha az utolsó NTP-szinkronizálás 17 perccel ezelőtt történt, és az NTP szolgáltató és a fogyasztó közötti eltolódás 3 milliszekundum, akkor 3 milliszekundum a jitter.

Most a Linux által az NTP megvalósításának néhány konkrét módját fogjuk megvitatni.

timedatectl

Ha a Linux disztribúció a `timedatectl`-t használja, akkor alapértelmezés szerint egy *SNTP* klienst valósít meg, nem pedig egy teljes NTP implementációt. Ez a hálózati idő kevésbé bonyolult megvalósítása, és azt jelenti, hogy a számítógépünk nem fog NTP-t szolgáltatni más csatlakoztatott számítógépeknek.

Ebben az esetben az SNTP csak akkor fog működni, ha a `timesyncd` szolgáltatás fut. Mint minden systemd szolgáltatás esetében, a futását a következővel tudjuk ellenőrizni:

```
$ systemctl status systemd-timesyncd
● systemd-timesyncd.service - Network Time Synchronization
   Loaded: loaded (/lib/systemd/system/systemd-timesyncd.service; enabled; vendor preset:
enabled)
  Drop-In: /lib/systemd/system/systemd-timesyncd.service.d
           └─disable-with-time-daemon.conf
   Active: active (running) since Thu 2020-01-09 21:01:50 EST; 2 weeks 1 days ago
     Docs: man:systemd-timesyncd.service(8)
 Main PID: 1032 (systemd-timesyn)
   Status: "Synchronized to time server for the first time 91.189.89.198:123
(ntp.ubuntu.com)."
    Tasks: 2 (limit: 4915)
   Memory: 3.0M
   CGroup: /system.slice/systemd-timesyncd.service
           └─1032 /lib/systemd/systemd-timesyncd

Jan 11 13:06:18 NeoMex systemd-timesyncd[1032]: Synchronized to time server for the first
time 91.189.91.157:123 (ntp.ubuntu.com).
...
```

A `timedatectl` SNTP szinkronizálás állapota a `show-timesync` használatával ellenőrizhető:

```
$ timedatectl show-timesync --all
LinkNTPServers=
SystemNTPServers=
FallbackNTPServers=ntp.ubuntu.com
ServerName=ntp.ubuntu.com
ServerAddress=91.189.89.198
RootDistanceMaxUSec=5s
PollIntervalMinUSec=32s
PollIntervalMaxUSec=34min 8s
PollIntervalUSec=34min 8s
```

```
NTPMessage={ Leap=0, Version=4, Mode=4, Stratum=2, Precision=-23, RootDelay=8.270ms,
RootDispersion=18.432ms, Reference=91EECB0E, OriginateTimestamp=Sat 2020-01-25 18:35:49 EST,
ReceiveTimestamp=Sat 2020-01-25 18:35:49 EST, TransmitTimestamp=Sat 2020-01-25 18:35:49 EST,
DestinationTimestamp=Sat 2020-01-25 18:35:49 EST, Ignored=no PacketCount=263, Jitter=2.751ms
}
Frequency=-211336
```

Ez a konfiguráció a legtöbb helyzetben megfelelő lehet, de mint már említettük, nem lesz elégséges, ha egy hálózaton belül több klienst szeretnénk szinkronizálni. Ebben az esetben ajánlott egy teljes körű NTP-kliens telepítése.

NTP Daemon

A rendszeridő rendszeres időközönként összehasonlításra kerül a hálózati idővel. Ahhoz, hogy ez működjön, szükségünk van egy *daemon*-ra, amely a háttérben fut. Sok Linux rendszerben ennek a a neve `ntpd`. Az `ntpd` lehetővé teszi, hogy egy gép ne csak *time consumer* (időfogyasztó) legyen (azaz képes legyen szinkronizálni a saját óráját egy külső forrásból), hanem *szolgáltasson* is időt más gépeknek.

Tegyük fel, hogy a számítógépünk systemd-alapú, és a `systemctl`-t használja a daemonok vezérlésére! Telepítsük az `ntp` csomagokat a megfelelő csomagkezelő segítségével, majd az állapotának ellenőrzésével győződjünk meg arról, hogy az `ntpd` démonunk fut:

```
$ systemctl status ntpd

●  ntpd.service - Network Time Service
   Loaded: loaded (/usr/lib/systemd/system/ntpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2019-12-06 03:27:21 EST; 7h ago
  Process: 856 ExecStart=/usr/sbin/ntpd -u ntp:ntp $OPTIONS (code=exited, status=0/SUCCESS)
 Main PID: 867 (ntpd)
   CGroup: /system.slice/ntpd.service
           `-867 /usr/sbin/ntpd -u ntp:ntp -g
```

Bizonyos esetekben szükség lehet az `ntpd` indítására és engedélyezésére is. A legtöbb Linux gépen ez a következővel érhető el:

```
# systemctl enable ntpd && systemctl start ntpd
```

Az NTP lekérdezések a 123-as TCP porton történnek. Ha az NTP nem működik, győződjünk meg róla, hogy ez a port nyitva van és hallgat!

NTP konfiguráció

Az NTP képes több forrást lekérdezni, és kiválasztani a legjobb jelölteket a rendszeridő beállításához. Ha a hálózati kapcsolat megszakad, az NTP a korábbi beállításokat használja az előzményekből a jövőbeli beállítások becsléséhez.

A Linux disztribúciótól függően a hálózati időszerverek listája különböző helyeken lesz tárolva. Tegyük fel, hogy az `ntp` telepítve van!

Az `/etc/ntp.conf` fájl tartalmazza a rendszer hálózati idővel való szinkronizálásának módjára vonatkozó konfigurációs információkat. Ez a fájl a `vi` vagy a `nano` segítségével olvasható és módosítható.

Alapértelmezés szerint a használt NTP-szerverek egy ilyen szakaszban lesznek megadva:

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
server 0.centos.pool.ntp.org iburst
server 1.centos.pool.ntp.org iburst
server 2.centos.pool.ntp.org iburst
server 3.centos.pool.ntp.org iburst
```

Az NTP-szerver hozzáadásának szintaxisa így néz ki:

```
server (IP Address)
server server.url.localhost
```

A szerverek címei lehetnek IP- vagy akár URL-címek is, ha a DNS megfelelően van konfigurálva. Ebben az esetben a szerver mindig lekérdezésre kerül.

A hálózati rendszergazda fontolóra veheti egy *pool* használatát (vagy létrehozását) is. Ebben az esetben feltételezzük, hogy több NTP-szolgáltató van, amelyek mindegyike NTP-daemont futtat és azonos idővel rendelkezik. Amikor egy kliens lekérdez egy pool-t, véletlenszerűen választ ki egy szolgáltatót. Ez segít elosztani a hálózati terhelést sok gép között, hogy a poolban ne egyetlen gép kezelje az összes NTP-lekérdezést.

Általában az `/etc/ntp.conf` egy `pool.ntp.org` nevű szerverpoolt tartalmaz. Így például a `server 0.centos.pool.ntp.org` a CentOS gépek számára biztosított alapértelmezett NTP-pool.

pool.ntp.org

Az alapértelmezés szerint használt NTP-szerverek nyílt forráskódú projektek. További információ a ntppool.org oldalon található.

> Consider if the NTP Pool is appropriate for your use. If business, organization or human life depends on having correct time or can be harmed by it being wrong, you shouldn't "just get it off the internet". The NTP Pool is generally very high quality, but it is a service run by volunteers in their spare time. Please talk to your equipment and service vendors about getting local and reliable service setup for you. See also our terms of service. We recommend time servers from Meinberg, but you can also find time servers from End Run, Spectracom and many others. (Mérlegeljük, hogy az NTP-pool megfelel-e az igényeinknek. Ha üzlet, szervezet vagy emberi élet függ a helyes időtől, vagy ha a hibás idő kárt okozhat, akkor nem szabad "csak úgy lekérni az internetről". Az NTP Pool általában nagyon jó minőségű, de ez egy önkéntesek által szabadidejükben működtetett szolgáltatás. Kérjük, beszéljen a berendezés- és szervizszolgáltatókkal a helyi és megbízható szolgáltatás beállításáról. Lsd. még a szolgáltatási feltételeinket. Mi a Meinberg időszervereit ajánljuk, de az End Run, a Spectracom és sok más cég időszerverei is megtalálhatók.)
>
> — ntppool.org

ntpdate

A kezdeti beállítás során a rendszeridő és az NTP szinkronizációja komolyan megszűnhet. Ha a rendszeridő és az NTP-idő közötti *eltolódás* nagyobb, mint 17 perc, akkor az NTP-daemon nem fog változtatni a rendszeridőn. Ebben az esetben kézi beavatkozásra lesz szükség.

Először is, ha az `ntpd` fut, akkor *le kell állítani* a szolgáltatást. Ehhez használjuk a `systemctl stop ntpd` parancsot!

Ezután az `ntpdate pool.ntp.org` paranccsal végezzünk el egy kezdeti egyszeri szinkronizálást, ahol a `pool.ntp.org` egy NTP-szerver IP-címére vagy URL-címére utal! Egynél több szinkronizálásra is szükség lehet.

ntpq

Az `ntpq` egy segédprogram az NTP állapotának megfigyelésére. Miután az NTP-daemont elindítottuk és konfiguráltuk, az `ntpq` segítségével ellenőrizhetjük az állapotát:

```
$ ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
```

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
+37.44.185.42    91.189.94.4      3 u   86  128  377  126.509  -20.398   6.838
+ntp2.0x00.lv    193.204.114.233  2 u   82  128  377  143.885   -8.105   8.478
*inspektor-vlan1 121.131.112.137  2 u   17  128  377  112.878  -23.619   7.959
 b1-66er.matrix. 18.26.4.105      2 u  484  128   10   34.907   -0.811  16.123
```

Ebben az esetben a `-p` a *print*-et jelenti, és egy összefoglalást jelenít meg a peerekről. A hostcímek IP-címként is visszaadhatók az `-n` használatával.

`remote`

Az NTP-szolgáltató hostneve.

`refid`

Az NTP-szolgáltató referencia ID-ja.

`st`

A szolgáltató rétege (stratum).

`when`

Az utolsó lekérdezés óta eltelt másodpercek száma.

`poll`

A lekérdezések közötti másodpercek száma.

`reach`

Állapotazonosító, amely jelzi, hogy sikerült-e elérni egy szervert. A sikeres kapcsolatok ezt a számot 1-gyel növelik.

`delay`

A lekérdezés és a szerver által adott válasz közötti idő ms-ban.

`offset`

A rendszeridő és az NTP-idő közti idő ms-ban.

`jitter`

A rendszeridő és az NTP közötti eltolódás ms-ban az utolsó lekérdezéskor.

Az `ntpq` rendelkezik egy interaktív móddal is, amely akkor érhető el, ha kapcsolók vagy argumentumok nélkül futtatjuk. A `?` kapcsolóval azon parancsok listáját kapjuk vissza, amelyeket az `ntpq` felismer.

chrony

A `chrony` egy újabb módja az NTP implementációjának. Egyes Linux rendszereken alapértelmezés szerint telepítve van, de minden nagyobb disztribúcióhoz letölthető. A `chronyd` a chrony-daemon, a `chronyc` pedig a parancssori interfész. Előfordulhat, hogy a `chronyc`-vel való interakció előtt el kell indítani és engedélyezni kell a `chronyd`-t.

Ha a chronyt alapértelmezett konfigurációval telepítjük, akkor a `chronyc tracking` parancs használatával információt kaphatunk az NTP-ről és a rendszeridőről:

```
$ chronyc tracking
Reference ID    : 3265FB3D (bras-vprn-toroon2638w-lp130-11-50-101-251-61.dsl.)
Stratum         : 3
Ref time (UTC)  : Thu Jan 09 19:18:35 2020
System time     : 0.000134029 seconds fast of NTP time
Last offset     : +0.000166506 seconds
RMS offset      : 0.000470712 seconds
Frequency       : 919.818 ppm slow
Residual freq   : +0.078 ppm
Skew            : 0.555 ppm
Root delay      : 0.006151616 seconds
Root dispersion : 0.010947504 seconds
Update interval : 129.8 seconds
Leap status     : Normal
```

Ez a kimenet rengeteg információt tartalmaz, többet, mint ami más implementációkból elérhető.

`Reference ID`
: Az a referenciaazonosító és név, amellyel a számítógép jelenleg szinkronizálva van.

`Stratum`
: Az ugrások (hop) száma a csatlakoztatott referenciaórával rendelkező számítógéphez.

`Ref time`
: Ez az az UTC-idő, amikor az utolsó mérés a referenciaforrásból megtörtént.

`System time`
: A rendszeróra késése a szinkronizált szervertől.

`Last offset`
: Az utolsó órafrissítés becsült eltolódása.

`RMS offset`

Az eltolódás hosszú távú átlaga.

`Frequency`

Ez az a sebesség, amellyel a rendszer órája tévedne, ha a chronyd nem korrigálná azt. Ezt az értéket ppm-ben (parts per million) látjuk.

`Residual freq`

Maradékfrekvencia, amely a referenciaforrásból származó mérések és a jelenleg használt frekvencia közötti különbséget jelzi.

`Skew`

A frekvencia becsült hibahatára.

`Root delay`

A hálózati útvonal késleltetésének összege a rétegszámítógépig, ahonnan a számítógépet szinkronizálják.

`Leap status`

Ez az ugrási állapot, amely a következő értékek egyike lehet: normál, másodperc beszúrása, másodperc törlése vagy nem szinkronizált.

A legutóbbi érvényes NTP-frissítésre vonatkozó részletes információkat is megtekinthetjük:

```
# chrony ntpdata
Remote address  : 172.105.97.111 (AC69616F)
Remote port     : 123
Local address   : 192.168.122.81 (C0A87A51)
Leap status     : Normal
Version         : 4
Mode            : Server
Stratum         : 2
Poll interval   : 6 (64 seconds)
Precision       : -25 (0.000000030 seconds)
Root delay      : 0.000381 seconds
Root dispersion : 0.000092 seconds
Reference ID    : 61B7CE58 ()
Reference time  : Mon Jan 13 21:50:03 2020
Offset          : +0.000491960 seconds
Peer delay      : 0.004312567 seconds
Peer dispersion : 0.000000068 seconds
Response time   : 0.000037078 seconds
```

```
Jitter asymmetry: +0.00
NTP tests       : 111 111 1111
Interleaved     : No
Authenticated   : No
TX timestamping : Daemon
RX timestamping : Kernel
Total TX        : 15
Total RX        : 15
Total valid RX  : 15
```

Végül a `chronyc sources` az idő szinkronizálásához használt NTP-szerverekről ad információt:

```
$ chronyc sources
210 Number of sources = 0
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
```

Jelenleg ezen a gépen nincsenek konfigurálva források. A chrony konfigurációs fájl megnyitásával hozzáadhatunk forrásokat a `pool.ntp.org` címről. Ez általában a `/etc/chrony.conf` címen található. Ha megnyitjuk ezt a fájlt, látnunk kell, hogy néhány szerver alapértelmezés szerint szerepel a listában:

```
210 Number of sources = 0
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
# Most computers using chrony will send measurement requests to one or
# more 'NTP servers'.  You will probably find that your Internet Service
# Provider or company have one or more NTP servers that you can specify.
# Failing that, there are a lot of public NTP servers.  There is a list
# you can access at http://support.ntp.org/bin/view/Servers/WebHome or
# you can use servers from the 3.arch.pool.ntp.org project.

! server 0.arch.pool.ntp.org iburst iburst
! server 1.arch.pool.ntp.org iburst iburst
! server 2.arch.pool.ntp.org iburst iburst

! pool 3.arch.pool.ntp.org iburst
```

Ezek a szerverek a saját szervereinkbe való belépéskor szintaxis-útmutatóként is szolgálnak. Ebben az esetben azonban egyszerűen csak eltávolítjuk a `!`-t minden sor elejéről, így megszüntetve a kikommentezést, és a `pool.ntp.org` projekt alapértelmezett szervereit

használjuk.

Ebben a fájlban megváltoztathatjuk továbbá az alapértelmezett konfigurációt a skew és a drift tekintetében, valamint a driftfile és a keyfile helyét.

Ezen a gépen nagy kezdeti órajelkorrekciót kell végeznünk. Úgy döntünk, hogy a következő sort nem kommentezzük ki:

```
! makestep 1.0 3
```

A konfigurációs fájl módosításai után indítsuk újra a `chronyd` szolgáltatást, majd a `chronyc makestep` segítségével manuálisan állítsuk át a rendszerórát:

```
# chronyc makestep
200 OK
```

Ezután a korábbiakhoz hasonlóan használhatjuk a `chronyc tracking`-t, hogy ellenőrizzük megtörténtek-e a változások.

Gyakorló feladatok

1. Írjuk le a megfelelő kifejezést az egyes definíciókhoz:

Definíció	**Kifejezés**
Egy számítógép, amely megosztja velünk a hálózati időt	
Távolság a referenciaórától, ugrásban (hop) vagy lépésben (step)	
A rendszeridő és a hálózati idő közötti különbség	
A rendszeridő és a hálózati idő közötti különbség az utolsó NTP-lekérdezés óta	
A hálózati időt szolgáltató és a terhelést egymás között megosztó szerverek csoportja	

2. Adjuk meg, hogy melyik parancsot használnánk a következő értékek kimenetéhez:

Érték	`chronyc tracking`	`timedatectl show-timesync --all`	`ntpq -pn`	`chrony ntpdata`	`chronyc sources`
Jitter					
Drift					
Interval of Poll					
Offset					
Stratum					
IP Address of Provider					
Root Delay					

3. Egy Linux szerverből és több Linux asztali számítógépből álló vállalati hálózatot hozunk létre. A szerver statikus IP-címe 192.168.0.101. Úgy döntünk, hogy a szerver csatlakozik a `pool.ntp.org` címre, majd NTP-időt biztosít az asztali számítógépek számára. Írjuk le a szerver és az asztali számítógépek konfigurációját!

4. A Linux gépen a helytelen idő van. Írjuk le az NTP hibaelhárításának lépéseit!

Gondolkodtató feladatok

1. Kutassuk fel az SNTP és az NTP közötti különbségeket!

SNTP	NTP

2. Miért dönthet úgy egy rendszergazda, hogy *nem* használja a `pool.ntp.org` címet?

3. Hogyan tudná eldönteni egy rendszergazda, hogy csatlakozna vagy más módon járulna hozzá a `pool.ntp.org` projekthez?

Összefoglalás

Ebben a leckében megtanultuk:

- Mi az NTP és miért fontos.
- Az NTP-daemon konfigurálása a `pool.ntp.org` projektből.
- Az `ntpq` használata az NTP konfiguráció ellenőrzésére.
- A `chrony` használata NTP alternatívaként.

A leckében használt parancsok:

`timedatectl show-timesync --all`
SNTP információk megjelenítése a `timedatectl` használata esetén.

`ntpdate <address>`
Manuális, egyszeri NTP step-frissítés (step update).

`ntpq -p`
Az NTP legutóbbi poll előzményeinek megjelenítése. Az `-n` az URL-címeket IP-címekkel helyettesíti.

`chronyc tracking`
Az NTP állapotának megjelenítése chrony használata esetén.

`chronyc ntpdata`
NTP-információk megjelenítése a legutóbbi pollról.

`chronyc sources`
NTP-szolgáltatókra vonatkozó információk megjelenítése.

`chronyc makestep`
Manuális, egyszeri NTP-step frissítés chrony használata esetén.

Válaszok a gyakorló feladatokra

1. Írjuk le a megfelelő kifejezést az egyes definíciókhoz:

Definíció	**Kifejezés**
Egy számítógép, amely megosztja velünk a hálózati időt	Provider (szolgáltató)
Távolság a referenciaórától, ugrásban (hop) vagy lépésben (step)	Stratum
A rendszeridő és a hálózati idő közötti különbség	Offset (eltolódás)
A rendszeridő és a hálózati idő közötti különbség az utolsó NTP-lekérdezés óta	Jitter
A hálózati időt szolgáltató és a terhelést egymás között megosztó szerverek csoportja	Pool

2. Adjuk meg, hogy melyik parancsot használnánk a következő értékek kimenetéhez:

Érték	`chronyc tracking`	`timedatectl show-timesync --all`	`ntpq -pn`	`chrony ntpdata`	`chronyc sources`
Jitter		X	X		
Drift					
Interval of Poll	X	X	X (`when` oszlop)	X	X
Offset	X		X	X	
Stratum	X	X	X	X	X
IP Address of Provider		X	X	X	X
Root Delay	X			X	

3. Egy Linux szerverből és több Linux asztali számítógépből álló vállalati hálózatot hozunk létre. A szerver statikus IP-címe 192.168.0.101. Úgy döntünk, hogy a szerver csatlakozik a `pool.ntp.org` címre, majd NTP-időt biztosít az asztali számítógépek számára. Írjuk le a

szerver és az asztali számítógépek konfigurációját!

Győződjünk meg róla, hogy a szerveren az SNTP helyett az ntpd szolgáltatás fut! Használjuk a `pool.ntp.org` poolokat az `/etc/ntp.conf` vagy `/etc/chrony.conf` fájlban! Minden kliens esetében adjuk meg a `192.168.0.101` címet minden `/etc/ntp.conf` vagy `/etc/chrony.conf` fájlban!

4. A Linux gépen a helytelen idő van. Írjuk le az NTP hibaelhárításának lépéseit!

   Először is győződjünk meg arról, hogy a gép csatlakozik az internethez! Ehhez használjuk a `ping` parancsot! Ellenőrizzük, hogy az ntpd vagy SNTP szolgáltatás fut-e a `systemctl status ntpd` vagy a `systemctl status systemd-timesyncd` segítségével! Előfordulhat, hogy hibaüzenetek jelennek meg, amelyek hasznos információkkal szolgálnak. Végül, egy olyan paranccsal, mint az `ntpq -p` vagy a `chrony tracking` ellenőrizzük, hogy történt-e kérés. Ha a rendszeridő drasztikusan eltér a hálózati időtől, előfordulhat, hogy a rendszeridő “insane”-nek minősül, és kézi beavatkozás nélkül nem módosítható. Ebben az esetben használjuk az előző leckében leírt parancsot vagy egy olyan parancsot, mint az `ntpdate pool.ntp.org`, hogy egyszeri ntp szinkronizálást hajtsunk végre!

Válaszok a gondolkodtató feladatokra

1. Kutassuk fel az SNTP és az NTP közötti különbségeket!

SNTP	**NTP**
kevésbé pontos	pontosabb
kevesebb erőforrást igényel	több erőforrást igényel
nem tud időszolgáltatóként működni	tud időszolgáltatóként működni
csak steps	steps vagy slews
egyetlen forrásból kéri az időt	több NTP-szervert is figyelhet, és az optimális szolgáltatót használhatja

2. Miért dönthet úgy egy rendszergazda, hogy *nem* használja a `pool.ntp.org` címet?

   Az ntppool.org oldalról: Ha feltétlenül fontos, hogy pontos idő legyen, akkor meg kell fontolni az alternatívákat. Hasonlóképpen, ha az internetszolgáltatónk rendelkezik időszerverrel, akkor ajánlott azt használni helyette.

3. Hogyan tudná eldönteni egy rendszergazda, hogy csatlakozna vagy más módon járulna hozzá a `pool.ntp.org` projekthez?

   A www.ntppool.org oldalról: A szervernek statikus IP-címmel és állandó internetkapcsolattal kell rendelkeznie. A statikus IP-cím egyáltalán nem, vagy évente legfeljebb egyszer változhat. Ezen túlmenően a sávszélességi követelmények szerények: 384 - 512 Kbit sávszélesség. Stratum 3 vagy 4 szerverek csatlakozhatnak.

108.2 Rendszerlogolás

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 102, Objective 108.2

Súlyozás

4

Kulcsfontosságú ismeretek

- Az rsyslog alapvető konfigurációja
- A szabványos eszközök, prioritások és műveletek megértése
- A systemd journal lekérdezése
- A systemd journal adatainak szűrése olyan kritériumok alapján, mint a dátum, a szolgáltatás vagy a prioritás
- A tartós systemd journal tárolásának és a journal méretének beállítása
- Régi systemd journal adatok törlése
- A systemd journal adatainak visszakeresése mentésből vagy fájlrendszer-másolatból
- Az rsyslog és a systemd-journald interakciójának megértése
- A logrotate konfigurálása
- A syslog és a syslog-ng ismerete

A használt fájlok, kifejezések és segédprogramok listája

- `/etc/rsyslog.conf`
- `/var/log/`
- `logger`
- `logrotate`

- `/etc/logrotate.conf`
- `/etc/logrotate.d/`
- `journalctl`
- `systemd-cat`
- `/etc/systemd/journald.conf`
- `/var/log/journal/`

108.2 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	108 Alapvető rendszerszolgáltatások
Fejezet:	108.2 Rendszernaplózás
Lecke:	1/2

Bevezetés

A logok (rendszernaplók) lehetnek a rendszergazda legjobb barátai. A logok olyan, általában szöveges fájlok, amelyekben a rendszer és a hálózat minden eseménye időrendben szerepel a rendszer indításának pillanatától kezdve. A logokban található információk köre gyakorlatilag a rendszer minden aspektusára kiterjed: sikertelen hitelesítési kísérletek, program- és szolgáltatáshibák, a tűzfal által blokkolt hostok stb. Sejthető tehát, hogy a logok nagyban megkönnyítik a rendszergazdák életét a hibaelhárítás, az erőforrások ellenőrzése, a programok rendellenes viselkedésének észlelése stb. terén.

Ebben a leckében a GNU/Linux disztribúciókban jelenleg megtalálható egyik legelterjedtebb logolási lehetőséget fogjuk megvitatni: `rsyslog`. Megvizsgáljuk a különböző létező logtípusokat, hogy hol vannak tárolva, milyen információkat tartalmaznak, és hogyan lehet ezeket az információkat megszerezni és szűrni. Megvitatjuk a logrotációt és a kernel gyűrűpufferét (kernel ring buffer), valamint azt, hogy hogyan lehet a logokat IP-hálózatokon keresztül, központi szervereken tárolni.

Rendszernaplózás

Abban a pillanatban, hogy a kernel és a rendszer különböző folyamatai elkezdenek futni és kommunikálni egymással, rengeteg információ keletkezik üzenetek formájában, amelyeket — többnyire — a logokba küldenek.

Logolás nélkül a rendszergazdáknak fejfájást okozna egy szerveren történt esemény megkeresése, ezért fontos, hogy szabványosított és központosított módon lehessen nyomon követni a rendszereseményeket. A logok hibaelhárítás és biztonság szempontjából meghatározóak és sokatmondóak, valamint megbízható adatforrások a rendszerstatisztikák megértéséhez és a trendjósláshoz.

Ha eltekintünk a `systemd-journald`-tól (amiről a következő leckében lesz szó), a logolást hagyományosan három fő dedikált szolgáltatás kezeli: `syslog`, `syslog-ng` (syslog new generation) és az `rsyslog` ("the rocket-fast system for log processing" (a rakétasebességű rendszer a logok feldolgozásához)). Az `rsyslog` fontos fejlesztéseket hozott (például RELP-támogatást), és napjainkra a legnépszerűbbé vált. Mindegyik szolgáltatás összegyűjti a más szolgáltatásoktól és programoktól érkező üzeneteket és azokat logfájlokban tárolja, jellemzően a `/var/log` alatt. Egyes szolgáltatások azonban saját logjaikról is gondoskodnak (vegyük például az Apache HTTPD webszervert vagy a CUPS nyomtatórendszert). Hasonlóképpen, a Linux kernel egy memórián belüli gyűrűpuffert használ a logüzenetek tárolására.

NOTE A `RELP` a *Reliable Event Logging Protocol* rövidítése, és a syslog protokoll funkcióit bővíti az üzenetek megbízható kézbesítése érdekében.

Mivel az `rsyslog` lett a *de facto* szabványos logolási lehetőség minden nagyobb disztribúcióban, ezért ebben a leckében erre fogunk koncentrálni. Az `rsyslog` kliens-szerver modellt használ. A kliens és a szerver lehet ugyanazon a gépen vagy különböző gépeken. Az üzenetek egy meghatározott formátumban kerülnek küldésre és fogadásra, és az IP-hálózatokon keresztül központosított `rsyslog` szervereken tárolhatók. Az rsyslog daemonja — `rsyslogd` — együtt dolgozik a `klogd`-vel (amely a kernel üzeneteket kezeli). A következő részekben az `rsyslog`-ról és a logolási infrastruktúrájáról lesz szó.

NOTE A daemon egy olyan szolgáltatás, ami a háttérben fut. Vegyük észre a `d` betűt a daemonok neveiben: `klogd` vagy `rsyslogd`.

Logtípusok

Mivel a logok *változó* (variable) adatok, alapesetben a `/var/log`-ban találhatók meg. Nagyjából *rendszerlogokra* (system logs) és *szolgáltatási vagy programlogokra* (service or program logs) oszthatjuk őket.

Lássunk néhány rendszerlogot és az általuk tárolt információkat:

`/var/log/auth.log`

Autentikációs folyamatokkal kapcsolatos tevékenységek: bejelentkezett felhasználók, `sudo` információk, cronjobok, sikertelen bejelentkezési kísérletek, stb.

`/var/log/syslog`

Gyakorlatilag az összes, az `rsyslogd` által rögzített log központi fájlja. Mivel nagyon sok információt tartalmaz, a logfájlok a `/etc/rsyslog.conf` állományban megadott konfigurációnak megfelelően más állományok között oszlanak meg.

`/var/log/debug`

Programok debug információi.

`/var/log/kern.log`

Kernel üzenetek.

`/var/log/messages`

Informatív üzenetek, amelyek nem a kernelhez, hanem más szolgáltatásokhoz kapcsolódnak. Ez az alapértelmezett remote klienslog célpontja is egy központosított log-szerver implementációjában.

`/var/log/daemon.log`

Daemonokhoz vagy egyéb, háttérben futó szolgáltatásokhoz kapcsolódó információk.

`/var/log/mail.log`

Az e-mail szerverhez kapcsolódó információk, pl. postfix.

`/var/log/Xorg.0.log`

A videokártyához kapcsolódó információk.

`/var/run/utmp` and `/var/log/wtmp`

Sikeres bejelentkezések.

`/var/log/btmp`

Sikertelen bejelentkezési kísérletek, például brute force támadás SSH-n keresztül.

`/var/log/faillog`

Sikertelen autentikációs kísérletek.

`/var/log/lastlog`

A legutóbbi felhasználói bejelentkezések dátuma és időpontja.

Lássunk néhány példát a szolgáltatások logjaira:

`/var/log/cups/`

A_Common Unix Printing System_ jogjainak mappája. Általában a következő alapértelmezett logfájlokat tartalmazza: `error_log`, `page_log` és `access_log`.

`/var/log/apache2/` or `/var/log/httpd`

Az *Apache Web Server* logjainak mappája. Általában a következő alapértelmezett logfájlokat tartalmazza: `access.log`, `error_log`, és `other_vhosts_access.log`.

`/var/log/mysql`

A *MySQL Relational Database Management System* logjainak mappája. Általában a következő alapértelmezett logfájlokat tartalmazza: `error_log`, `mysql.log` és `mysql-slow.log`.

`/var/log/samba/`

A *Session Message Block* (SMB) protokoll logjaiinak mappája. Általában a következő alapértelmezett logfájlokat tartalmazza: `log.`, `log.nmbd` és `log.smbd`.

NOTE

A logfájlok pontos neve és tartalma Linux-disztribúcióról Linux-disztribúcióra változhat. Léteznek bizonyos disztribúciókra jellemző logfájlok is, mint például a Debian GNU/Linux és az ebből származó változatok esetében a `/var/log/dpkg.log` (amely az `dpkg` csomagokkal kapcsolatos információkat tartalmazza).

Logok olvasása

A logfájlok olvasásához először meg kell győződnünk arról, hogy root felhasználóként vagyunk bejelentkezve, vagy olvasási jogosultságunk van a fájlon. Számos segédprogramot használhatunk, mint például:

`less` or `more`

Lapozók, amelyek egyszerre egy oldal megtekintését és görgetését teszik lehetővé:

```
root@debian:~# less /var/log/auth.log
Sep 12 18:47:56 debian sshd[441]: Received SIGHUP; restarting.
Sep 12 18:47:56 debian sshd[441]: Server listening on 0.0.0.0 port 22.
Sep 12 18:47:56 debian sshd[441]: Server listening on :: port 22.
Sep 12 18:47:56 debian sshd[441]: Received SIGHUP; restarting.
```

```
Sep 12 18:47:56 debian sshd[441]: Server listening on 0.0.0.0 port 22.
Sep 12 18:47:56 debian sshd[441]: Server listening on :: port 22.
Sep 12 18:49:46 debian sshd[905]: Accepted password for carol from 192.168.1.65 port
44296 ssh2
Sep 12 18:49:46 debian sshd[905]: pam_unix(sshd:session): session opened for user carol
by (uid=0)
Sep 12 18:49:46 debian systemd-logind[331]: New session 2 of user carol.
Sep 12 18:49:46 debian systemd: pam_unix(systemd-user:session): session opened for user
carol by (uid=0)
(...)
```

`zless` or `zmore`

Ugyanaz, mint a `less` és a `more`, de a `gzip`-pel tömörített logok esetére (a `logrotate` egyik funkciója):

```
root@debian:~# zless /var/log/auth.log.3.gz
Aug 19 20:05:57 debian sudo:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/sbin/shutdown -h now
Aug 19 20:05:57 debian sudo: pam_unix(sudo:session): session opened for user root by
carol(uid=0)
Aug 19 20:05:57 debian lightdm: pam_unix(lightdm-greeter:session): session closed for
user lightdm
Aug 19 23:50:49 debian systemd-logind[333]: Watching system buttons on /dev/input/event2
(Power Button)
Aug 19 23:50:49 debian systemd-logind[333]: Watching system buttons on /dev/input/event3
(Sleep Button)
Aug 19 23:50:49 debian systemd-logind[333]: Watching system buttons on /dev/input/event4
(Video Bus)
Aug 19 23:50:49 debian systemd-logind[333]: New seat seat0.
Aug 19 23:50:49 debian sshd[409]: Server listening on 0.0.0.0 port 22.
(...)
```

`tail`

A fájl utolsó sorainak megtekintése (az alapértelmezett érték 10 sor). A tail ereje — nagyrészt — az `-f` kapcsolóban rejlik, amely dinamikusan mutatja az új sorokat, amint azok hozzáadásra kerülnek:

```
root@suse-server:~# tail -f /var/log/messages
2019-09-14T13:57:28.962780+02:00 suse-server sudo: pam_unix(sudo:session): session closed
for user root
2019-09-14T13:57:38.038298+02:00 suse-server sudo:    carol : TTY=pts/0 ; PWD=/home/carol
```

```
; USER=root ; COMMAND=/usr/bin/tail -f /var/log/messages
2019-09-14T13:57:38.039927+02:00 suse-server sudo: pam_unix(sudo:session): session opened
for user root by carol(uid=0)
2019-09-14T14:07:22+02:00 debian carol: appending new message from client to remote
server...
```

`head`

Egy fájl első sorainak megtekintésére szolgál (az alapértelmezés 10 sor):

```
root@suse-server:~# head -5 /var/log/mail
2019-06-29T11:47:59.219806+02:00 suse-server postfix/postfix-script[1732]: the Postfix
mail system is not running
2019-06-29T11:48:01.355361+02:00 suse-server postfix/postfix-script[1925]: starting the
Postfix mail system
2019-06-29T11:48:01.391128+02:00 suse-server postfix/master[1930]: daemon started --
version 3.3.1, configuration /etc/postfix
2019-06-29T11:55:39.247462+02:00 suse-server postfix/postfix-script[3364]: stopping the
Postfix mail system
2019-06-29T11:55:39.249375+02:00 suse-server postfix/master[1930]: terminating on signal
15
```

`grep`

Filterezésre szolgáló segédprogram, amely lehetővé teszi bizonyos stringek keresését:

```
root@debian:~# grep "dhclient" /var/log/syslog
Sep 13 11:58:48 debian dhclient[448]: DHCPREQUEST of 192.168.1.4 on enp0s3 to 192.168.1.1
port 67
Sep 13 11:58:49 debian dhclient[448]: DHCPACK of 192.168.1.4 from 192.168.1.1
Sep 13 11:58:49 debian dhclient[448]: bound to 192.168.1.4 -- renewal in 1368 seconds.
(...)
```

Amint azt már észrevehettük, a kimenet a következő formátumban jelenik meg:

- Időbélyeg
- Hostnév, ahonnan az üzenet származik
- Az üzenetet generáló program/szolgáltatás neve
- Az üzenetet generáló program PID azonosítója
- A végrehajtott művelet leírása

Van néhány példa, amikor a logok nem szöveges, hanem bináris fájlok, és — következésképpen — speciális parancsokat kell használnunk az elemzésükhöz:

`/var/log/wtmp`

`who` (vagy `w`) használata:

```
root@debian:~# who
root    pts/0        2020-09-14 13:05 (192.168.1.75)
root    pts/1        2020-09-14 13:43 (192.168.1.75)
```

`/var/log/btmp`

`utmpdump` vagy `last -f` használata:

```
root@debian:~# utmpdump /var/log/btmp
Utmp dump of /var/log/btmp
[6] [01287] [    ] [dave     ] [ssh:notty   ] [192.168.1.75        ] [192.168.1.75   ]
[2019-09-07T19:33:32,000000+0000]
```

`/var/log/faillog`

`faillog` használata:

```
root@debian:~# faillog -a | less
Login       Failures Maximum Latest                   On

root            0        0   01/01/70 01:00:00 +0100
daemon          0        0   01/01/70 01:00:00 +0100
bin             0        0   01/01/70 01:00:00 +0100
sys             0        0   01/01/70 01:00:00 +0100
sync            0        0   01/01/70 01:00:00 +0100
games           0        0   01/01/70 01:00:00 +0100
man             0        0   01/01/70 01:00:00 +0100
lp              0        0   01/01/70 01:00:00 +0100
mail            0        0   01/01/70 01:00:00 +0100
(...)
```

`/var/log/lastlog`

`lastlog` használata:

```
root@debian:~# lastlog | less
```

```
Username         Port     From             Latest
root                                       Never logged in
daemon                                     Never logged in
bin                                        Never logged in
sys                                        Never logged in
(...)
sync                                       Never logged in
avahi                                      Never logged in
colord                                     Never logged in
saned                                      Never logged in
hplip                                      Never logged in
carol            pts/1    192.168.1.75     Sat Sep 14 13:43:06 +0200 2019
dave             pts/3    192.168.1.75     Mon Sep  2 14:22:08 +0200 2019
```

NOTE Vannak grafikus eszközök is a logfájlok olvasására, például: `gnome-logs` és `KSystemLog`.

Hogyan lesznek az üzenetekből logok?

A következő folyamat azt szemlélteti, hogyan íródik egy üzenet a logba:

1. Az alkalmazások, a szolgáltatások és a kernel üzeneteket írnak speciális fájlokba (socketek és memóriapufferek), például a `/dev/log` vagy a `/dev/kmsg` fájlokba.
2. A `rsyslogd` megkapja az információ a socketekből vagy a memóriapufferekből.
3. Az `/etc/rsyslog.conf` állományban és/vagy az `/etc/ryslog.d/` állományokban található szabályoktól függően az `rsyslogd` a megfelelő logfájlba (jellemzően a `/var/log` fájlba) továbbítja az információt.

NOTE A socket egy speciális fájl, amelyet a különböző folyamatok közötti információátvitelre használnak. A rendszerben lévő összes socket listázásához a `systemctl list-sockets --all` parancsot használhatjuk.

Eszközök, prioritások és intézkedések

Az `rsyslog` konfigurációs fájl az `/etc/rsylog.conf` (néhány disztribúcióban a `/etc/rsyslog.d/` állományban is találhatunk konfigurációs fájlokat). Általában három részre van osztva: `MODULES`, `GLOBAL DIRECTIVES` és `RULES`. Nézzük meg ezeket a Debian GNU/Linux 10 (buster) gépünk `rsyslog.conf` fájljának feltárásával — ehhez használhatjuk a `sudo less /etc/rsyslog.conf` parancsot!

A `MODULES` tartalmazza a logolás, az üzenetküldési képesség és az UDP/TCP logfogadás modul-

támogatását:

```
#################
#### MODULES ####
#################

module(load="imuxsock") # provides support for local system logging
module(load="imklog")   # provides kernel logging support
#module(load="immark")  # provides --MARK-- message capability

# provides UDP syslog reception
#module(load="imudp")
#input(type="imudp" port="514")

# provides TCP syslog reception
#module(load="imtcp")
#input(type="imtcp" port="514")
```

A `GLOBAL DIRECTIVES` lehetővé teszi számunkra, hogy számos dolgot konfiguráljunk, például a logok és a logmappák jogosultságait:

```
###########################
#### GLOBAL DIRECTIVES ####
###########################

#
# Use traditional timestamp format.
# To enable high precision timestamps, comment out the following line.
#
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

#
# Set the default permissions for all log files.
#
$FileOwner root
$FileGroup adm
$FileCreateMode 0640
$DirCreateMode 0755
$Umask 0022

#
# Where to place spool and state files
#
```

```
$WorkDirectory /var/spool/rsyslog

#
# Include all config files in /etc/rsyslog.d/
#
$IncludeConfig /etc/rsyslog.d/*.conf
```

A `RULES` az, ahol az *eszközök* (facilities), a *prioritások* (priorities) és a *intézkedések* (actions) megjelennek. Az ebben a szakaszban található beállítások utasítják a logolási daemont, hogy bizonyos szabályok szerint szűrje az üzeneteket, és szükség esetén logolja vagy küldje el azokat. Ahhoz, hogy megértsük ezeket a szabályokat, először is el kell magyaráznunk az `rsyslog` eszközök és prioritások fogalmát. Minden logüzenet kap egy *facility* számot és egy kulcsszót, amelyek az üzenetet előállító Linux belső alrendszeréhez kapcsolódnak:

Szám	Kulcsszó	Leírás
`0`	`kern`	Linux kernel üzenetk
`1`	`user`	Felhasználói szintű üzenetek
`2`	`mail`	Mail rendszer
`3`	`daemon`	System daemonok
`4`	`auth`, `authpriv`	Biztonsági/Autorizációs üzenetek
`5`	`syslog`	syslogd üzenetek
`6`	`lpr`	Line printer alrendszer
`7`	`news`	Hálózati hírek alrendszer
`8`	`uucp`	UUCP (Unix-to-Unix Copy Protocol) alrendszer
`9`	`cron`	Óra daemon
`10`	`auth`, `authpriv`	Biztonsági/Autorizációs üzenetek
`11`	`ftp`	FTP (File Transfer Protocol) daemon
`12`	`ntp`	NTP (Network Time Protocol) daemon
`13`	`security`	Log audit

Szám	Kulcsszó	Leírás
`14`	`console`	Log alert
`15`	`cron`	Óra daemon
`16 - 23`	`local0` - `local7`	Helyi használat 0 - 7

Továbbá minden üzenetnek van egy *prioritási* szintje:

Kód	Súlyosság (severity)	Kulcsszó	Leírás
`0`	Vészhelyzet (emergency)	`emerg`, `panic`	A rendszer használhatatlan.
`1`	Riasztás (alert)	`alert`	Azonnali cselekvés szükséges
`2`	Kritikus (critical)	`crit`	Kritikus állapotok
`3`	Hiba (error)	`err`, `error`	Hibahelyzet.
`4`	Figyelmeztetés (warning)	`warn`, `warning`	Figyelmeztető állapotok
`5`	Tájékoztatás (notice)	`notice`	Normál, de szignifikáns állapotok
`6`	Informális (informational)	`info`	Tájékoztató üzenetek
`7`	Hibakeresés (debug)	`debug`	Hibakeresési szintű üzenetek

Íme egy részlet az `rsyslog.conf` állományból a Debian GNU/Linux 10 (buster) rendszerünkből, amely tartalmaz néhány mintaszabályt:

```
###############
#### RULES ####
###############

# First some standard log files.  Log by facility.
#
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
#cron.*                         /var/log/cron.log
daemon.*                        -/var/log/daemon.log
kern.*                          -/var/log/kern.log
```

```
lpr.*                           -/var/log/lpr.log
mail.*                          -/var/log/mail.log
user.*                          -/var/log/user.log

#
# Logging for the mail system.  Split it up so that
# it is easy to write scripts to parse these files.
#
mail.info                       -/var/log/mail.info
mail.warn                       -/var/log/mail.warn
mail.err                        /var/log/mail.err

#
# Some "catch-all" log files.
#
*.=debug;\
        auth,authpriv.none;\
    news.none;mail.none     -/var/log/debug
*.=info;*.=notice;*.=warn;\
    auth,authpriv.none;\
    cron,daemon.none;\
    mail,news.none          -/var/log/messages
```

A szabály formátuma a következő: `<facility>.<priority>` `<action>`

A `<facility>.<priority>` szelektor kiszűri a megfelelő üzeneteket. A prioritási szintek hierarchikusan inkluzívak, ami azt jelenti, hogy az rsyslog a megadott prioritású és magasabb prioritású üzeneteket fogja megtalálni. Az `<action>` megadja, hogy milyen műveletet kell végrehajtani (hova küldje a logüzenetet). Az egyértelműség kedvéért íme néhány példa:

```
auth,authpriv.*                 /var/log/auth.log
```

A prioritástól (`*`) függetlenül, az `auth` vagy `authpriv` eszközök minden üzenete a `/var/log/auth.log` mappába kerül.

```
*.*;auth,authpriv.none          -/var/log/syslog
```

Minden üzenet — prioritástól (`*`) függetlenül — minden eszköztől (`*`) — az `auth` vagy `authpriv` üzenetek kivételével (ezért a `.none` utótag) — a `/var/log/syslog` mappába íródik (az útvonal előtti mínusz jel (`-`) megakadályozza a túlzott lemezírást). Figyeljük meg a pontosvesszőt (`;`) a választó kettéválasztására és a vesszőt (`,`) két lehetőség egy szabályban való összekapcsolására

(`auth,authpriv`).

```
mail.err                        /var/log/mail.err
```

A `mail` szolgáltatásból érkező, `error` vagy magasabb prioritási szintű (`critical`, `alert` vagy `emergency`) üzenetek a `var/log/mail.err`-be kerülnek.

```
*.=debug;\
        auth,authpriv.none;\
    news.none;mail.none     -/var/log/debug
```

Az összes eszközből származó `debug` prioritású és semmilyen más (`=`) üzenetek a `/var/log/debug` mappába kerülnek — kivéve az `auth`, `authpriv`, `news` és `mail` szolgáltatásból származó üzeneteket (vegyük észre a szintaxist: `;\`).

Manuális bejegyzések a rendszerlogba: `logger`

A `logger` parancs jól jön shell scriptek készítéséhez vagy tesztelési célokra. A `logger` minden kapott üzenetet a `/var/log/syslog` állományhoz csatol (vagy a `/var/log/messages` állományba, ha egy távoli központi logszerverre (central log server) logolunk, ahogy azt a lecke későbbi részében látni fogjuk):

```
carol@debian:~$ logger this comment goes into "/var/log/syslog"
```

A `/var/log/syslog` utolsó sorának kiíratásához használjuk a `tail` parancsot a `-1` kapcsolóval:

```
root@debian:~# tail -1 /var/log/syslog
Sep 17 17:55:33 debian carol: this comment goes into /var/log/syslog
```

`rsyslog`, mint Central Log Server

A téma elmagyarázásához egy új host hozzáadásával fogjuk bővíteni a beállításainkat. Az elrendezés a következő:

Szerep	Hostnév	OS	IP Cím
Central Log Server	`suse-server`	openSUSE Leap 15.1	192.168.1.6

Szerep	Hostnév	OS	IP Cím
Client	`debian`	Debian GNU/Linux 10 (buster)	192.168.1.4

Kezdjük a szerver konfigurálásával. Először is győződjünk meg róla, hogy az `rsyslog` működik:

```
root@suse-server:~# systemctl status rsyslog
 rsyslog.service - System Logging Service
   Loaded: loaded (/usr/lib/systemd/system/rsyslog.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2019-09-17 18:45:58 CEST; 7min ago
     Docs: man:rsyslogd(8)
           http://www.rsyslog.com/doc/
 Main PID: 832 (rsyslogd)
    Tasks: 5 (limit: 4915)
   CGroup: /system.slice/rsyslog.service
           └─832 /usr/sbin/rsyslogd -n -iNONE
```

Az openSUSE a távoli logoláshoz külön konfigurációs fájlt tartalmaz: `/etc/rsyslog.d/remote.conf`. Engedélyezzük a TCP-n keresztül érkező üzenetek fogadását a kliensektől (távoli hostoktól). Vissza kell vonnunk a kikommentezést a modul betöltését és a TCP-szerver elindítását végző sorokból az 514-es porton:

```
# ######### Receiving Messages from Remote Hosts ##########
# TCP Syslog Server:
# provides TCP syslog reception and GSS-API (if compiled to support it)
$ModLoad imtcp.so  # load module
##$UDPServerAddress 10.10.0.1  # force to listen on this IP only
$InputTCPServerRun 514  # Starts a TCP server on selected port

# UDP Syslog Server:
#$ModLoad imudp.so  # provides UDP syslog reception
##$UDPServerAddress 10.10.0.1  # force to listen on this IP only
#$UDPServerRun 514  # start a UDP syslog server at standard port 514
```

Ha ez megtörtént, újra kell indítanunk az rsyslog szolgáltatást, és ellenőriznünk kell, hogy a szerver az 514-es porton figyel-e:

```
root@suse-server:~# systemctl restart rsyslog
root@suse-server:~# netstat -nltp | grep 514
[sudo] password for root:
```

```
tcp        0      0 0.0.0.0:514             0.0.0.0:*               LISTEN
2263/rsyslogd
tcp6       0      0 :::514                  :::*                    LISTEN
2263/rsyslogd
```

Ezután nyissuk meg a portokat a tűzfalon, és töltsük újra a konfigurációt:

```
root@suse-server:~# firewall-cmd --permanent --add-port 514/tcp
success
root@suse-server:~# firewall-cmd --reload
success
```

NOTE Az openSUSE Leap 15.0 megjelenésével a `firewalld` teljesen felváltotta a klasszikus `SuSEFirewall2`-t.

Sablonok és szűrési feltételek

Alapértelmezés szerint a kliens logjai a szerver `/var/log/messages` fájljába kerülnek — a szerver logjaival együtt. Létrehozunk azonban egy *sablont* (template) és egy *szűrési feltételt* (filter condition), hogy a kliens logjait saját, egyértelmű mappákban tároljuk. Ehhez a következőket kell hozzáadnunk az `/etc/rsyslog.conf` (vagy `/etc/rsyslog.d/remote.conf`) állományhoz:

```
$template RemoteLogs,"/var/log/remotehosts/%HOSTNAME%/%$NOW%.%syslogseverity-text%.log"
if $FROMHOST-IP=='192.168.1.4' then ?RemoteLogs
& stop
```

Template

A sablon az első sornak felel meg, és lehetővé teszi a lognevek formátumának megadását a dinamikus fájlnévgenerálással. A sablon a következőkből áll:

- Sablon utasítás (`$template`)
- Sablon neve (`RemoteLogs`)
- Sablon szövege (`"/var/log/remotehosts/%HOSTNAME%/%$NOW%.%syslogseverity-text%.log"`)
- Kapcsolók (opcionális)

A sablonunk neve `RemoteLogs`, a szövege pedig a `/var/log` elérési útvonala. A távoli hostunk összes logja a `remotehosts` mappába kerül, ahol a gép hostneve (`%HOSTNAME%`) alapján létrehozunk egy almappát. Ebben a mappában minden egyes fájlnév a dátumot (`%$NOW%`), az

üzenet súlyosságát (más néven prioritását) szöveges formátumban (`%syslogseverity-text%`) és a `.log` utótagot tartalmazza. A százalékjelek közötti szavak a *tulajdonságok* (properties), és lehetővé teszik a logüzenet tartalmának (dátum, prioritás stb.) elérését. A `syslog` üzenetnek számos jól definiált tulajdonsága van, amelyek sablonokban használhatók. Ezekhez a tulajdonságokhoz az úgynevezett *property replacer* segítségével férhetünk hozzá — és módosíthatjuk őket --, ami azt jelenti, hogy a százalékjelek közé írjuk őket.

Filter Condition

A fennmaradó két sor a szűrőfeltételnek és a hozzá tartozó műveletnek felel meg:

- Expression-Based Filter (`if $FROMHOST-IP=='192.168.1.4'`)
- Action (`then ?RemoteLogs`, `& stop`)

Az első sor ellenőrzi a logot küldő távoli host IP-címét, és — ha ez megegyezik a Debian kliensünkével — alkalmazza a `RemoteLogs` sablont. Az utolsó sor (`& stop`) garantálja, hogy az üzenetek nem kerülnek egyszerre a `/var/log/messages` mappába (hanem csak a `/var/log/remotehosts` mappában lévő állományokba).

NOTE A sablonokról, tulajdonságokról és szabályokról többet megtudhatunk az `rsyslog.conf` man oldaláról.

A frissített konfigurációval újraindítjuk az `rsyslog` rendszert, és megerősítjük, hogy a `/var/log` mappában még nincs `remotehosts` mappa:

```
root@suse-server:~# systemctl restart rsyslog
root@suse-server:~# ls /var/log/
acpid             chrony     localmessages   pbl.log          Xorg.0.log
alternatives.log  cups       mail            pk_backend_zypp  Xorg.0.log.old
apparmor          firebird   mail.err        samba            YaST2
audit             firewall   mail.info       snapper.log      zypp
boot.log          firewalld  mail.warn       tallylog         zypper.log
boot.msg          krb5       messages        tuned
boot.omsg         lastlog    mysql           warn
btmp              lightdm    NetworkManager  wtmp
```

A szerver most már konfigurálva van. Most pedig konfiguráljuk a klienst!

Ismét meg kell győződnünk arról, hogy az `rsyslog` telepítve van és fut:

```
root@debian:~# sudo systemctl status rsyslog
 rsyslog.service - System Logging Service
```

```
   Loaded: loaded (/lib/systemd/system/rsyslog.service; enabled; vendor preset:
   Active: active (running) since Thu 2019-09-17 18:47:54 CEST; 7min ago
     Docs: man:rsyslogd(8)
           http://www.rsyslog.com/doc/
 Main PID: 351 (rsyslogd)
    Tasks: 4 (limit: 4915)
   CGroup: /system.slice/rsyslog.service
           └─351 /usr/sbin/rsyslogd -n
```

A példánkban a névfeloldást a kliensen az `/etc/hosts` állományba a `192.168.1.6 suse-server` sor hozzáadásával valósítottuk meg. Így a szerverre vagy névvel (`suse-server`) vagy IP-címmel (`192.168.1.6`) hivatkozhatunk.

A mi Debian kliensünkben nincs meg az `/etc/rsyslog.d/` állományban található `remote.conf` fájl, ezért a konfigurációnkat az `/etc/rsyslog.conf` állományban fogjuk alkalmazni. A következő sort írjuk a fájl végére:

```
*.* @@suse-server:514
```

Végül újraindítjuk az `rsyslog`-ot.

```
root@debian:~# systemctl restart rsyslog
```

Most menjünk vissza a `suse-server` gépünkre, és ellenőrizzük a `/var/log`-ban a `remotehosts` létezését:

```
root@suse-server:~# ls /var/log/remotehosts/debian/
2019-09-17.info.log  2019-09-17.notice.log
```

Már van két log a `/var/log/remotehosts'-en belül, ahogyan azt a sablonunkban leírtuk. A szakasz befejezéséhez futtassuk a `tail -f 2019-09-17.notice.log` parancsot a `suse-server`-en, miközben *manuálisan* elküldünk egy logot a Debian kliensünkről, és megerősítjük, hogy az üzenetek a várt módon kerülnek a logfájlba (a `-t` kapcsoló egy címkét (tag) ad az üzenetünkhöz):

```
root@suse-server:~# tail -f /var/log/remotehosts/debian/2019-09-17.notice.log
2019-09-17T20:57:42+02:00 debian dbus[323]: [system] Successfully activated service
'org.freedesktop.nm_dispatcher'
2019-09-17T21:01:41+02:00 debian anacron[1766]: Anacron 2.3 started on 2019-09-17
```

```
2019-09-17T21:01:41+02:00 debian anacron[1766]: Normal exit (0 jobs run)
```

```
carol@debian:~$ logger -t DEBIAN-CLIENT Hi from 192.168.1.4
```

```
root@suse-server:~# tail -f /var/log/remotehosts/debian/2019-09-17.notice.log
2019-09-17T20:57:42+02:00 debian dbus[323]: [system] Successfully activated service
'org.freedesktop.nm_dispatcher'
2019-09-17T21:01:41+02:00 debian anacron[1766]: Anacron 2.3 started on 2019-09-17
2019-09-17T21:01:41+02:00 debian anacron[1766]: Normal exit (0 jobs run)
2019-09-17T21:04:21+02:00 debian DEBIAN-CLIENT: Hi from 192.168.1.4
```

A logrotáció mechanizmusa

A logok rendszeres rotálódnak, ami két fő célt szolgál:

- Megakadályozza, hogy a régebbi logfájlok a szükségesnél több helyet foglaljanak a lemezen.
- Kezelhető hosszúságúan tartja a logot, hogy megkönnyítse a megtekintést.

A logok rotációjáért (vagy ciklikusságáért) felelős segédprogram a `logrotate`, és olyan műveleteket végez, mint a logfájlok új néven való áthelyezése, archiválása és/vagy tömörítése, néha e-mailben történő elküldése a rendszergazdának, és végül, elöregedésük esetén, a törlésük. Különböző konvenciók léteznek ezeknek a rotált logfájloknak a megnevezésére (például a dátumot tartalmazó utótag hozzáadása a fájlnévhez); azonban az általános gyakorlat az, hogy egyszerűen egy egész számot tartalmazó utótagot adunk hozzá:

```
root@debian:~# ls /var/log/messages*
/var/log/messages  /var/log/messages.1  /var/log/messages.2.gz  /var/log/messages.3.gz
/var/log/messages.4.gz
```

Most nézzük, mi fog történni a következő logrotáció során:

1. A `messages.4.gz` törlődik és elvész.
2. A `messages.3.gz` tartalma átkerül a `messages.4.gz`-be.
3. A `messages.2.gz` tartalma átkerül a `messages.3.gz`-be.
4. A `messages.1` tartalma átkerül a `messages.2.gz`-be.
5. A `messages` tartalma átkerül a `messages.1`-be, és a `messages` üres lesz és készen áll az új logbejegyzések regisztrálására.

Figyeljük meg, hogy — a `logrotate` utasítások szerint, amelyeket hamarosan látni fogunk — a három régebbi logfájl tömörítve van, míg a két legújabb nem. Továbbá, az elmúlt 4-5 hét logfájljait is megtartjuk. Az 1 hetes üzenetek olvasásához a `messages.1`-t fogjuk felkeresni (és így tovább).

A `logrotate` naponta fut automatikus folyamatként vagy cronjobként a `/etc/cron.daily/logrotate` script segítségével, és az `/etc/logrotate.conf` konfigurációs fájlt olvassa be. Ez a fájl jól kommentezett, tartalmaz néhány globális opciót, és mindegyiknek rövid magyarázattal mutatja be a célját:

```
carol@debian:~$ sudo less /etc/logrotate.conf
# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

(...)
```

Látható, hogy az `/etc/logrotate.d` állományban található konfigurációs fájlok is szerepelnek egyes csomagokhoz. Ezek az állományok — többnyire — helyi definíciókat tartalmaznak, és megadják, hogy mely logfájlokat kell rotálni (ne feledjük, hogy a helyi definíciók elsőbbséget élveznek a globálisakkal szemben, és a későbbi definíciók felülírják a korábbiakat). Az alábbiakban a `/etc/logrotate.d/rsyslog` definíciójának egy részletét olvashatjuk:

```
/var/log/messages
{
        rotate 4
        weekly
        missingok
        notifempty
        compress
        delaycompress
```

```
        sharedscripts
        postrotate
                invoke-rc.d rsyslog rotate > /dev/null
        endscript
}
```

Mint látható, minden direktívát szóköz és/vagy egy opcionális egyenlőségjel (`=`) választ el az értékétől. A `postrotate` és `endscript` közötti soroknak azonban önmagukban kell megjelenniük. A magyarázat a következő:

`rotate 4`

4 hétig őrzi meg a logokat.

`weekly`

Hetente rotálja a logfájlokat.

`missingok`

Ne adjon ki hibaüzenetet, ha a logfájl hiányzik; csak lépjen tovább a következőre.

`notifempty`

Ne rotálja a logot, ha az üres.

`compress`

A logfájlok tömörítése az `gzip` programmal (alapértelmezett).

`delaycompress`

Az előző logfájl tömörítésének elhalasztása a következő rotációs ciklusra (csak a tömöréssel együtt használva hatékony). Ez akkor hasznos, ha egy programnak nem lehet megmondani, hogy zárja be a logfájlját, és így egy ideig még írhat az előző logfájlba.

`sharedscripts`

A *prerotate* és *postrotate* scriptekhez kapcsolódik. Annak érdekében, hogy egy scriptet ne kelljen többször végrehajtani, azaz függetlenül attól, hogy hány logfájl felel meg egy adott mintának (pl. `/var/log/mail/*`), a scriptet csak egyszer futtassa. A scriptek azonban nem fognak lefutni, ha a mintában szereplő logfájlok egyike sem igényel rotálást. Ráadásul, ha a scriptek hibával lépnek ki, a fennmaradó műveletek nem kerülnek végrehajtásra egyetlen log esetében sem.

`postrotate`

Egy *postrotate* script kezdetének a jelzése.

`invoke-rc.d rsyslog rotate > /dev/null`

A `/bin/sh` használata a `invoke-rc.d rsyslog rotate > /dev/null` futtatásához a logrotáció után.

`endscript`

Egy *postrotate* script végének a jelzése.

NOTE A direktívák teljes listája és magyarázata megtalálható a `logrotate.conf` man oldalán.

A kernel gyűrűpuffer

Mivel a kernel számos üzenetet generál, mielőtt az `rsyslogd` elérhetővé válik a rendszerindításkor, szükség van egy mechanizmusra ezen üzenetek regisztrálására. Itt jön a képbe a *kernel gyűrűpuffer* (kernel ring buffer). Ez egy fix méretű adatstruktúra, és — ezért — ahogy növekszik a mérete az új üzenetek miatt, a legrégebbiek eltűnnek.

A `dmesg` parancs kiírja a kernel gyűrűpufferét. A puffer mérete miatt ezt a parancsot általában a `grep` szövegszűrő segédprogrammal együtt használjuk. Itt láthatunk egy példát az univerzális soros busz eszközökkel kapcsolatos üzenetek keresésére:

```
root@debian:~# dmesg | grep "usb"
[    1.241182] usbcore: registered new interface driver usbfs
[    1.241188] usbcore: registered new interface driver hub
[    1.250968] usbcore: registered new device driver usb
[    1.339754] usb usb1: New USB device found, idVendor=1d6b, idProduct=0001, bcdDevice=
4.19
[    1.339756] usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
(...)
```

Gyakorló feladatok

1. Milyen segédprogramokat/parancsokat használhatunk a következő esetekben:

Cél és logfájl	**Segédprogram**
`/var/log/syslog.7.gz` olvasása	
`/var/log/syslog` olvasása	
Szűrés a `renewal` szóra a `/var/log/syslog` -ban	
`/var/log/faillog` olvasása	
`/var/log/syslog` dinamikus olvasása	

2. Rendezzük át a következő logbejegyzéseket úgy, hogy azok egy érvényes, megfelelő szerkezetű logüzenet legyenek:
   - `debian-server`
   - `sshd`
   - `[515]:`
   - `Sep 13 21:47:56`
   - `Server listening on 0.0.0.0 port 22`

   A helyes sorrend:

3. Milyen szabályokat kell hozzáadni az `/etc/rsyslog.conf` állományhoz a következők megvalósítása érdekében:
   - Küldjük az összes, `crit` (vagy magasabb) prioritású/súlyosságú üzenetet a `mail` eszközből a `/var/log/mail.crit`-be:
   - Küldjük az összes, `alert` és `emergency` üzenetet a `mail` eszközből a `/var/log/mail.urgent`-be:
   - A `cron` és `ntp` eszközökből jövő üzenetek kivételével, küldje az összes üzenetet — eszköztől és prioritástól függetlenül — a `/var/log/allmessages`-be:

- Az összes szükséges beállítás megfelelő konfigurálásával küldjük el az összes üzenetet a `mail` eszközből egy távoli hostnak, amelynek IP-címe `192.168.1.88`, TCP használatával és az alapértelmezett port megadásával:

- Eszköztől függetlenül, küldjük az összes `warning` prioritású üzenetet (*csak a* `warning` *prioritásúakat*) a `/var/log/warnings`-ba úgy, hogy megakadályozzuk a lemezre történő túlzott írást:

4. Tekintsük a következő részt az `/etc/logrotate.d/samba` állományból és magyarázzuk el a különböző opciókat:

```
carol@debian:~$ sudo head -n 11 /etc/logrotate.d/samba
/var/log/samba/log.smbd {
        weekly
        missingok
        rotate 7
        postrotate
                [ ! -f /var/run/samba/smbd.pid ] || /etc/init.d/smbd reload > /dev/null
        endscript
        compress
        delaycompress
        notifempty
}
```

Opció	Jelentés
`weekly`	
`missingok`	
`rotate 7`	
`postrotate`	
`endscript`	
`compress`	
`delaycompress`	
`notifyempty`	

Gondolkodtató feladatok

1. A “Sablonok és szűrési feltételek” szekcióban egy *kifejezés-alapú szűrőt* (expression-based filter) használtunk szűrési feltételként. A *tulajdonság-alapú szűrő* (property-based filter) másfajta szűrőtípus, amely csak az `rsyslogd` esetén létezik. Alakítsuk át a *kifejezés-alapú szűrőt* *tulajdonság-alapú szűrőre*:

Kifejezés-alapú szűrő	**Tulajdonság-alapú szűrő**
`if $FROMHOST-IP=='192.168.1.4' then ?RemoteLogs`	

2. Az `omusrmsg` egy beépített `rsyslog` modul, amely megkönnyíti a felhasználók értesítését (logüzeneteket küld a felhasználó termináljára). Írjunk egy szabályt, amely az összes eszköz összes *emergency* üzenetét elküldi mind a `root`, mind a `carol` normál felhasználónak!

Összefoglalás

Ebben a leckében megtanultuk:

- A rendszeradminisztráció ellátásához kritikusan fontos a logolás.
- Az `rsyslogd` a logok rendben tartásáért felelős szolgáltató.
- Néhány szolgáltatás gondoskodik a saját logjairól.
- A logok nagyjából rendszerlogokra és szolgáltatási/programlogokra oszthatók.
- Számos eszköz van, ami kényelmesebbé teszi a logok olvasását : `less`, `more`, `zless`, `zmore`, `grep`, `head` és `tail`.
- A legtöbb logfájl egyszerű szöveges fájl, azonban van néhány bináris logfájl is.
- A logolás szempontjából az `rsyslogd` speciális fájlokból (socketek és memóriapufferek) kapja a vonatkozó információkat, mielőtt feldolgozná azokat.
- A logok osztályozásához az `rsyslogd` az `/etc/rsyslog.conf`-ban vagy az `/etc/rsyslog.d/*`-ban található szabályokat használja.
- Bármely felhasználó manuálisan is beírhatja saját üzeneteit a rendszerlogba a `logger` segédprogrammal.
- Az `rsyslog` lehetővé teszi, hogy az összes IP-hálózati logot egy központi log-szerveren tartsa.
- A logfájlok dinamikus formázásához hasznosak lehetnek a sablonok.
- A logrotációnak két célja van: megakadályozni, hogy a régi logok túl sok helyet foglaljanak a lemezen, és a konzultációs logok kezelhetővé váljanak.

Válaszok a gyakorló feladatokra

1. Milyen segédprogramokat/parancsokat használhatunk a következő esetekben:

Cél és logfájl	**Segédprogram**
`/var/log/syslog.7.gz` olvasása	`zmore` vagy `zless`
`/var/log/syslog` olvasása	`more` vagy `less`
Szűrés a `renewal` szóra a `/var/log/syslog` -ban	`grep`
`/var/log/faillog` olvasása	`faillog -a`
`/var/log/syslog` dinamikus olvasása	`tail -f`

2. Rendezzük át a következő logbejegyzéseket úgy, hogy azok egy érvényes, megfelelő szerkezetű logüzenet legyenek:
   - `debian-server`
   - `sshd`
   - `[515]:`
   - `Sep 13 21:47:56`
   - `Server listening on 0.0.0.0 port 22`

   A helyes sorrend:

   ```
   Sep 13 21:47:56 debian-server sshd[515]: Server listening on 0.0.0.0 port 22
   ```

3. Milyen szabályokat kell hozzáadni az `/etc/rsyslog.conf` állományhoz a következők megvalósítása érdekében:
   - Küldjük az összes, `crit` (vagy magasabb) prioritású/súlyosságú üzenetet a `mail` eszközből a `/var/log/mail.crit`-be:

     ```
     mail.crit                /var/log/mail.crit
     ```

   - Küldjük az összes, `alert` és `emergency` üzenetet a `mail` eszközből a `/var/log/mail.urgent`-be:

```
mail.alert                      /var/log/mail.urgent
```

- A `cron` és `ntp` eszközökből jövő üzenetek kivételével, küldje az összes üzenetet — eszköztől és prioritástól függetlenül — a `/var/log/allmessages`-be:

```
*.*;cron.none;ntp.none                /var/log/allmessages
```

- Az összes szükséges beállítás megfelelő konfigurálásával küldjük el az összes üzenetet a `mail` eszközből egy távoli hostnak, amelynek IP-címe `192.168.1.88`, TCP használatával és az alapértelmezett port megadásával:

```
mail.* @@192.168.1.88:514
```

- Eszköztől függetlenül, küldjük az összes `warning` prioritású üzenetet (*csak a* `warning` *prioritásúakat*) a `/var/log/warnings`-ba úgy, hogy megakadályozzuk a lemezre történő túlzott írást:

```
*.=warning                      -/var/log/warnings
```

4. Tekintsük a következő részt az `/etc/logrotate.d/samba` állományból és magyarázzuk el a különböző opciókat:

```
carol@debian:~$ sudo head -n 11 /etc/logrotate.d/samba
/var/log/samba/log.smbd {
        weekly
        missingok
        rotate 7
        postrotate
                [ ! -f /var/run/samba/smbd.pid ] || /etc/init.d/smbd reload > /dev/null
        endscript
        compress
        delaycompress
        notifempty
}
```

Opció	Jelentés
`weekly`	Logrotáció hetente.

Opció	Jelentés
`missingok`	Ne legyen hibaüzenet, ha a log hiányzik, csak lépjen a következőre.
`rotate 7`	Tartson meg 7 heti logot.
`postrotate`	Futtassa a következő sorban található scriptet a logrotáció után.
`endscript`	Jelezze a *postrotate* script végét.
`compress`	Tömörítse a fájlokat a `gzip`-pel.
`delaycompress`	Halassza el a tömörítést a következő rotációs ciklusra a `compress` kombinálásával.
`notifyempty`	Ne rotálja a logot, ha üres.

Válaszok a gondolkodtató feladatokra

1. A “Sablonok és szűrési feltételek” szekcióban egy *kifejezés-alapú szűrőt* (expression-based filter) használtunk szűrési feltételként. A *tulajdonság-alapú szűrő* (property-based filter) másfajta szűrőtípus, amely csak az `rsyslogd` esetén létezik. Alakítsuk át a *kifejezés-alapú szűrőt tulajdonság-alapú szűrőre*:

Kifejezés-alapú szűrő	**Tulajdonság-alapú szűrő**
`if $FROMHOST-IP=='192.168.1.4' then ?RemoteLogs`	`:fromhost-ip, isequal, "192.168.1.4" ?RemoteLogs`

2. Az `omusrmsg` egy beépített `rsyslog` modul, amely megkönnyíti a felhasználók értesítését (logüzeneteket küld a felhasználó termináljára). Írjunk egy szabályt, amely az összes eszköz összes *emergency* üzenetét elküldi mind a `root`, mind a `carol` normál felhasználónak!

```
*.emerg                         :omusrmsg:root,carol
```

Linux Professional Institute

108.2 Lecke 2

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	108 Alapvető rendszerszolgáltatások
Fejezet:	108.2 Rendszernaplózás
Lecke:	2/2

Bevezetés

A `systemd` általános elterjedésével minden nagyobb disztribúcióban a naplózó daemon (`systemd-journald`) vált a szabványos naplózási szolgáltatássá. Ebben a leckében arról lesz szó, hogyan működik, és arról, hogy hogyan használhatjuk számos dologra: lekérdezhetjük, szűrhetjük az információkat különböző kritériumok szerint, konfigurálhatjuk a tárolását és a méretét, törölhetjük a régi adatokat, lekérdezhetjük az adatait egy mentési rendszerből vagy a fájlrendszer másolatából, és — végül, de nem utolsósorban — megérthetjük az `rsyslod`-vel való kölcsönhatását.

A `systemd` alapjai

A Fedora rendszerben először bevezetett `systemd` fokozatosan felváltotta a SysV Init-et, mint *de facto* rendszer- és szolgáltatáskezelőt a legtöbb nagyobb Linux disztribúcióban. Erősségei közé tartoznak a következők:

- Könnyű konfiguráció: unit fájlok a SysV Init scriptekkel szemben.
- Sokoldalú kezelés: a daemonok és folyamatok mellett eszközöket, socketeket és csatolási

pontokat (mount point) is kezel.

- Visszafelé kompatibilis (backward compatibility) mind a SysV Init, mind az Upstart rendszerrel.
- Párhuzamos betöltés a rendszerindítás során: a szolgáltatások párhuzamosan töltődnek be, szemben a Sysv Init szekvenciális betöltésével.
- A *journal* nevű naplózási szolgáltatást használja, amely a következő előnyökkel jár:
  - Az összes logot egy helyen központosítja.
  - Nem igényel logrotációt.
  - A logok letilthatók, RAM-ba tölthetők vagy állandóvá tehetők.

Egységek és célpontok

A `systemd` *egységekkel* (unit) dolgozik. A unit bármilyen erőforrás, amelyet a `systemd` kezelni tud (pl. hálózat, bluetooth, stb.). A unitokat — viszont — a *unit fájlok* szabályozzák. Ezek egyszerű szöveges fájlok, amelyek a `/lib/systemd/system` állományban találhatók, és tartalmazzák a konfigurációs beállításokat — *szekciók* (sections) és *direktívák* (directives) formájában — egy adott kezelendő erőforráshoz. Számos egységtípus létezik: `service`, `mount`, `automount`, `swap`, `timer`, `device`, `socket`, `path`, `timer`, `snapshot`, `slice`, `scope` és `target`. Így minden egységfájl neve a `<forrás_név>.<egység_típus>` mintát követi (pl. `reboot.service`).

A *célpont* (target) egy speciális egységtípus, amely hasonlít a SysV Init klasszikus futási szintjeihez (runlevel). Ennek oka, hogy a *target unit* különböző erőforrásokat egyesít egy adott rendszerállapot reprezentálására (pl. a `graphical.target` hasonló a `runlevel 5`-hez, stb.). A rendszerben lévő aktuális célpontot a `systemctl get-default` paranccsal ellenőrizhetjük:

```
carol@debian:~$ systemctl get-default
graphical.target
```

Másrészt a célpontok és a futási szintek abban különböznek egymástól, hogy az előbbiek kölcsönösen magukban foglalják egymást, míg az utóbbiak nem. Így egy célpont más célpontokat is előhívhat — ami a runlevelek esetében nem lehetséges.

NOTE A `systemd` egységek működésének magyarázata meghaladja ennek a leckének a kereteit.

A System Journal: `systemd-journald`

A `systemd-journald` a rendszer szolgáltatása, amely a naplózási információk fogadásáról gondoskodik a különböző forrásokból: kernelüzenetek, egyszerű és strukturált rendszerüzenetek,

szolgáltatások standard output és standard error üzenetei és a kernel audit alrendszeréből származó auditrekordok (további részletek a `systemd-journald` man oldalán). Feladata egy strukturált és indexált napló (journal) létrehozása és fenntartása.

A konfigurációs fájlja a `/etc/systemd/journald.conf`; és — ahogy sok más szolgáltatás esetén — a `systemctl` paranccsal *indíthatjuk el, indíthatjuk újra, állíthatjuk meg* vagy — egyszerűen — ellenőrizhetjük a *státuszát*:

```
root@debian:~# systemctl status systemd-journald
 systemd-journald.service - Journal Service
   Loaded: loaded (/lib/systemd/system/systemd-journald.service; static; vendor preset:
enabled)
   Active: active (running) since Sat 2019-10-12 13:43:06 CEST; 5min ago
     Docs: man:systemd-journald.service(8)
           man:journald.conf(5)
 Main PID: 178 (systemd-journal)
   Status: "Processing requests..."
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/systemd-journald.service
           └─178 /lib/systemd/systemd-journald
(...)
```

A `journal.conf.d/*.conf` típusú konfigurációs fájlok — amelyek tartalmazhatnak csomag-specifikus konfigurációkat — szintén létezhetnek (további információkért lsd. a `journald.conf` man oldalát).

Ha engedélyezve van, a napló tárolható tartósan a lemezen vagy volatilis módon egy RAM-alapú fájlrendszerben. A napló nem egyszerű szöveges fájl, hanem bináris, így nem használhatunk szövegelemző eszközöket, mint például a `less` vagy a `more` a tartalmának olvasására; helyette a `journalctl` parancsot kell használni.

A journal tartalmának lekérdezése

A `journalctl` az a segédprogram, amellyel a `systemd` naplót lekérdezhetjük. Vagy root-nak kell lennünk, vagy a `sudo` parancsot kell használnunk a meghívásához. Ha kapcsolók használata nélkül kérdezzük le, akkor a teljes naplót időrendi sorrendben fogja kiírni (a legrégebbi bejegyzésekkel az élen):

```
root@debian:~# journalctl
-- Logs begin at Sat 2019-10-12 13:43:06 CEST, end at Sat 2019-10-12 14:19:46 CEST. --
Oct 12 13:43:06 debian kernel: Linux version 4.9.0-9-amd64 (debian-kernel@lists.debian.org)
```

```
(...)
Oct 12 13:43:06 debian kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64
root=UUID=b6be6117-5226-4a8a-bade-2db35ccf4cf4 ro qu
(...)
```

Számos kapcsoló segítségével specifikusabbá tehetjük a lekérdezéseket:

`-r`

A journal üzenetei fordított sorrendben kerülnek megjelenítésre:

```
root@debian:~# journalctl -r
-- Logs begin at Sat 2019-10-12 13:43:06 CEST, end at Sat 2019-10-12 14:30:30 CEST. --
Oct 12 14:30:30 debian sudo[1356]: pam_unix(sudo:session): session opened for user root
by carol(uid=0)
Oct 12 14:30:30 debian sudo[1356]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/bin/journalctl -r
Oct 12 14:19:53 debian sudo[1348]: pam_unix(sudo:session): session closed for user root
(...)
```

`-f`

Kiírja a legfrissebb naplóüzeneteket, és folyamatosan frissíti az új bejegyzésekkel, amint azok hozzáadásra kerülnek — hasonlóan a `tail -f`-hez:

```
root@debian:~# journalctl -f
-- Logs begin at Sat 2019-10-12 13:43:06 CEST. --
(...)
Oct 12 14:44:42 debian sudo[1356]: pam_unix(sudo:session): session closed for user root
Oct 12 14:44:44 debian sudo[1375]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/bin/journalctl -f
Oct 12 14:44:44 debian sudo[1375]: pam_unix(sudo:session): session opened for user root
by carol(uid=0)

(...)
```

`-e`

A napló végére ugrik, így a legfrissebb bejegyzések láthatók a lapozóban:

```
root@debian:~# journalctl -e
(...)
Oct 12 14:44:44 debian sudo[1375]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
```

```
COMMAND=/bin/journalctl -f
Oct 12 14:44:44 debian sudo[1375]: pam_unix(sudo:session): session opened for user root
by carol(uid=0)
Oct 12 14:45:57 debian sudo[1375]: pam_unix(sudo:session): session closed for user root
Oct 12 14:48:39 debian sudo[1378]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/bin/journalctl -e
Oct 12 14:48:39 debian sudo[1378]: pam_unix(sudo:session): session opened for user root
by carol(uid=0)
```

`-n <value>, --lines=<value>`

A legfrissebb sorok *értékét* fogja kiírni (ha nincs megadva `<value>`, az alapértelmezett érték 10):

```
root@debian:~# journalctl -n 5
(...)
Oct 12 14:44:44 debian sudo[1375]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/bin/journalctl -f
Oct 12 14:44:44 debian sudo[1375]: pam_unix(sudo:session): session opened for user root
by carol(uid=0)
Oct 12 14:45:57 debian sudo[1375]: pam_unix(sudo:session): session closed for user root
Oct 12 14:48:39 debian sudo[1378]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/bin/journalctl -e
Oct 12 14:48:39 debian sudo[1378]: pam_unix(sudo:session): session opened for user root
by carol(uid=0)
```

`-k`, `--dmesg`

A `dmesg` parancs használatával egyenértékű:

```
root@debian:~# journalctl -k
-- Logs begin at Sat 2019-10-12 13:43:06 CEST, end at Sat 2019-10-12 14:53:20 CEST. --
Oct 12 13:43:06 debian kernel: Linux version 4.9.0-9-amd64 (debian-
kernel@lists.debian.org) (gcc version 6.3.0 20170516 (Debian 6.3.0-18
Oct 12 13:43:06 debian kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64
root=UUID=b6be6117-5226-4a8a-bade-2db35ccf4cf4 ro qu
Oct 12 13:43:06 debian kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating
point registers'
Oct 12 13:43:06 debian kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
(...)
```

Navigáció és keresés a journalban

A journal kimenetében az alábbi módokon navigálhatunk:

- A PageUp, PageDown és a kurzorbillentyűk a felfelé, lefelé, balra és jobbra mozgatáshoz.
- `>` a kimenet végére való ugrás.
- `<` a kimenet elejére való ugrás.

Az aktuális pozíciótól előre és hátrafelé is kereshetünk stringeket:

- Keresés előre: A `/` lenyomása után írjuk be a keresni kívánt stringet és nyomjuk meg az Entert.
- Keresés hátra: A `?` lenyomása után írjuk be a keresni kívánt stringet és nyomjuk meg az Entert.

A keresésekben található találatok között a `N` billentyűkombinációval léphetünk a találat következő előfordulásához, a `Shift` + `N` billentyűkombinációval pedig az előzőhöz.

A journal adatainak szűrése

A journal lehetővé teszi a logok szűrését különböző kritériumok alapján:

Boot number

`--list-boots`

Az összes elérhető bootot felsorolja. A kimenet három oszlopból áll; az első a boot számát adja meg (`0` az aktuális bootra utal, `-1` az előző, `-2` az előzőt megelőző és így tovább); a második oszlop a boot azonosítója; a harmadik az időbélyegeket mutatja:

```
root@debian:~# journalctl --list-boots
 0 83df3e8653474ea5aed19b41cdb45b78 Sat 2019-10-12 18:55:41 CEST—Sat 2019-10-12
19:02:24 CEST
```

`-b`, **`--boot`**

Megjeleníti az aktuális rendszerindítás összes üzenetét. A korábbi rendszerindítások naplóüzeneteinek megtekintéséhez csak adjunk hozzá egy offset paramétert a fentiek szerint! Például, ha az előző indításból származó üzeneteket szeretnénk látni, akkor írjuk be a `journalctl -b -1` paramétert! Ne feledjük azonban, hogy a korábbi logokból származó információk visszaállításához engedélyezni kell a journal állandóságát (persistence — ennek módját a következő szekcióban fogjuk megtanulni):

```
root@debian:~# journalctl -b -1
```

```
Specifying boot ID has no effect, no persistent journal was found
```

Priority

`-p`

Érdekes módon a `p` kapcsolóval súlyosság/prioritás szerint is szűrhetünk:

```
root@debian:~# journalctl -b -0 -p err
-- No entries --
```

A journal tájékoztat minket arról, hogy — eddig — nem érkezett egyetlen `error` (vagy annál magasabb) prioritású üzenet sem az aktuális rendszerindításból. Megjegyzés: a `-b -0` elhagyható, ha az aktuális indításról van szó.

NOTE Az összes `syslog` súlyossági fokozat (más néven prioritás) teljes listáját lsd. az előző leckében.

Time Interval

A `--since` és `--until` kapcsolók használatával a `journalctl` csak a meghatározott időn belül naplózott üzeneteket írja ki. A dátum megadásával az `YYYY-MM-DD HH:MM:SS` formátumot kell követnie. Ha az időt kihagyjuk, éjfél lesz az alapértelmezés, ha pedig a dátumot, akkor az aktuális nap. Például, ha a 19:00 és 19:01 között naplózott üzeneteket szeretnénk látni, akkor a következőket kell beírnunk:

```
root@debian:~# journalctl --since "19:00:00" --until "19:01:00"
-- Logs begin at Sat 2019-10-12 18:55:41 CEST, end at Sat 2019-10-12 20:10:50 CEST. --
Oct 12 19:00:14 debian systemd[1]: Started Run anacron jobs.
Oct 12 19:00:14 debian anacron[1057]: Anacron 2.3 started on 2019-10-12
Oct 12 19:00:14 debian anacron[1057]: Normal exit (0 jobs run)
Oct 12 19:00:14 debian systemd[1]: anacron.timer: Adding 2min 47.988096s random time.
```

Használhatunk egy kissé eltérő időmeghatározást is: `"integer time-unit ago"`. Így a két perccel ezelőtt naplózott üzenetek megtekintéséhez a `sudo journalctl --since "2 minutes ago"`-t kell beírni. A `+` és a `-` is használható az aktuális időhöz viszonyított időpontok megadására, így a `--since "-2 minutes"` és a `--since "2 minutes ago"` egyenértékű.

A numerikus kifejezések helyett használhatunk néhány kulcsszót is:

`yesterday`

Az aktuális napot megelőző nap éjfélkor.

`today`

Az aktuális nap éjfélkor.

`tomorrow`

Az aktuális napot követő nap éjfélkor.

`now`

Az aktuális idő.

Nézzük meg az összes üzenetet, amik tegnap éjfél és ma 21:00 között keletkeztek:

```
root@debian:~# journalctl --since "today" --until "21:00:00"
-- Logs begin at Sat 2019-10-12 20:45:29 CEST, end at Sat 2019-10-12 21:06:15 CEST. --
Oct 12 20:45:29 debian sudo[1416]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root
; COMMAND=/bin/systemctl r
Oct 12 20:45:29 debian sudo[1416]: pam_unix(sudo:session): session opened for user
root by carol(uid=0)
Oct 12 20:45:29 debian systemd[1]: Stopped Flush Journal to Persistent Storage.
(...)
```

NOTE Az időspecifikáció különböző szintaxisairől többet tudhatunk meg a `systemd.time` man oldalán.

Program

Egy adott futtatható programhoz kapcsolódó journal üzenetek megtekintéséhez a következő szintaxis használható: `journalctl /path/to/executable`:

```
root@debian:~# journalctl /usr/sbin/sshd
-- Logs begin at Sat 2019-10-12 20:45:29 CEST, end at Sat 2019-10-12 21:54:49 CEST. --
Oct 12 21:16:28 debian sshd[1569]: Accepted password for carol from 192.168.1.65 port
34050 ssh2
Oct 12 21:16:28 debian sshd[1569]: pam_unix(sshd:session): session opened for user carol
by (uid=0)
Oct 12 21:16:54 debian sshd[1590]: Accepted password for carol from 192.168.1.65 port
34052 ssh2
Oct 12 21:16:54 debian sshd[1590]: pam_unix(sshd:session): session opened for user carol
by (uid=0)
```

Unit

Ne feledjük, hogy az egység bármely erőforrás, amelyet a `systemd` kezel, és ezek alapján is

szűrhetünk.

`-u`

Egy adott egységről jelenít meg üzeneteket:

```
root@debian:~# journalctl -u ssh.service
-- Logs begin at Sun 2019-10-13 10:50:59 CEST, end at Sun 2019-10-13 12:22:59 CEST. --
Oct 13 10:51:00 debian systemd[1]: Starting OpenBSD Secure Shell server...
Oct 13 10:51:00 debian sshd[409]: Server listening on 0.0.0.0 port 22.
Oct 13 10:51:00 debian sshd[409]: Server listening on :: port 22.
(...)
```

NOTE Az összes betöltött és aktív egység kiíratásához használjuk a `systemctl list-units`; az összes telepített unit-fájlok kiíratásához pedig a `systemctl list-unit-files` parancsot!

Fields

A naplót meghatározott *mezők* (field) szerint is lehet szűrni az alábbi szintaxisok bármelyikével:

- `<mező-neve>=<érték>`
- `_<mező-neve>=<érték>_`
- `__<mező-neve>=<érték>`

`PRIORITY=`

A nyolc lehetséges `syslog` prioritási érték egyike, decimális stringként formázva:

```
root@debian:~# journalctl PRIORITY=3
-- Logs begin at Sun 2019-10-13 10:50:59 CEST, end at Sun 2019-10-13 14:30:50 CEST.
--
Oct 13 10:51:00 debian avahi-daemon[314]: chroot.c: open() failed: No such file or
directory
```

Figyeljük meg, hogy ugyanezt a kimenetet elérhettük volna a fentebb látott `sudo journalctl -p err` paranccsal is!

`SYSLOG_FACILITY=`

A lehetséges eszközkódok bármelyike tizedes stringként formázva. Például az összes felhasználói szintű üzenet megtekintéséhez használhatjuk az alábbit:

```
root@debian:~# journalctl SYSLOG_FACILITY=1
-- Logs begin at Sun 2019-10-13 10:50:59 CEST, end at Sun 2019-10-13 14:42:52 CEST.
--
Oct 13 10:50:59 debian mtp-probe[227]: checking bus 1, device 2:
"/sys/devices/pci0000:00/0000:00:06.0/usb1/1-1"
Oct 13 10:50:59 debian mtp-probe[227]: bus: 1, device: 2 was not an MTP device
Oct 13 10:50:59 debian mtp-probe[238]: checking bus 1, device 2:
"/sys/devices/pci0000:00/0000:00:06.0/usb1/1-1"
Oct 13 10:50:59 debian mtp-probe[238]: bus: 1, device: 2 was not an MTP device
```

`_PID=`

Egy adott folyamat ID által előállított üzenetek megjelenítése. A `systemd` által előállított összes üzenet megtekintéséhez írjuk be a következőt:

```
root@debian:~# journalctl _PID=1
-- Logs begin at Sun 2019-10-13 10:50:59 CEST, end at Sun 2019-10-13 14:50:15 CEST.
--
Oct 13 10:50:59 debian systemd[1]: Mounted Debug File System.
Oct 13 10:50:59 debian systemd[1]: Mounted POSIX Message Queue File System.
Oct 13 10:50:59 debian systemd[1]: Mounted Huge Pages File System.
Oct 13 10:50:59 debian systemd[1]: Started Remount Root and Kernel File Systems.
Oct 13 10:50:59 debian systemd[1]: Starting Flush Journal to Persistent Storage...
(...)
```

`_BOOT_ID`

A rendszerindítási azonosító alapján kiválaszthatjuk az adott rendszerindításból származó üzeneteket, például: `sudo journalctl _BOOT_ID=83df3e8653474ea5aed19b41cdb45b78`.

`_TRANSPORT`

Egy adott transzportról érkezett üzenetek megjelenítése. A lehetséges értékek a következők: `audit` (kernel audit alrendszer), `driver` (belsőleg generált), `syslog` (syslog socket), `journal` (natív journal protokoll), `stdout` (szolgáltatások standard kimenete vagy hibája), `kernel` (kernel gyűrűpuffer — ugyanaz, mint a `dmesg`, `journalctl -k` vagy `journalctl --dmesg`):

```
root@debian:~# journalctl _TRANSPORT=journal
-- Logs begin at Sun 2019-10-13 20:19:58 CEST, end at Sun 2019-10-13 20:46:36 CEST.
--
Oct 13 20:19:58 debian systemd[1]: Started Create list of required static device
```

```
nodes for the current kernel.
Oct 13 20:19:58 debian systemd[1]: Starting Create Static Device Nodes in /dev...
Oct 13 20:19:58 debian systemd[1]: Started Create Static Device Nodes in /dev.
Oct 13 20:19:58 debian systemd[1]: Starting udev Kernel Device Manager...
(...)
```

Mezők kombinálása

A mezők nem zárják ki egymást, így egy lekérdezésben több mezőt is használhatunk, azonban csak azok az üzenetek jelennek meg, amelyek mindkét mező értékének egyidejűleg megfelelnek:

```
root@debian:~# journalctl PRIORITY=3 SYSLOG_FACILITY=0
-- No entries --
root@debian:~# journalctl PRIORITY=4 SYSLOG_FACILITY=0
-- Logs begin at Sun 2019-10-13 20:19:58 CEST, end at Sun 2019-10-13 20:21:55 CEST. --
Oct 13 20:19:58 debian kernel: acpi PNP0A03:00: fail to add MMCONFIG information, can't
access extended PCI configuration (...)
```

Hacsak nem használjuk a `+` elválasztójelet két kifejezés összekapcsolására a logikai *OR* (VAGY) módjára:

```
root@debian:~# journalctl PRIORITY=3 + SYSLOG_FACILITY=0
-- Logs begin at Sun 2019-10-13 20:19:58 CEST, end at Sun 2019-10-13 20:24:02 CEST. --
Oct 13 20:19:58 debian kernel: Linux version 4.9.0-9-amd64 (debian-kernel@lists.debian.org)
(...9
Oct 13 20:19:58 debian kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64
root=UUID= (...)
(...)
```

Másrészt, két értéket is megadhatunk ugyanarra a mezőre, és akkor az összes olyan bejegyzés megjelenik, amely bármelyik értéknek megfelel:

```
root@debian:~# journalctl PRIORITY=1
-- Logs begin at Sun 2019-10-13 17:16:24 CEST, end at Sun 2019-10-13 17:30:14 CEST. --
-- No entries --
root@debian:~# journalctl PRIORITY=1 PRIORITY=3
-- Logs begin at Sun 2019-10-13 17:16:24 CEST, end at Sun 2019-10-13 17:32:12 CEST. --
Oct 13 17:16:27 debian connmand[459]: __connman_inet_get_pnp_nameservers: Cannot read /pro
Oct 13 17:16:27 debian connmand[459]: The name net.connman.vpn was not provided by any .se
```

NOTE A journal mezők az alábbi kategóriák valamelyikébe tartoznak: “User Journal Fields”, “Trusted Journal Fields”, “Kernel Journal Fields”, “Fields on behalf of a different program” és “Address Fields”. A témáról többet — beleértve a mezők teljes listáját — a `systemd.journal-fields(7)` man oldalán találhatunk.

Manuális bejegyzések a system journalban: `systemd-cat`

Ahogy a `logger` parancsot arra használjuk, hogy a parancssorból üzeneteket küldjünk a rendszerlogba (ahogy azt az előző leckében láttuk), a `systemd-cat` parancs hasonló — de sokkal szélesebb körű — célt szolgál a rendszer journallal. Lehetővé teszi, hogy szabványos bemeneti (*stdin*), kimeneti (*stdout*) és hiba (*stderr*) üzeneteket küldjünk a journalba.

Ha paraméterek nélkül hívjuk meg, akkor mindent, amit az *stdin*-ből beolvas, elküld a journalba. Ha végeztünk, nyomjuk meg a `Ctrl` + `C`: billentyűkombinációt!

```
carol@debian:~$ systemd-cat
This line goes into the journal.
^C
```

Ha egy csővezetékezett (piped) parancs kimenete kerül átadásra, akkor ez is elküldésre kerül:

```
carol@debian:~$ echo "And so does this line." | systemd-cat
```

Ha egy parancs követi, akkor a parancs kimenete is elküldésre kerül a — a *stderr*-rel együtt (ha van ilyen):

```
carol@debian:~$ systemd-cat echo "And so does this line too."
```

Lehetőség van a prioritási szint megadására is a `-p` kapcsolóval:

```
carol@debian:~$ systemd-cat -p emerg echo "This is not a real emergency."
```

A többi opcióról a `systemd-cat` man oldalán tudhatunk meg többet.

A journal utolsó négy sorának megjelenítése:

```
carol@debian:~$ journalctl -n 4
(...)
-- Logs begin at Sun 2019-10-20 13:43:54 CEST. --
```

```
Nov 13 23:14:39 debian cat[1997]: This line goes into the journal.
Nov 13 23:19:16 debian cat[2027]: And so does this line.
Nov 13 23:23:21 debian echo[2030]: And so does this line too.
Nov 13 23:26:48 debian echo[2034]: This is not a real emergency.
```

NOTE Az *emergency* prioritási szintű naplóbejegyzések a legtöbb rendszeren félkövéren és piros színűen jelennek meg.

Perzisztens journal tárhely

Amint korábban említettük, három lehetőség közül választhatunk a journal helyére vonatkozóan:

- A naplózás (journaling) teljesen kikapcsolható (az átirányítás más eszközökre, például a konzolra továbbra is lehetséges).
- Memóriában tartás — ami volatilissé teszi — és törlésük minden újraindításkor. Ebben az esetben a `/run/log/journal` mappa jön létre és az lesz használatban.
- Perzisztenssé tehetjük, hogy a naplók a lemezre kerüljenek. Ebben az esetben a naplóüzenetek a `/var/log/journal` mappába kerülnek.

Az alapértelmezett viselkedés a következő: ha a `/var/log/journal/` nem létezik, a naplófájlok a `/run/log/journal/` mappába mentődnek, és — ezért — újraindításkor elvesznek. A mappa neve — az `/etc/machine-id` — egy newline-terminált, hexadecimális, 32 karakteres, kisbetűs string:

```
carol@debian:~$ ls /run/log/journal/8821e1fdf176445697223244d1dfbd73/
system.journal
```

Ha megpróbálunk beleolvasni a less-el, figyelmeztetést kapunk, ezért használjuk helyette a `journalctl` parancsot:

```
root@debian:~# less /run/log/journal/9a32ba45ce44423a97d6397918de1fa5/system.journal
"/run/log/journal/9a32ba45ce44423a97d6397918de1fa5/system.journal" may be a binary file.
See it anyway?
root@debian:~# journalctl
-- Logs begin at Sat 2019-10-05 21:26:38 CEST, end at Sat 2019-10-05 21:31:27 CEST. --
(...)
Oct 05 21:26:44 debian systemd-journald[1712]: Runtime journal
(/run/log/journal/9a32ba45ce44423a97d6397918de1fa5) is 4.9M, max 39.5M, 34.6M free.
Oct 05 21:26:44 debian systemd[1]: Started Journal Service.
(...)
```

Ha a `/var/log/journal/` létezik, a naplófájlok tartósan ott lesznek tárolva. Ha ez a mappa törlődik, a `systemd-journald` nem hozza létre újra, hanem a `/run/log/journal` mappába ír. Amint újra létrehozzuk a `/var/log/journal/`-t és újraindítjuk a daemont, a perzisztens naplózás újraindul:

```
root@debian:~# mkdir /var/log/journal/
root@debian:~# systemctl restart systemd-journald
root@debian:~# journalctl
(...)
Oct 05 21:33:49 debian systemd-journald[1712]: Received SIGTERM from PID 1 (systemd).
Oct 05 21:33:49 debian systemd[1]: Stopped Journal Service.
Oct 05 21:33:49 debian systemd[1]: Starting Journal Service...
Oct 05 21:33:49 debian systemd-journald[1768]: Journal started
Oct 05 21:33:49 debian systemd-journald[1768]: System journal
(/var/log/journal/9a32ba45ce44423a97d6397918de1fa5) is 8.0M, max 1.1G, 1.1G free.
Oct 05 21:33:49 debian systemd[1]: Started Journal Service.
Oct 05 21:33:49 debian systemd[1]: Starting Flush Journal to Persistent Storage...
(...)
```

NOTE

Alapértelmezés szerint minden bejelentkezett felhasználóhoz külön naplófájlok tartoznak, amelyek a `/var/log/journal/` mappában találhatók, így — a `system.journal` fájlokkal együtt — a `user-1000.journal` típusú fájlokat is megtaláljuk!

Az imént említetteken túlmenően a journal daemon naplótárolási módját a telepítés után a konfigurációs fájl módosításával meg lehet változtatni: `/etc/systemd/journald.conf`. A legfontosabb opció a `Storage=`, és a következő értékeket veheti fel:

`Storage=volatile`

A naplóadatok kizárólag a memóriában tárolódnak — a `/run/log/journal/` alatt. Ha nincs ilyen mappa, akkor létrejön.

`Storage=persistent`

Alapértelmezés szerint a naplóadatok a lemezen tárolódnak — a `/var/log/journal/` alatt — ami visszakerül a memóriába (`/run/log/journal/`) a boot korai fázisaiban, és ha a lemez nem írható. Szükség esetén mindkét mappa létrejön.

`Storage=auto`

Az `auto` hasonló a `persistent`-hez, de a `/var/log/journal` mappa nem jön létre, akkor sem, ha szükség van rá. Ez az alapértelmezés.

`Storage=none`

Az összes naplóadatot el kell vetni. A továbbítás más célpontokra, például a konzolra, a kernel naplózási pufferébe vagy egy syslog socketre azonban továbbra is lehetséges.

Például, ha a `systemd-journald` létrehozza a `/var/log/journal/`-t és átáll a perzisztens tárolásra, akkor szerkesszük az `/etc/systemd/journald.conf` fájlt és állítsuk be a `Storage=persistent` értéket, mentsük el a fájlt és indítsuk újra a daemont a `sudo systemctl restart systemd-journald`-al. Hogy megbizonyosodjunk arról, hogy az újraindítás hibátlanul ment, bármikor ellenőrizhetjük a daemon állapotát:

```
root@debian:~# systemctl status systemd-journald
 systemd-journald.service - Journal Service
   Loaded: loaded (/lib/systemd/system/systemd-journald.service; static; vendor preset:
enabled)
   Active: active (running) since Wed 2019-10-09 10:03:40 CEST; 2s ago
     Docs: man:systemd-journald.service(8)
           man:journald.conf(5)
 Main PID: 1872 (systemd-journal)
   Status: "Processing requests..."
    Tasks: 1 (limit: 3558)
   Memory: 1.1M
   CGroup: /system.slice/systemd-journald.service
           └─1872 /lib/systemd/systemd-journald

Oct 09 10:03:40 debian10 systemd-journald[1872]: Journal started
Oct 09 10:03:40 debian10 systemd-journald[1872]: System journal
(/var/log/journal/9a32ba45ce44423a97d6397918de1fa5) is 8.0M, max 1.2G, 1.2G free.
```

NOTE

A `/var/log/journal/<gép-id>/` vagy `/run/log/journal/<gép-id>/` naplófájlok a `.journal` utótagot kapják (pl. `system.journal`). Ha azonban sérültek, vagy a daemon nem megfelelően áll le, akkor a `~` hozzáfűződik a fájl nevéhez (pl. `system.journal~`), és a daemon egy új, tiszta fájlba kezd el írni.

Régi journal adatok törlése: a journal mérete

A naplók *journal fájlokban* kerülnek mentésre, amelyek fájlneve `.journal`-ra vagy `.journal~`-ra végződik , és a megfelelő mappában (`/run/log/journal` vagy `/var/log/journal` a beállítások szerint) található. Annak ellenőrzéséhez, hogy mennyi lemezterületet foglalnak jelenleg a naplófájlok (mind az archivált, mind az aktív), használjuk a `--disk-usage` kapcsolót:

```
root@debian:~# journalctl --disk-usage
```

```
Archived and active journals take up 24.0M in the filesystem.
```

A `systemd` naplófájlok alapértelmezés szerint legfeljebb a tároló fájlrendszer méretének 10%-át foglalják el. Például egy 1 GB-os fájlrendszeren nem foglalnak el több, mint 100 MB-ot. Amint ezt a határt elérjük, a régi naplók elkezdenek eltűnni, hogy a határérték közelében maradjunk.

A tárolt naplófájlok méretkorlátozásának beállítása azonban az `/etc/systemd/journald.conf` állományban található konfiguráció módosításával kezelhető. Ezek az opciók a használt fájlrendszer típusától függően két kategóriába sorolhatók: állandó (`/var/log/journal`) vagy memórián belüli (`/run/log/journal`). Az előbbi a `System` előtaggal ellátott opciókat használja, és csak akkor érvényes, ha a tartós naplózás megfelelően engedélyezve van, és ha a rendszer teljesen elindult. Az utóbbiban az opciónevek a `Runtime` szóval kezdődnek, és a következő esetekben lesznek alkalmazhatók:

`SystemMaxUse=`, **`RuntimeMaxUse=`**

Ezek szabályozzák a journal által elfoglalható lemezterület nagyságát. Alapértelmezés szerint a fájlrendszer méretének 10%-a, de módosítható (pl. `SystemMaxUse=500M`), amíg nem haladja meg a maximális 4GiB-ot.

`SystemKeepFree=`, **`RuntimeKeepFree=`**

Ezek szabályozzák a többi felhasználó számára szabadon hagyandó lemezterület mennyiségét. Alapértelmezés szerint a fájlrendszer méretének 15%-a, de módosítható (pl. `SystemKeepFree=500M`), amíg nem haladja meg a maximális 4GiB-ot.

Ami a `*MaxUse` és a `*KeepFree` elsőbbségét illeti, a `systemd-journald` a két érték közül a kisebbet használja. Ne feledjük, hogy csak az archivált (nem pedig az aktív) naplófájlok kerülnek törlésre.

`SystemMaxFileSize=`, **`RuntimeMaxFileSize=`**

Ezek szabályozzák az egyes naplófájlok maximális méretét. Az alapértelmezett érték a `*MaxUse` 1/8-a. A méretcsökkentés szinkron módon történik, és az értékek megadhatók bájtban vagy a `K`, `M`, `G`, `T`, `P`, `E` használatával, utalva a Kibibyte, Mebibyte, Gibibyte, Tebibyte, Pebibyte and Exbibyte kifejezésekre.

`SystemMaxFiles=`, **`RuntimeMaxFiles=`**

Meghatározzák a tárolható egyedi és archivált naplófájlok maximális számát (az aktív naplófájlokat ez nem érinti). Az alapértelmezett érték 100.

A naplóüzenetek méretalapú törlése és rotációja mellett a `systemd-journald` a következő két opcióval időalapú kritériumokat is lehetővé tesz: `MaxRetentionSec=` és `MaxFileSec=`. A `journald.conf` man oldalán találhatunk további információkat ezekről és más opciókról.

NOTE Amikor a `systemd-journald` alapértelmezett viselkedését az `/etc/systemd/journald.conf` állományban lévő opciók megjegyzéseinek feloldásával és szerkesztésével módosítjuk, újra kell indítanunk a daemont, hogy a módosítások hatályba lépjenek.

A journal kiporszívózása

Az archivált naplófájlokat bármikor manuálisan kitakaríthatjuk az alábbi három lehetőség bármelyikével:

`--vacuum-time=`

Ez az időalapú beállítás a megadott időkeretnél régebbi időbélyegzővel rendelkező üzeneteket törli a naplófájlokból. Az értékeket a következő utótagok bármelyikével kell írni: `s`, `m`, `h`, `days` (vagy `d`), `months`, `weeks` (vagy `w`) és `years` (vagy `y`). Például az archivált journal fájlokban lévő, 1 hónapnál régebbi üzenetek eltávolításához használhatjuk a következőt:

```
root@debian:~# journalctl --vacuum-time=1months
Deleted archived journal
/var/log/journal/7203088f20394d9c8b252b64a0171e08/system@27dd08376f71405a91794e632ede97ed
-0000000000000001-00059475764d46d6.journal (16.0M).
Deleted archived journal /var/log/journal/7203088f20394d9c8b252b64a0171e08/user-
1000@e7020d80d3af42f0bc31592b39647e9c-000000000000008e-00059479df9677c8.journal (8.0M).
```

`--vacuum-size=`

Ez a méretalapú opció addig törli az archivált naplófájlokat, amíg azok a megadott méret alatti értéket nem foglalnak el. Az értékeket a következő utótagok bármelyikével kell jelölni: `K`, `M`, `G` vagy `T`. Példa az archivált naplófájlok törlésére, amíg azok mérete nem csökken 100 Mebibyte alá:

```
root@debian:~# journalctl --vacuum-size=100M
Vacuuming done, freed 0B of archived journals from
/run/log/journal/9a32ba45ce44423a97d6397918de1fa5.
```

`--vacuum-files=`

Ez az opció gondoskodik arról, hogy a megadott számnál több archivált naplófájl ne maradjon. Az érték egy egész szám. Példa az archivált naplófájlok számának 10-re való korlátozására:

```
root@debian:~# journalctl --vacuum-files=10
Vacuuming done, freed 0B of archived journals from
```

```
/run/log/journal/9a32ba45ce44423a97d6397918de1fa5.
```

Az úgynevezett porszívózás (vacuuming) csak az archivált journal fájlokat távolítja el. Ha mindentől meg akarunk szabadulni (beleértve az aktív journal fájlokat is), akkor egy olyan jelet (signal) kell használnunk (`SIGUSR2`), amely a `--rotate` kapcsolóval kéri az azonnali rotálásukat. Más fontos jeleket a következő opciókkal lehet meghívni:

`--flush` (`SIGUSR1`)

A journal lehúzását kéri a `/run/'-ből a `/var/'-be, hogy perzisztens legyen. Ehhez az kell, hogy a perzisztens naplózás engedélyezve legyen, és a `/var/` mountolva legyen.

`--sync` (`SIGRTMIN+1`)

Arra szolgál, hogy az összes meg nem írt naplóadatot a lemezre írja.

NOTE A naplófájl belső konzisztenciájának ellenőrzéséhez használjuk a `journalctl`-t a `--verify` kapcsolóval! Az ellenőrzés során egy előrehaladási sávot látunk majd és minden lehetséges probléma megjelenik.

Journal adatok kinyerése egy rescue rendszerből

Rendszergazdaként előfordulhat, hogy olyan helyzetbe kerülünk, amikor egy meghibásodott gép merevlemezén lévő naplófájlokhoz kell hozzáférnünk egy rescue rendszeren keresztül (egy bootolható CD vagy USB-kulcs, amely egy élő Linux-disztribúciót tartalmaz).

A `journalctl` a `/var/log/journal/<gép-id>/` mappában keresi a naplófájlokat. Mivel a rescue- és a hibás rendszer gépazonosítója eltérő lesz, a következő kapcsolót kell használnunk:

`-D </path/to/dir>`, `--directory=</path/to/dir>`

Ezzel a kapcsolóval megadjuk egy mappa elérési útvonalát, ahol a `journalctl` a naplófájlokat keresi az alapértelmezett futásidejű és a rendszer lokációi helyett.

Ezért szükséges, hogy a hibás rendszer `rootfs'-jét` (/dev/sda1`) csatlakoztassuk a rescue rendszer fájlrendszerére, és így olvassuk be a naplófájlokat:

```
root@debian:~# journalctl -D /media/carol/faulty.system/var/log/journal/
-- Logs begin at Sun 2019-10-20 12:30:45 CEST, end at Sun 2019-10-20 12:32:57 CEST. --
oct 20 12:30:45 suse-server kernel: Linux version 4.12.14-lp151.28.16-default
(geeko@buildhost) (...)
oct 20 12:30:45 suse-server kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.12.14-
lp151.28.16-default root=UUID=7570f67f-4a08-448e-aa09-168769cb9289 splash=>
```

```
oct 20 12:30:45 suse-server kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating
point registers'
oct 20 12:30:45 suse-server kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
(...)
```

Más, ebben a helyzetben hasznos kapcsolók az alábbiak lehetnek:

`-m`, **`--merge`**
: Összefűzi az összes elérhető journal bejegyzését a `/var/log/journal` alatt, beleértve a remote naplókat is.

`--file`
: Megmutatja egy adott fájl bejegyzéseit, például: `journalctl --file /var/log/journal/64319965bda04dfa81d3bc4e7919814a/user-1000.journal`.

`--root`
: A gyökérmappát jelölő mappa elérési útját adja át argumentumként. A `journalctl` ott fogja keresni a naplófájlokat (pl. `journalctl --root /faulty.system/`).

A `journalctl` man oldalán több információt is találhatunk.

Naplóadatok továbbítása egy hagyományos `syslog` daemonnak

A journal naplóadatai elérhetővé tehetők egy hagyományos `syslog` daemon számára a következő módon:

- Üzenetek továbbítása a `/run/systemd/journal/syslog` socket fájlba, hogy a `syslogd` olvashassa. Ez a lehetőség a `ForwardToSyslog=yes` opcióval engedélyezhető.
- Egy `syslog` daemon, amely úgy viselkedik, mint a `journalctl`, tehát a naplóüzeneteket közvetlenül a journal fájlokból olvassa. Ebben az esetben a `Storage` opció a releváns; ennek az értéke bármi lehet, ami nem `none`.

NOTE

Hasonlóképpen, a következő beállításokkal továbbíthatjuk a naplóüzeneteket más célállomásokra: `ForwardToKMsg` (kernel logpuffer — `kmsg`), `ForwardToConsole` (a rendszerkonzol) vagy `ForwardToWall` (minden, `wall`-al bejelentkezett felhasználó). További információt a `journald.conf` man oldalán találhatunk.

Gyakorló feladatok

1. Feltételezve, hogy `root` jogunk van, töltsük ki a táblázatot a megfelelő `journalctl` paranccsal:

Cél	**Parancs**
Kernel bejegyzések kiíratása	
Üzenetek kiíratása a második indítástól a journal elejétől kezdve	
Üzenetek kiíratása a második indítástól a journal végétől kezdve	
Legutóbbi naplóüzenetek kiíratása és az újak figyelése	
Csak az új üzenetek nyomtatása mostantól, és a kimenet folyamatos frissítése	
Az előző rendszerindításból származó, `warning` prioritású üzenetek kiíratása fordított sorrendben	

2. A journal daemon tárolással kapcsolatos viselkedését leginkább a `/etc/systemd/journald.conf` állományban található `Storage` opció értéke szabályozza. Jelezzük az alábbi táblázatban, hogy milyen viselkedés milyen értékhez kapcsolódik:

Viselkedés	`Storage=auto`	`Storage=none`	`Storage=persistent`	`Storage=volatile`
A naplóadatok törlésre kerülnek, de a továbbítás lehetséges.				
A rendszer indítása után a naplóadatok a `/var/log/journal` alatt kerülnek tárolásra. Ha még nem létezik, a mappa létrejön.				

Viselkedés	`Storage=auto`	`Storage=none`	`Storage=persistent`	`Storage=volatile`
A rendszer indítása után a naplóadatok a `/var/log/journal` alatt kerülnek tárolásra. A mappa akkor sem jön létre, ha még nem létezik.				
A naplóadatokat a `/var/run/journal` alatt tárolja, de nem maradnak meg az újraindítások után.				

3. Mint azt megtanultuk, a naplót idő, méret és a fájlok száma alapján manuálisan is ki lehet takarítani. Végezzük el a következő feladatokat a `journalctl` és a megfelelő kapcsolók segítségével:
   - Ellenőrizzük, hogy mennyi lemezterületet foglalnak el a naplófájlok:
   - Csökkentsük az archivált naplófájlok számára fenntartott hely mennyiségét, és állítsuk be 200MiB-ra:
   - Ellenőrizzük a lemezterületet ismét és magyarázzuk meg a részleteket:

Gondolkodtató feladatok

1. Milyen beállításokat kell módosítanunk az `/etc/systemd/journald.conf` állományban, hogy az üzeneteket a `/dev/tty5`-be továbbítsuk? Milyen értékeket kell adni a kapcsolóknak?

2. Adjuk meg a megfelelő `journalctl` szűrőt a következők kiírásához:

Cél	**Szűrő + érték**
Egy adott felhasználóhoz tartozó üzenetek kiíratása	
Egy `debian` nevű host üzeneteinek kiíratása	
Egy adott csoporthoz tartozó üzenetek kiíratása	
A `root` csoporthoz tartozó üzenetek kiíratása	
A futtatható elérési útvonal alapján `sudo` üzenetek kiíratása	
A parancs neve alapján `sudo` üzenetek kiíratása	

3. Prioritás szerinti szűrés esetén a megadottnál magasabb prioritású naplók is szerepelnek a listában; például a `journalctl -p err` az *error*, *critical*, *alert* és *emergency* üzeneteket fogja kiírni. A `journalctl` azonban csak egy adott tartományt is megjeleníthet. Milyen parancsot kellene használnunk ahhoz, hogy a `journalctl` csak a *warning*, *error* és *critical* prioritási szintekbe tartozó üzeneteket írja ki?

4. A prioritási szintek numerikusan is megadhatók. Írjuk át az előző feladatban szereplő parancsot a prioritási szintek numerikus ábrázolásával:

Összefoglalás

Ebben a leckében megtanultuk:

- A `systemd` rendszer- és szolgáltatásmenedzser használatának előnyeit.
- A `systemd` egységeinek (unit) és célpontjainak (target) alapjait.
- Honnan kapja a `systemd-journald` a naplózási adatokat.
- A kapcsolókat, amiket átadhatunk a `systemctl`-nek, hogy irányítsa a `systemd-journald`-t: `start`, `status`, `restart` és `stop`.
- Hol található a journal konfigurációs fájlja — `/etc/systemd/journald.conf` — és a fő opcióit.
- Hogyan lehet lekérni általánosságban journalt és hogyan lehet konkrét adatokat lekérni szűrők használatával.
- Hogyan lehet navigálni és keresni a journalban.
- Hogyan kezeljük a journal fájlok tárolását: memóriában vagy lemezen.
- A naplózás teljes letiltása.
- Hogyan ellenőrizhetjük a journal által elfoglalt lemezterületet, hogyan érvényesíthetjük a tárolt naplófájlok méretkorlátozását és hogyan törölhetjük az archivált naplófájlokat manuálisan (*vacumming*).
- Hogyan lehet lekérni a naplóadatokat egy rescue rendszerből.
- Hogyan továbbítsuk a naplóadatokat egy hagyományos `syslog` daemonnak.

A leckében használt parancsok:

`systemctl`
: A `systemd` rendszer- és szolgáltatáskezelő vezérlése.

`journalctl`
: A `systemd` journal lekérdezése.

`ls`
: Mappa tartalmának listázása.

`less`
: Fájl tartalmának megtekintése.

`mkdir`

Mappa létrehozása.

Válaszok a gyakorló feladatokra

1. Feltételezve, hogy `root` jogunk van, töltsük ki a táblázatot a megfelelő `journalctl` paranccsal:

Cél	**Parancs**
Kernel bejegyzések kiíratása	`journalctl -k` vagy `journalctl --dmesg`
Üzenetek kiíratása a második indítástól a journal elejétől kezdve	`journalctl -b 2`
Üzenetek kiíratása a második indítástól a journal végétől kezdve	`journalctl -b -2 -r` vagy `journalctl -r -b -2`
Legutóbbi naplóüzenetek kiíratása és az újak figyelése	`journalctl -f`
Csak az új üzenetek nyomtatása mostantól, és a kimenet folyamatos frissítése	`journalctl --since "now" -f`
Az előző rendszerindításból származó, `warning` prioritású üzenetek kiíratása fordított sorrendben	`journalctl -b -1 -p warning -r`

2. A journal daemon tárolással kapcsolatos viselkedését leginkább a `/etc/systemd/journald.conf` állományban található `Storage` opció értéke szabályozza. Jelezzük az alábbi táblázatban, hogy milyen viselkedés milyen értékhez kapcsolódik:

Viselkedés	`Storage=auto`	`Storage=none`	`Storage=persistent`	`Storage=volatile`
A naplóadatok törlésre kerülnek, de a továbbítás lehetséges.		x		
A rendszer indítása után a naplóadatok a `/var/log/journal` alatt kerülnek tárolásra. Ha még nem létezik, a mappa létrejön.			x	

Viselkedés	`Storage=auto`	`Storage=none`	`Storage=persistent`	`Storage=volatile`
A rendszer indítása után a naplóadatok a `/var/log/journal` alatt kerülnek tárolásra. A mappa akkor sem jön létre, ha még nem létezik.	x			
A naplóadatokat a `/var/run/journal` alatt tárolja, de nem maradnak meg az újraindítások után.				x

3. Mint azt megtanultuk, a naplót idő, méret és a fájlok száma alapján manuálisan is ki lehet takarítani. Végezzük el a következő feladatokat a `journalctl` és a megfelelő kapcsolók segítségével:
   - Ellenőrizzük, hogy mennyi lemezterületet foglalnak el a naplófájlok:

     ```
     journalctl --disk-usage
     ```

   - Csökkentsük az archivált naplófájlok számára fenntartott hely mennyiségét, és állítsuk be 200MiB-ra:

     ```
     journalctl --vacuum-size=200M
     ```

   - Ellenőrizzük a lemezterületet ismét és magyarázzuk meg a részleteket:

     ```
     journalctl --disk-usage
     ```

     Nincs összefüggés, mivel a `--disk-usage` az aktív és az archivált naplófájlok által elfoglalt

helyet mutatja, míg a `--vacuum-size` csak az archivált fájlokra vonatkozik.

Válaszok a gondolkodtató feladatokra

1. Milyen beállításokat kell módosítanunk az `/etc/systemd/journald.conf` állományban, hogy az üzeneteket a `/dev/tty5`-be továbbítsuk? Milyen értékeket kell adni a kapcsolóknak?

```
ForwardToConsole=yes
TTYPath=/dev/tty5
```

2. Adjuk meg a megfelelő `journalctl` szűrőt a következők kiírásához:

Cél	**Szűrő + érték**
Egy adott felhasználóhoz tartozó üzenetek kiíratása	`_ID=<user-id>`
Egy `debian` nevű host üzeneteinek kiíratása	`_HOSTNAME=debian`
Egy adott csoporthoz tartozó üzenetek kiíratása	`_GID=<group-id>`
A `root` csoporthoz tartozó üzenetek kiíratása	`_UID=0`
A futtatható elérési útvonal alapján `sudo` üzenetek kiíratása	`_EXE=/usr/bin/sudo`
Based on the command name, print `sudo` messages	`_COMM=sudo`

3. Prioritás szerinti szűrés esetén a megadottnál magasabb prioritású naplók is szerepelnek a listában; például a `journalctl -p err` az *error*, *critical*, *alert* és *emergency* üzeneteket fogja kiírni. A `journalctl` azonban csak egy adott tartományt is megjeleníthet. Milyen parancsot kellene használnunk ahhoz, hogy a `journalctl` csak a *warning*, *error* és *critical* prioritási szintekbe tartozó üzeneteket írja ki?

```
journalctl -p warning..crit
```

4. A prioritási szintek numerikusan is megadhatók. Írjuk át az előző feladatban szereplő parancsot a prioritási szintek numerikus ábrázolásával:

```
journalctl -p 4..2
```


108.3 A Mail Transfer Agent (MTA) alapjai

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 102, Objective 108.3

Súlyozás

3

Kulcsfontosságú ismeretek

- E-mail aliasok létrehozása.
- E-mail továbbítás beállítása.
- Az általánosan elérhető MTA programok (postfix, sendmail, exim) ismerete (nincs konfiguráció).

A használt fájlok, kifejezések és segédprogramok listája

- `~/.forward`
- sendmail emulation layer commands
- `newaliases`
- `mail`
- `mailq`
- `postfix`
- `sendmail`
- `exim`

108.3 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	108 Alapvető rendszerszolgáltatások
Fejezet:	108.3 A Mail Transfer Agent (MTA) alapjai
Lecke:	1/1

Bevezetés

A Unix-szerű operációs rendszerekben, mint például a Linux, minden felhasználónak saját *inbox*-a van: egy speciális hely a fájlrendszeren, amely más, nem root felhasználók számára elérhetetlen, és a felhasználó személyes e-mail üzeneteit tárolja. Az új bejövő üzeneteket a *Mail Transfer Agent* (MTA) teszi hozzá a felhasználó postaládájához. Az MTA egy rendszerszolgáltatásként futó program, amely összegyűjti a más helyi fiókok által küldött üzeneteket, valamint a hálózatról érkező, távoli felhasználói fiókokból küldött üzeneteket.

Ugyanez az MTA felelős az üzenetek hálózatra küldéséért is, ha a célcím egy távoli fiókra mutat. Ezt úgy oldja meg, hogy egy fájlrendszeri helyet használ a rendszer összes felhasználója számára e-mail *kimenő postafiókként* (outbox): amint egy felhasználó új üzenetet helyez el a kimenő postafiókban, az MTA azonosítja a célhálózati csomópontot a cél e-mail cím által megadott domainnévből — a `@` jel utáni részből --, majd megpróbálja az üzenetet a távoli MTA-nak továbbítani a *Simple Mail Transfer Protocol* (SMTP) segítségével. Az SMTP-t a megbízhatatlan hálózatok figyelembevételével tervezték, ezért megpróbál alternatív kézbesítési útvonalakatis létrehozni, ha az elsődleges levelezési célcsomópont elérhetetlen lenne.

Lokális és távoli MTA

A hálózatba kapcsolt gépek hagyományos felhasználói fiókjai alkotják a legegyszerűbb e-mail üzenetváltó forgatókönyvet, ahol minden hálózati csomópont saját MTA daemont futtat. Az MTA-n kívül nincs szükség más szoftverre az e-mailek küldéséhez és fogadásához. A gyakorlatban azonban gyakoribb, hogy távoli e-mail fiókot használnak, és nincs aktív helyi MTA szolgáltatás (azaz ehelyett egy e-mail kliensalkalmazást használnak a távoli fiók eléréséhez).

A helyi fiókokkal ellentétben a távoli e-mail fiók — más néven *távoli postafiók* (remote mailbox) — felhasználói hitelesítést igényel a felhasználó postafiókjához és a távoli MTA-hoz (ebben az esetben egyszerűen *SMTP-szervernek* nevezik) való hozzáféréshez. Míg a helyi postafiókkal és MTA-val interakcióba lépő felhasználót a rendszer már azonosítja, addig egy távoli rendszernek ellenőriznie kell a felhasználó személyazonosságát, mielőtt IMAP vagy POP3 segítségével kezelné üzeneteit.

NOTE

Napjainkban az e-mailek küldésének és fogadásának legelterjedtebb módja egy távoli szerveren lévő fiókon keresztül történik, például egy vállalat központi e-mail szerverén, amely az összes alkalmazott fiókját hostolja, vagy egy személyes e-mail szolgáltatáson keresztül, mint például a Google *Gmail*. A helyben kézbesített üzenetek összegyűjtése helyett az e-mail kliensalkalmazás a távoli postafiókhoz csatlakozik, és onnan kéri le az üzeneteket. Az üzenetek távoli szerverről való lekérdezésére általában a POP3 és az IMAP protokollt használják, de más, nem szabványos, saját protokollok is használhatók.

Ha egy MTA-daemon fut a helyi rendszeren, a helyi felhasználók e-mailt küldhetnek más helyi felhasználóknak vagy egy távoli gépen lévő felhasználóknak, feltéve, hogy a rendszerükön is van olyan MTA-szolgáltatás, amely elfogadja a hálózati kapcsolatokat. A 25-ös TCP port az SMTP-kommunikáció szabványos portja, de más portok is használhatók a használt hitelesítési és/vagy titkosítási sémától függően (ha van ilyen).

Ha eltekintünk a távoli postafiókokhoz való hozzáféréssel kapcsolatos topológiáktól, akkor egy egyszerű Linux felhasználói fiókok közötti e-mail csererhálózat is megvalósítható, amennyiben minden hálózati csomópont rendelkezik egy aktív MTA-val, amely képes a következő feladatok elvégzésére:

- Fenntartja az elküldendő üzenetek kimeneti várólistáját. Minden egyes sorbaállított üzenet esetében a helyi MTA a címzett címéről értékeli a cél MTA-t.
- Kommunikáció távoli MTA-daemonokkal SMTP használatával. A helyi MTA-nak képesnek kell lennie az SMTP (Simple Mail Transfer Protocol) használatára a TCP/IP stack-en keresztül, hogy üzeneteket fogadjon, küldjön és átirányítson más távoli MTA daemonoktól/daemonokhoz.

- Minden helyi felhasználói fiókhoz egyedi postafiók fenntartása. Az MTA általában *mbox* formátumban tárolja az üzeneteket: egyetlen szöveges fájl, amely az összes e-mail üzenetet sorban tartalmazza.

Általában az e-mail címek egy domainnevet adnak meg helyként, pl. `lpi.org` az `info@lpi.org` -ban. Ebben az esetben a feladó MTA lekérdezi a DNS-szolgáltatásból a megfelelő MX rekordot. A DNS MX rekord tartalmazza az adott domain e-mailjét kezelő MTA IP-címét. Ha ugyanannak a domainnek egynél több MX rekordja van megadva a DNS-ben, az MTA-nak meg kell próbálnia felvenni velük a kapcsolatot a prioritási értéküknek megfelelően. Ha a címzett címe nem ad meg domainnevet, vagy a domain nem rendelkezik MX rekorddal, akkor a `@` szimbólum utáni részt a cél MTA hostjaként kezeli.

A biztonsági szempontokat figyelembe kell venni, ha az MTA-hostok láthatóak lesznek az interneten lévő hostok számára. Lehetséges például, hogy egy ismeretlen felhasználó a helyi MTA-t arra használja, hogy egy másik felhasználónak adja ki magát, és potenciálisan káros e-maileket küldjön. Az olyan MTA-t, amely vakon továbbít egy e-mailt, *nyitott közvetítőnek* (open relay) nevezzük, ha közvetítőként használható az üzenet valódi feladójának esetleges elrejtésére. Az ilyen visszaélések megelőzése érdekében az ajánlás szerint csak engedélyezett domainekből fogadjunk el kapcsolatokat, és alkalmazzunk biztonságos hitelesítési sémát.

Ezen kívül számos különböző MTA-implementáció létezik Linuxra, amelyek mindegyike olyan speciális szempontokra összpontosít, mint a kompatibilitás, teljesítmény, biztonság stb. Mindazonáltal minden MTA ugyanazokat az alapelveket követi és hasonló funkciókat biztosít.

Linux MTA-k

A Linux rendszerekhez elérhető hagyományos MTA a *Sendmail*, egy nagyon rugalmas, általános célú MTA, amelyet számos Unix-szerű operációs rendszer használ. Néhány más elterjedt MTA a *Postfix*, *qmail* és *Exim*. Egy alternatív MTA választásának fő oka, hogy könnyebben implementálhassunk speciális funkciókat, mivel az egyéni e-mail szerverek konfigurálása a Sendmailben bonyolult feladat lehet. Emellett minden disztribúciónak lehet saját preferált MTA-ja, a legtöbb gyakori beállításhoz megfelelő, előre definiált beállításokkal. Minden MTA a Sendmail cseréjeként szolgál, így minden Sendmail-kompatibilis alkalmazásnak működnie kell, függetlenül attól, hogy milyen MTA-t használunk.

Ha az MTA fut, de nem fogad hálózati kapcsolatokat, akkor csak a helyi gépen tud e-mail üzeneteket kézbesíteni. A `sendmail` MTA esetében az `/etc/mail/sendmail.mc` fájlt úgy kell módosítani, hogy a nem helyi kapcsolatokat is elfogadja. Ehhez a

```
DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```


bejegyzést a megfelelő hálózati címre kell módosítani és újraindítani a szolgáltatást. Egyes Linux disztribúciók, mint például a Debian, konfigurációs eszközöket ajánlhatnak fel, amelyek segítenek az e-mail szerver előre meghatározott, gyakran használt funkciókkal való felállításában.

TIP

A biztonsági problémák miatt a legtöbb Linux disztribúció alapértelmezés szerint nem telepít MTA-t. A leckében bemutatott példák teszteléséhez győződjünk meg arról, hogy minden gépen fut egy MTA, és hogy a 25-ös TCP porton fogadnak kapcsolatokat. Biztonsági okokból a tesztelés során ezeket a rendszereket nem szabad kitenni a nyilvános internetről érkező kapcsolatoknak.

Amint az MTA fut és fogad kapcsolatokat a hálózatról, az új e-mail üzenetek SMTP-parancsokkal kerülnek továbbításra, amelyeket egy TCP-kapcsolaton keresztül küldenek. Az `nc` parancs — egy hálózati segédprogram, amely általános adatokat olvas és ír a hálózaton keresztül — használható SMTP-parancsok közvetlen küldésére az MTA-nak. Ha az `nc` parancs nem elérhető, akkor a *ncat* vagy *nmap-ncat* csomaggal együtt települ, a használt csomagkezelő rendszertől függően. Az SMTP parancsok írása közvetlenül az MTA-nak segít jobban megérteni a protokollt és más általános e-mail koncepciókat, de segíthet a levelek kézbesítési folyamatában felmerülő problémák diagnosztizálásában is.

Ha például az `emma` felhasználó a `lab1.campus` hoston üzenetet akar küldeni a `dave` felhasználónak a `lab2.campus` hoston, akkor az `nc` paranccsal közvetlenül csatlakozhat a `lab2.campus` MTA-hoz, feltéve, hogy az a 25-ös TCP porton figyel:

```
$ nc lab2.campus 25
220 lab2.campus ESMTP Sendmail 8.15.2/8.15.2; Sat, 16 Nov 2019 00:16:07 GMT
HELO lab1.campus
250 lab2.campus Hello lab1.campus [10.0.3.134], pleased to meet you
MAIL FROM: emma@lab1.campus
250 2.1.0 emma@lab1.campus... Sender ok
RCPT TO: dave@lab2.campus
250 2.1.5 dave@lab2.campus... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
Subject: Recipient MTA Test

Hi Dave, this is a test for your MTA.
.
250 2.0.0 xAG0G7Y0000595 Message accepted for delivery
QUIT
221 2.0.0 lab2.campus closing connection
```

A kapcsolat létrejötte után a távoli MTA azonosítja magát, majd készen áll az SMTP-parancsok

fogadására. A példa első SMTP-parancsa, a `HELO lab1.campus`, a `lab1.campus`-t jelöli a csere kezdeményezőjeként. A következő két parancs, a `MAIL FROM: emma@lab1.campus` és az `RCPT TO: dave@lab2.campus` a feladót és a címzettet jelöli. A megfelelő e-mail a `DATA` parancs után kezdődik, és egy ponttal végződik egy sorban. Ha az e-mailhez `subject` (tárgy) mezőt is hozzá szeretnénk adni, annak a `DATA` parancs utáni első sorban kell szerepelnie, ahogy a példában látható. A tárgy mező használata esetén egy üres sornak kell azt elválasztania az e-mail tartalmától. A `QUIT` parancs befejezi a kapcsolatot az MTA-val a `lab2.campus` hoston.

A `lab2.campus` gépen a `dave` felhasználó a `You have a new mail in /var/spool/mail/dave` üzenethez hasonló üzenetet kap, amint belép egy shell munkamenetbe. Ez a fájl tartalmazza az `emma` által küldött nyers e-mail üzenetet, valamint az MTA által hozzáadott fejléceket (header):

```
$ cat /var/spool/mail/dave
From emma@lab1.campus  Sat Nov 16 00:19:13 2019
Return-Path: <emma@lab1.campus>
Received: from lab1.campus (lab1.campus [10.0.3.134])
    by lab2.campus (8.15.2/8.15.2) with SMTP id xAG0G7Y0000595
    for dave@lab2.campus; Sat, 16 Nov 2019 00:17:06 GMT
Date: Sat, 16 Nov 2019 00:16:07 GMT
From: emma@lab1.campus
Message-Id: <201911160017.xAG0G7Y0000595@lab2.campus>
Subject: Recipient MTA Test

Hi Dave, this is a test for your MTA.
```

A `Received:` header azt mutatja, hogy a `lab1.campus` üzenetét közvetlenül a `lab2.campus` fogadta. Alapértelmezés szerint az MTA-k csak a helyi címzetteknek szóló üzeneteket fogadják el. Az alábbi hiba valószínűleg akkor fordul elő, ha `emma` felhasználó megpróbál e-mailt küldeni `henry` felhasználónak a `lab3.campus` hoston, de a `lab2.campus` MTA-t használja a megfelelő `lab3.campus` MTA helyett:

```
$ nc lab2.campus 25
220 lab2.campus ESMTP Sendmail 8.15.2/8.15.2; Sat, 16 Nov 2019 00:31:44 GMT
HELO lab1.campus
250 lab2.campus Hello lab1.campus [10.0.3.134], pleased to meet you
MAIL FROM: emma@lab1.campus
250 2.1.0 emma@lab1.campus... Sender ok
RCPT TO: henry@lab3.campus
550 5.7.1 henry@lab3.campus... Relaying denied
```

Az 5-tel kezdődő SMTP-válaszszámok, mint például a `Relaying denied` (továbbítás megtagadva)

üzenet, hibát jeleznek. Vannak olyan legitim helyzetek, amikor a továbítás elvárt, például amikor az e-maileket küldő és fogadó hostok nincsenek állandóan kapcsolatban: egy köztes MTA-t úgy lehet beállítani, hogy fogadja a más hostoknak szánt e-maileket, és *relay* SMTP-szerverként működik, amely továbbítja az üzeneteket az MTA-k között.

Az a képesség, hogy az e-mail forgalmat közbenső SMTP-szervereken keresztül lehet továbbítani, megakadályozza, hogy közvetlenül a címzett e-mail címe által megjelölt hosthoz csatlakozzanak, ahogyan azt az előző példák mutatják. Ráadásul az e-mail címekben gyakran domainnév szerepel lokációként (a `@` után), így a megfelelő MTA-host tényleges nevét a DNS-en keresztül kell lekérdezni. Ezért ajánlatos a megfelelő célállomás azonosításának feladatát a helyi MTA-ra vagy távoli postafiókok használata esetén a távoli SMTP-szerverre bízni.

A Sendmail a `sendmail` parancsot biztosítja számos, az e-mailekkel kapcsolatos művelet elvégzéséhez, beleértve az új üzenetek összeállításának segítését is. Ez is megköveteli, hogy a felhasználó kézzel írja be az e-mail fejléceket, de barátságosabb módon, mint az SMTP parancsok közvetlen használata esetén. Tehát a megfelelőbb módszer az `emma@lab1.campus` felhasználó számára egy e-mail küldése a `dave@lab2.campus` címre az alábbi lenne:

```
$ sendmail dave@lab2.campus
From: emma@lab1.campus
To: dave@lab2.campus
Subject: Sender MTA Test

Hi Dave, this is a test for my MTA.
.
```

Ebben az esetben is az új sorban, önmagában álló pont fejezi be az üzenetet. Az üzenetet azonnal el kell küldeni a címzettnek, kivéve, ha a helyi MTA nem tudott kapcsolatba lépni a távoli MTA-val. A `mailq` parancs, ha a root futtatja, megjeleníti az összes nem kézbesített üzenetet. Ha például a `lab2.campus` MTA nem válaszolt, akkor a `mailq` parancs felsorolja a nem kézbesített üzenetet és a hiba okát:

```
# mailq
        /var/spool/mqueue (1 request)
-----Q-ID----- --Size-- -----Q-Time----- ------------Sender/Recipient-----------
xAIK3D9S000453       36 Mon Nov 18 20:03 <emma@lab1.campus>
                 (Deferred: Connection refused by lab2.campus.)
                     <dave@lab2.campus>
        Total requests: 1
```

A kimenő levelek várólistájának alapértelmezett helye a `/var/spool/mqueue/`, de a különböző

MTA-k más-más helyet használhatnak a `/var/spool/` mappában. A Postfix például a `/var/spool/postfix/` alatt hoz létre egy mappastruktúrát a várólista kezeléséhez. A `mailq` parancs egyenértékű a `sendmail -bp` paranccsal, és a rendszerben telepített MTA-tól függetlenül jelen kell lennie. A visszafelé kompatibilitás biztosítása érdekében a legtöbb MTA biztosítja ezeket a hagyományos levélkezelő parancsokat.

Ha az elsődleges e-mail célállomás — ha az a domain MX DNS rekordjából származik — nem elérhető, az MTA megpróbál kapcsolatba lépni az alacsonyabb prioritásúakkal (ha vannak ilyenek megadva). Ha egyik sem érhető el, az üzenet a helyi kimeneti várólistán marad, hogy később újra megpróbálhassa elküldeni. Ha így van beállítva, az MTA rendszeresen ellenőrizheti a távoli host elérhetőségét, és újabb kézbesítési kísérletet hajthat végre. Sendmail-kompatibilis MTA használata esetén a `sendmail -q` paranccsal azonnal megpróbálja újra elküldeni az e-mailt.

A Sendmail a beérkező leveleket a megfelelő postaláda tulajdonosáról elnevezett fájlban tárolja, például `/var/spool/mail/dave`. Más MTA-k, mint például a Postfix, a bejövő e-maileket olyan helyeken tárolhatják, mint a `/var/mail/dave`, de a fájl tartalma ugyanaz. A példában a feladó hostján a `sendmail` parancsot használták az üzenet összeállításához, így a nyers üzenetfejlécek azt mutatják, hogy az e-mail további lépéseket tett meg, mielőtt eljutott a végső célállomásra:

```
$ cat /var/spool/mail/dave
From emma@lab1.campus  Mon Nov 18 20:07:39 2019
Return-Path: <emma@lab1.campus>
Received: from lab1.campus (lab1.campus [10.0.3.134])
    by lab2.campus (8.15.2/8.15.2) with ESMTPS id xAIK7clC000432
    (version=TLSv1.3 cipher=TLS_AES_256_GCM_SHA384 bits=256 verify=NOT)
    for <dave@lab2.campus>; Mon, 18 Nov 2019 20:07:38 GMT
Received: from lab1.campus (localhost [127.0.0.1])
    by lab1.campus (8.15.2/8.15.2) with ESMTPS id xAIK3D9S000453
    (version=TLSv1.3 cipher=TLS_AES_256_GCM_SHA384 bits=256 verify=NOT)
    for <dave@lab2.campus>; Mon, 18 Nov 2019 20:03:13 GMT
Received: (from emma@localhost)
    by lab1.campus (8.15.2/8.15.2/Submit) id xAIK0doL000449
    for dave@lab2.campus; Mon, 18 Nov 2019 20:00:39 GMT
Date: Mon, 18 Nov 2019 20:00:39 GMT
Message-Id: <201911182000.xAIK0doL000449@lab1.campus>
From: emma@lab1.campus
To: dave@lab2.campus
Subject: Sender MTA Test

Hi Dave, this is a test for my MTA.
```

Alulról felfelé haladva a `Received:` kezdetű sorok az üzenet útvonalát mutatják. Az üzenetet az

`emma` felhasználó küldte el a `sendmail dave@lab2.campus` paranccsal a `lab1.campus` címen, amint azt az első `Received:` header is jelzi. Ezután, még mindig a `lab1.campus`-on, az MTA használja az ESMTPS-t — az SMTP egy supersetje, amely titkosítási kiterjesztésekkel bővíti azt — az üzenet elküldéséhez a `lab2.campus`-on lévő MTA-nak, ahogyan azt az utolsó (felső) `Received:` header mutatja.

Az MTA befejezi a munkáját, miután az üzenet elmentésre került a felhasználó postaládájába. Gyakori, hogy van valamilyen e-mail szűrés, például spamblokkoló vagy a felhasználó által meghatározott szűrési szabályok érvényesítése. Ezeket a feladatokat harmadik féltől származó alkalmazások hajtják végre, az MTA-val együttműködve. Az MTA például meghívhatja a *SpamAssassin* segédprogramot, hogy annak szövegelemző funkciói segítségével jelölje meg a gyanús üzeneteket.

Lehetséges, hogy nem kifejezetten kényelmes a postafiókfájl közvetlen olvasása, ezért javasoljuk, hogy használjunk helyette valamelilyen e-mail kliensprogramot (pl. Thunderbird, Evolution vagy KMail), amely elemzi a fájlt és megfelelően kezeli az üzeneteket. Az ilyen programok extra funkciókat is tartalmaznak, például parancsikonokat a gyakori műveletekhez, almappákat a postaládán belül, stb.

A `mail` parancs és a Mail User Agents (MUA)

Lehetőség van arra, hogy egy e-mailt közvetlenül a nyers formátumában írjunk meg, de sokkal praktikusabb egy kliensalkalmazás — más néven MUA (*Mail User Agent*) — használata a folyamat felgyorsítása és a hibák elkerülése érdekében. A MUA elvégzi a munkát a háttérben, azaz az e-mail kliens megjeleníti és rendszerezi a kapott üzeneteket, és a felhasználó által az e-mail összeállítása után a megfelelő kommunikációt intézi az MTA-val.

A Mail User Agents-nek számos különböző típusa létezik. Az asztali alkalmazások, mint a *Mozilla Thunderbird* és a Gnome *Evolution* támogatják a helyi és távoli e-mail fiókokat is. Még a *Webmail* interfészek is egyfajta MUA-nak tekinthetők, mivel közvetítik a felhasználó és a mögöttes MTA közötti interakciót. Az e-mail kliensek azonban nem korlátozódnak a grafikus felületekre: a konzolos e-mail klienseket széles körben használják a grafikus felülettel nem integrált postafiókok elérésére és az e-mailhez kapcsolódó feladatok automatizálására shell scripteken belül.

Eredetileg a Unix `mail` parancsát csak a helyi rendszerfelhasználók közötti üzenetváltásra szánták (az első `mail` parancs az 1971-ben kiadott első Unix-kiadásból ered). Amikor a hálózati e-mail üzenetváltás egyre inkább előtérbe került, más programok jöttek létre az új kézbesítési rendszer kezelésére, és fokozatosan felváltották a régi `mail` programot.

Manapság a legáltalánosabb `mail` parancsot a *mailx* csomag biztosítja, amely kompatibilis az összes modern e-mail funkcióval. A legtöbb Linux disztribúcióban a `mail` parancs csak egy

szimbolikus link a `mailx` parancsra. Más implementációk, mint például a *GNU Mailutils* csomag, alapvetően ugyanazokat a funkciókat biztosítják, mint a `mailx`. Van azonban néhány különbség közöttük, különösen a parancssori lehetőségek tekintetében.

A `mail` parancs minden modern változata, függetlenül a megvalósítástól, kétféle üzemmódban működik: *normál módban* (normal mode) és *küldés módban* (send mode). Ha a `mail` parancs argumentumaként egy e-mail címet adunk meg, akkor a parancs küldési módba lép, egyébként normál (olvasási) módba. Normál módban a beérkezett üzenetek egy-egy numerikus indexszel vannak felsorolva, így a felhasználó külön-külön hivatkozhat rájuk, amikor az interaktív promptban parancsokat ír be. A `print 1` parancs például az 1. számú üzenet tartalmának megjelenítésére használható. Az interaktív parancsok rövidíthetők, így az olyan parancsok, mint a `print`, `delete` vagy `reply` helyettesíthetők `p`, `d` vagy `r` betűvel. A `mail` parancs mindig az utoljára kapott vagy az utoljára megtekintett üzenetet veszi figyelembe, ha az üzenet indexszáma elmarad. A `quit` vagy `q` parancs kilép a programból.

A *küldés mód* különösen hasznos automatikus e-mail üzenetek küldéséhez. Használható például arra, hogy e-mailt küldjön a rendszergazdának, ha egy ütemezett karbantartási script nem teljesíti a feladatát. Küldési módban a `mail` a *standard input* tartalmát használja az üzenet törzseként:

```
$ mail -s "Maintenance fail" henry@lab3.campus <<<"The maintenance script failed at `date`"
```

Ebben a példában az `-s` kapcsolót használtuk, azért, hogy az üzenethez egy tárgymezőt is hozzáadjunk. Az üzenet törzsét a *Hereline* átirányítással a standard bemenetre adtuk meg, de egy fájl tartalma vagy egy parancs kimenete is átvezethető lenne a program *stdin*-jébe. Ha a szabványos bemenetre történő átirányítással nem adunk meg tartalmat, akkor a program megvárja, hogy a felhasználó megadja az üzenet törzsét. Ebben az esetben a `Ctrl` + `D` billentyűleütés befejezi az üzenetet. A `mail` parancs azonnal kilép, miután az üzenet a kimenő levelek közé került.

A küldés testreszabása

Alapértelmezés szerint a Linux rendszeren lévő e-mail fiókok a szabványos rendszerfiókokhoz vannak társítva. Például, ha Carol felhasználó bejelentkezési neve `carol` a `lab2.campus` hoston, akkor az e-mail címe `carol@lab2.campus` lesz. A rendszerfiókok és a postafiókok közötti egy-egy kapcsolat kiterjeszthető a legtöbb Linux disztribúció által biztosított szabványos módszerekkel, különösen az `/etc/aliases' fájl által biztosított e-mail-routing mechanizmussal.

Az e-mail alias egy “virtuális” e-mail címzett, akinek a fogadó üzeneteit a rendszer átirányítja a meglévő helyi postafiókokba vagy más típusú üzenettároló vagy -feldolgozási célhelyekre. Az aliasok hasznosak például a `postmaster@lab2.campus` címre küldött üzenetek Carol fiókjába

juttatására, amely egy közönséges helyi postafiók a `lab2.campus` rendszerben. Ehhez a `postmaster: carol` sort hozzá kell adni a `lab2.campus` `/etc/aliases` fájljához. Az `/etc/aliases` fájl módosítása után a `newaliases` parancsot kell lefuttatni az MTA alias adatbázisának frissítéséhez és a változtatások érvényesítéséhez. A `sendmail -bi` vagy a `sendmail -I` parancsok is használhatók az aliasok adatbázisának frissítésére.

Az aliasokból soronként egyet adunk meg, `<alias>: <célállomás>` formátumban. A szokásos helyi postafiókokon kívül, amelyeket a megfelelő felhasználónév jelez, más célpontok is megadhatók:

- Egy fájl teljes elérési útja (`/` karakterrel kezdődik). A megfelelő aliasra küldött üzenetek hozzá lesznek fűzve a fájlhoz.
- Egy olyan parancs, ami feldolgozza az üzenetet. A `<cél>`-nak pipe karakterrel kell kezdődnie, és ha a parancs speciális karaktereket (például szóközöket) tartalmaz, akkor dupla idézőjelek közé kell tenni. Például a `subscribe: |subscribe.sh` alias a `lab2.campus`-ban a `subscribe@lab2.campus` címre küldött összes üzenetet a `subscribe.sh` parancs szabványos bemenetére továbbítja. Ha a sendmail *korlátozott shell módban* (restricted shell mode) fut, akkor az engedélyezett parancsoknak — vagy a rájuk mutató linkeknek — az `/etc/smrsh/` mappában kell lenniük.
- Egy include fájl. Egy aliashoz több cél is tartozhat (vesszővel elválasztva), így célszerűbb lehet ezeket egy külső fájlban tartani. Az `:include:` kulcsszónak meg kell jelölnie a fájl elérési útját, mint az `:include:/var/local/destinations`.
- Külső cím. Az aliasok külső e-mail címekre is továbbíthatják az üzeneteket.
- Egy másik alias.

Egy jogosulatlan helyi felhasználó aliasokat adhat meg saját e-mailjeihez a saját mappájában található `.forward` fájl szerkesztésével. Mivel az aliasok csak a felhasználó saját postafiókjára vonatkozik, csak a `<cél>` rész megadása szükséges. Ha például az összes bejövő e-mailt egy külső címre szeretné továbbítani, a `dave` felhasználó a `lab2.campus` fájlban létrehozhatja a következő `~/.forward` fájlt:

```
$ cat ~/.forward
emma@lab1.campus
```

A `dave@lab2.campus` címre küldött összes e-mailt az `emma@lab1.campus` címre továbbítja. Az `/etc/aliases` fájlhoz hasonlóan más átirányítási szabályok is hozzáadhatók a `.forward` fájlhoz, soronként egy. Ennek ellenére a `.forward` fájlt csak a tulajdonosa írhatja, és nem szükséges futtatni a `newaliases` parancsot a módosítás után. A ponttal kezdődő fájlok nem jelennek meg a normál fájllistában, ami miatt a felhasználó figyelmen kívül hagyhatja az aktív aliasokat. Ezért az

e-mail továbbítás esetleges problémái esetén fontos ellenőrizni, hogy a fájl egyáltalán létezik-e.

Gyakorló feladatok

1. További kapcsolók és argumentumok nélkül a `mail henry@lab3.campus` parancs beviteli módba lép, így a felhasználó beírhatja az üzenetet a `henry@lab3.campus` címre. Az üzenet befejezése után melyik billentyűleütés zárja be a beviteli módot és küldi el az e-mailt?

2. Melyik parancsot hajthatja végre a root felhasználó a helyi rendszerről származó, kézbesítetlen üzenetek listázásához?

3. Hogyan használhatja egy nem privilegizált felhasználó a standard MTA módszert arra, hogy minden bejövő levelét automatikusan továbbítsa a `dave@lab2.campus` címre?

Gondolkodtató feladatok

1. A `mailx` által biztosított `mail` parancs használatával melyik paranccsal küldhetnénk üzenetet az `emma@lab1.campus` címre úgy, hogy az e-mail törzse a `uname -a` parancs kimenete, a csatolmány pedig a `logs.tar.gz` fájl?

2. Az e-mail szolgáltatás adminisztrátora szeretné figyelni a hálózaton keresztüli e-mail-átvitelt, de nem akarja tesztüzenetekkel telezsúfolni a postafiókját. Hogyan tud ez a rendszergazda úgy beállítani egy rendszerszintű e-mail aliast, hogy a `teszt` felhasználónak küldött összes e-mailt átirányítsa a `/dev/null` fájlba?

3. A `newaliases` mellett milyen paranccsal lehetne frissíteni az aliasok adatbázisát, miután újabbat adtunk hozzá az `/etc/aliases` fájlhoz?

Összefoglalás

Ez a lecke a Mail Transfer Agents szerepével és használatával foglalkozik a Linux rendszerekben. Az MTA szabványos módszert nyújt a felhasználói fiókok közötti kommunikációhoz, és más szoftverekkel kombinálva extra funkciókat biztosíthat. A lecke a következő témákat tárgyalta:

- Fogalmak az e-mailekkel kapcsolatos technológiákról, postafiókokról és protokollokról.
- Hogyan váltanak egymással üzeneteket a Linux MTA-k a hálózaton keresztül.
- Konzolos e-mail kliensek és MUA-k (Mail User Agents).
- Helyi e-mail aliasing és továbbítás.

A bemutatott technológiák, parancsok és procedúrák a következők voltak:

- SMTP és kapcsolódó protokollok.
- Linuxhoz elérhető MTA-k: Postfix, qmail, Exim.
- MTA és MUA parancsok: `sendmail` és `mail`.
- Adminisztrációs fájlok és parancsok: `mailq`, `/etc/aliases`, `newaliases`, `~/.forward`.

Válaszok a gyakorló feladatokra

1. További kapcsolók és argumentumok nélkül a `mail henry@lab3.campus` parancs beviteli módba lép, így a felhasználó beírhatja az üzenetet a `henry@lab3.campus` címre. Az üzenet befejezése után melyik billentyűleütés zárja be a beviteli módot és küldi el az e-mailt?

   A `Ctrl` + `D` billentyűkombináció lenyomása a program bezárását és az e-mail elküldését eredményezi.

2. Melyik parancsot hajthatja végre a root felhasználó a helyi rendszerről származó, kézbesítetlen üzenetek listázásához?

   A `mailq` vagy `sendmail -bp` parancs.

3. Hogyan használhatja egy nem privilegizált felhasználó a standard MTA módszert arra, hogy minden bejövő levelét automatikusan továbbítsa a `dave@lab2.campus` címre?

   A felhasználónak hozzá kell adnia a `dave@lab2.campus`-t a `~/.forward`-hoz.

Válaszok a gondolkodtató feladatokra

1. A `mailx` által biztosított `mail` parancs használatával melyik paranccsal küldhetnénk üzenetet az `emma@lab1.campus` címre úgy, hogy az e-mail törzse a `uname -a` parancs kimenete, a csatolmány pedig a `logs.tar.gz` fájl?

   ```
   uname -a | mail -a logs.tar.gz emma@lab1.campus
   ```

2. Az e-mail szolgáltatás adminisztrátora szeretné figyelni a hálózaton keresztüli e-mail-átvitelt, de nem akarja tesztüzenetekkel telezsúfolni a postafiókját. Hogyan tud ez a rendszergazda úgy beállítani egy rendszerszintű e-mail aliast, hogy a `teszt` felhasználónak küldött összes e-mailt átirányítsa a `/dev/null` fájlba?

   A `test: /dev/null` sor a `/etc/aliases`-ben minden, `test` helyi postafiókjába küldött levelet a `/dev/null` fájlba irányít át.

3. A `newaliases` mellett milyen paranccsal lehetne frissíteni az aliasok adatbázisát, miután újabbat adtunk hozzá az `/etc/aliases` fájlhoz?

   A `sendmail -bi` vagy `sendmail -I` parancs.

108.4 Nyomtatók menedzselése és nyomtatás

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 102, Objective 108.4

Súlyozás

2

Kulcsfontosságú ismeretek

- Alapvető CUPS-konfiguráció (helyi és távoli nyomtatókhoz)
- A felhasználói nyomtatási várólisták kezelése
- Általános nyomtatási problémák elhárítása
- Jobok hozzáadása és eltávolítása a konfigurált nyomtatósorokból

A használt fájlok, kifejezések és segédprogramok listája

- CUPS konfigurációs fájlok, eszközök és segédprogramok
- `/etc/cups/`
- lpd legacy interfész (`lpr`, `lprm`, `lpq`)

108.4 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	108 Alapvető rendszerszolgáltatások
Fejezet:	108.4 Nyomtatók menedzselése és nyomtatás
Lecke:	1/1

Bevezetés

A számítógépek megjelenése által előidézett “papírmentes társadalomról” szóló nyilatkozatok a mai napig hamisnak bizonyultak. Sok szervezet még mindig a nyomtatott, vagy “hard copy” információs oldalakra támaszkodik. Ezt szem előtt tartva láthatjuk, mennyire fontos, hogy egy számítógép-felhasználó tudja, hogyan nyomtathat egy rendszerből, valamint azt is, hogy egy rendszergazdának tudnia kell azt, hogy hogyan kell fenntartani a számítógép nyomtatókkal való együttműködési képességét.

A Linuxon, valamint számos más operációs rendszeren a *Common Unix Printing System* (CUPS) szoftvercsomag teszi lehetővé a nyomtatást és a nyomtatók kezelését. Az alábbiakban egy nagyon leegyszerűsített vázlatban bemutatjuk, hogyan kerül kinyomtatásra egy fájl Linuxon a CUPS segítségével:

1. A felhasználó megad egy nyomtatandó fájlt.
2. A CUPS daemon, a `cupsd`, ezután *besorolja* (spool) a nyomtatási feladatot. A nyomtatási feladatot a CUPS egy feladatszámmal és a nyomtatandó dokumentum nevével látja el, valamint azzal az információval, hogy melyik nyomtatási sorban van a feladat.

3. A CUPS telepített *szűrőket* használ a nyomtató által használható formázott fájl létrehozásához.
4. A CUPS ezután elküldi az újraformázott fájlt a nyomtatónak nyomtatásra.

Ezeket a lépéseket részletesebben is megnézzük, valamint azt, hogyan telepíthetünk és kezelhetünk nyomtatót Linuxon.

A CUPS szolgáltatás

A legtöbb asztali Linux telepítésével együtt a CUPS csomagok is felkerülnek a rendszerre. A minimalista Linuxok esetén a disztribúciótól függően előfordulhat, hogy a CUPS csomagok nincsenek telepítve. Egy egyszerű CUPS telepítés Debian rendszerre a következő módon végezhető el:

```
$ sudo apt install cups
```

Fedora rendszereken a telepítési folyamat ugyanilyen egyszerű. A Fedora és más Red Hat-alapú disztribúciókon a telepítés után manuálisan kell elindítani a CUPS szolgáltatást:

```
$ sudo dnf install cups
...
$ sudo systemctl start cups.service
```

A telepítés befejezése után a `systemctl` parancs segítségével ellenőrizhetjük, hogy a CUPS szolgáltatás fut-e:

```
$ systemctl status cups.service
● cups.service - CUPS Scheduler
   Loaded: loaded (/lib/systemd/system/cups.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2020-06-25 14:35:47 EDT; 41min ago
     Docs: man:cupsd(8)
 Main PID: 3136 (cupsd)
    Tasks: 2 (limit: 1119)
   Memory: 3.2M
   CGroup: /system.slice/cups.service
           ├─3136 /usr/sbin/cupsd -l
           └─3175 /usr/lib/cups/notifier/dbus dbus://
```

Mint sok más Linux-daemon, a CUPS is konfigurációs fájlokra támaszkodik a működésekor. Az alábbiakban felsoroljuk a rendszergazda számára legfontosabbakat:

`/etc/cups/cupsd.conf`

Ez a fájl magának a CUPS szolgáltatásnak a konfigurációs beállításait tartalmazza. A CUPS konfigurációs fájlja nagyon hasonlít az Apache webszerver konfigurációs fájljára, így ha azt már láttuk, akkor ismerős lehet az általa használt szintaxis. A `cupsd.conf` fájl olyan beállításokat tartalmaz, mint például a rendszerben használt különböző nyomtatási várólistákhoz való hozzáférés szabályozása, a CUPS webes felület engedélyezése, valamint a daemon által használt naplózás szintje.

`/etc/printcap`

Ez az a legacy fájl, amelyet a CUPS megjelenése előtt az LPD (*Line Printer Daemon*) protokoll használt. A CUPS a visszamenőleges kompatibilitás érdekében továbbra is létrehozza ezt a fájlt a rendszereken és gyakran a `/run/cups/printcap`-ra mutató szimbolikus link. A fájl minden egyes sora egy olyan nyomtatót tartalmaz, amelyhez a rendszernek hozzáférése van.

`/etc/cups/printers.conf`

Ez a fájl tartalmazza a CUPS-rendszer által használni kívánt nyomtatókat. Minden egyes nyomtató és a hozzá tartozó nyomtatási sor ebben a fájlban egy `<Printer></Printer>` keretbe van foglalva. Ez a fájl tartalmazza az `/etc/printcap` állományban található egyedi nyomtatólistákat.

WARNING Az `/etc/cups/printers.conf` állományt nem szabad módosítani a parancssorban, amíg a CUPS szolgáltatás fut!

`/etc/cups/ppd/`

Ez nem egy konfigurációs fájl, hanem egy mappa, amely a *PostScript Printer Description* (PPD) fájlokat tartalmazza az ezeket használó nyomtatók számára. Minden nyomtató működési képességeit egy PPD fájlban tárolja (a kiterjesztés `.ppd`). Ezek egyszerű szöveges fájlok, és egy meghatározott formátumot követnek.

A CUPS szolgáltatás ugyanúgy használja a logolást, mint az Apache 2 szolgáltatás. A logfájlok a `/var/log/cups/'-ben tárolódnak, és tartalmaznak egy `access_log'-ot, egy `page_log'-ot és egy `error_log'-ot. Az `access_log` a CUPS webes felületéhez való hozzáférést, valamint az azon belül végrehajtott műveleteket, például a nyomtatók kezelését tartja nyilván. A `page_log` a CUPS telepítés által kezelt nyomtatási sorokba beküldött nyomtatási feladatokat követi nyomon. Az `error_log` a sikertelen nyomtatási feladatokról szóló üzeneteket és a webes felület által rögzített egyéb hibákat tartalmazza.

A következőkben a CUPS szolgáltatás kezeléséhez használt eszközöket és segédprogramokat fogjuk megvizsgálni.

A webes felület használata

Mint korábban említettük, az `/etc/cups/cupsd.conf` konfigurációs fájl határozza meg, hogy a CUPS rendszer webes felülete engedélyezve van-e. A konfigurációs opció így néz ki:

```
# Web interface setting...
WebInterface Yes
```

Ha a webes felület engedélyezve van, akkor a CUPS egy böngészőből kezelhető az alapértelmezett URL-címen: `http://localhost:631`. Alapértelmezés szerint a rendszer egy felhasználója megtekintheti a nyomtatókat és a nyomtatási várólistákat, de a konfiguráció bármilyen módosításához root hozzáféréssel rendelkező felhasználónak kell hitelesítenie magát a webszolgáltatással. Az `/etc/cups/cupsd.conf` fájlban a rendszergazdai képességekhez való hozzáférés korlátozására szolgáló konfigurációs rész a következőképp néz ki:

```
# All administration operations require an administrator to authenticate...
<Limit CUPS-Add-Modify-Printer CUPS-Delete-Printer CUPS-Add-Modify-Class CUPS-Delete-Class
CUPS-Set-Default>
  AuthType Default
  Require user @SYSTEM
  Order deny,allow
</Limit>
```

Az alábbiakban kifejtjük ezeket az opciókat:

`AuthType Default`
: alapszintű hitelesítési kérést fog használni, ha egy művelet root hozzáférést igényel.

`Require user @SYSTEM`
: jelzi, hogy a művelethez rendszergazdai jogosultságokkal rendelkező felhasználóra van szükség. Ez megváltoztatható `@groupname`-re, ahol a `groupname` tagjai adminisztrálhatják a CUPS szolgáltatást, vagy az egyes felhasználók megadhatók egy listával, mint például a `Require user carol, tim`.

`Order deny,allow`
: hasonlóan működik, mint az Apache 2 konfigurációs opciója, ahol a művelet alapértelmezés szerint meg van tagadva, hacsak egy felhasználó (vagy egy csoport tagja) nincs hitelesítve.

A CUPS webes felülete letiltható, ha először leállítjuk a CUPS szolgáltatást, a `WebInterface` opciót `Yes`-ről `No`-ra változtatjuk, majd újraindítjuk a CUPS szolgáltatást.

A CUPS webes felülete úgy épül fel, mint egy egyszerű weboldal, a CUPS rendszer különböző részeihez tartozó navigációs fülekkel. A webes felület fülei a következők:

Home

A kezdőlapon megjelenik a CUPS aktuálisan telepített verziója. A CUPS-t olyan szakaszokra bontja, mint például:

CUPS for Users

A CUPS leírása, parancssori lehetőségek a nyomtatókkal és nyomtatási várólistákkal való munkához, valamint a CUPS felhasználói fórumának linkje.

CUPS for Administrators

A felületen linkeket biztosít a nyomtatók telepítéséhez és kezeléséhez, valamint a hálózati nyomtatókkal való munkával kapcsolatos információkhoz.

CUPS for Developers

A CUPS fejlesztéséhez, valamint a nyomtatók PPD-fájljainak létrehozásához biztosít linkeket.

Administration

Az adminisztrációs oldal szintén szekciókra van osztva:

Printers

Itt a rendszergazda új nyomtatókat adhat hozzá a rendszerhez, megkeresheti a rendszerhez csatlakoztatott nyomtatókat, és kezelheti a már telepítetteket..

Classes

Az osztályok olyan mechanizmusok, amelyek segítségével a nyomtatókat meghatározott házirendekkel rendelkező csoportokba lehet felvenni. Egy osztály például tartalmazhatja olyan nyomtatók csoportját, amelyek egy épület egy adott emeletéhez tartoznak, és amelyekről csak egy adott részleg felhasználói nyomtathatnak. Egy másik osztály korlátozhatja, hogy egy felhasználó hány oldalt nyomtathat. Az osztályok nem alapértelmezés szerint jönnek létre a CUPS telepítésénél, azokat a rendszergazdának kell definiálnia. A CUPS webes felületének ez az a része, ahol új osztályok hozhatók létre és kezelhetők.

Jobs

Itt a rendszergazda megtekintheti az összes nyomtatási feladatot, amely jelenleg sorban áll az összes olyan nyomtatón, amelyet ez a CUPS telepítés kezel.

Server

Itt a rendszergazda módosíthatja az `/etc/cups/cupsd.conf` fájlt. További konfigurációs lehetőségek is rendelkezésre állnak jelölőnégyzeteken keresztül, például a CUPS-telepítéshez csatlakoztatott nyomtatók hálózaton való megosztásának engedélyezése, a fejlett hitelesítés és a nyomtatók távoli felügyeletének engedélyezése.

Classes

Ha a rendszerben vannak nyomtatóosztályok konfigurálva, akkor azok ezen az oldalon lesznek felsorolva. Minden nyomtatóosztályban lehetőség van az osztályba tartozó összes nyomtató egyidejű kezelésére, valamint az adott osztályba tartozó nyomtatókon várakozó összes feladat megtekintésére.

Help

Ez a fül a rendszerre telepített CUPS összes elérhető dokumentációjának linkjeit tartalmazza.

Jobs

Ez a fül lehetővé teszi az egyes nyomtatási feladatok keresését, valamint a szerver által kezelt összes aktuális nyomtatási feladat listázását.

Printers

Ezen a fülön a rendszer által jelenleg kezelt összes nyomtatót, valamint az egyes nyomtatók állapotának gyors áttekintését találjuk meg. A felsorolt nyomtatókra kattintva a rendszergazda arra az oldalra jut, ahol az adott nyomtatót tovább lehet kezelni. Az ezen a fülön található nyomtatókra vonatkozó információk az `/etc/cups/printers.conf` fájlból származnak.

Nyomtató telepítése

Egy nyomtató hozzáadása a rendszerhez egyszerű folyamat a CUPS webes felületén:

1. Kattintsunk az **Administration** fülre és az **Add Printer** gombra!
2. A következő oldalon különböző beállítási lehetőségek jelennek meg attól függően, hogy a nyomtató hogyan van csatlakoztatva a rendszerhez. Ha helyi nyomtatóról van szó, válasszuk ki a legmegfelelőbb opciót, például azt, hogy a nyomtató melyik porthoz van csatlakoztatva, vagy azt, hogy milyen harmadik féltől származó nyomtatószoftver van telepítve. A CUPS megpróbálja felismerni a hálózathoz csatlakoztatott nyomtatókat is, és azokat is itt jeleníti meg. A hálózati nyomtatóhoz való közvetlen csatlakozási lehetőséget is kiválaszthatjuk attól függően, hogy a nyomtató milyen hálózati nyomtatási protokollokat támogat. Válasszuk ki a megfelelő opciót, és kattintsunk a **Continue** gombra!
3. A következő oldalon megadhatjuk a nyomtató nevét, leírását és helyét (például “hátsó iroda”

vagy “első asztal” stb.). Ha szeretnénk megosztani a nyomtatót a hálózaton keresztül, akkor ezen az oldalon bejelölhetjük az erre vonatkozó jelölőnégyzetet is. A beállítások megadása után kattintsunk a **Continue** gombra!

4. A következő oldalon a nyomtató gyártmánya és modellje választható ki. Ez lehetővé teszi a CUPS számára, hogy a helyileg telepített adatbázisában keresse meg a nyomtatóval használható legmegfelelőbb illesztőprogramokat és PPD-fájlokat. Ha rendelkezésünkre áll a nyomtató gyártója által biztosított PPD-fájl, keressük meg annak helyét, és válasszuk ki itt a használathoz! Ha ez megtörtént, kattintsunk az **Add Printer** gombra!
5. Az utolsó oldalon állíthatjuk be az alapértelmezett beállításokat, például a nyomtató által használt oldalméretet és az oldalra nyomtatott karakterek felbontását. Kattintsunk a **Set Default Options** gombra, és a nyomtató máris telepítve van!

NOTE

A Linux számos asztali verziója rendelkezik különböző eszközökkel, amelyekkel nyomtatót lehet telepíteni. A GNOME és a KDE asztali környezetek saját beépített alkalmazásokkal rendelkeznek, amelyek a nyomtatók telepítésére és kezelésére használhatók. Emellett egyes disztribúciók külön nyomtatókezelő alkalmazásokat is biztosítanak. Ha azonban olyan szerverről van szó, amelyet sok felhasználó fog nyomtatásra használni, a CUPS webes felülete nyújtja a legjobb eszközöket a feladathoz.

A nyomtató a hagyományos LPD/LPR parancsok segítségével is telepíthető. Íme egy példa az `lpadmin` parancs használatára:

```
$ sudo lpadmin -p ENVY-4510 -L "office" -v socket://192.168.150.25 -m everywhere
```

A parancsot részekre bontjuk, hogy szemléltetni tudjuk az itt használt kapcsolókat:

- Mivel a nyomtató hozzáadása a rendszerhez rendszergazdai jogosultságokkal rendelkező felhasználót igényel, az `lpadmin` parancs elé a `sudo` parancsot illesztjük.
- A `-p` kapcsoló a nyomtatási feladatok célállomása. Lényegében egy barátságos név a felhasználó számára, hogy tudja, hová kerülnek a nyomtatási feladatok. Általában megadhatjuk a nyomtató nevét.
- Az `-L` kapcsoló a nyomtató helyét adja meg. Ez nem kötelező, de hasznos, ha több nyomtatót kell kezelniünk különböző helyeken.
- A `-v` kapcsoló a nyomtató eszköz URI-jét adja meg. A CUPS nyomtatási várólistának az eszköz URI-jára van szüksége ahhoz, hogy a nyomtatási feladatokat egy adott nyomtatóra küldje. Példánkban egy hálózati helyet használunk a megadott IP-cím segítségével.
- Az utolsó kapcsoló, az `-m`, az “everywhere” (mindenhol) értékre van állítva. Ez beállítja a

nyomtató modelljét, hogy a CUPS meghatározhassa, melyik PPD-fájlt kell használni. A CUPS modern verzióiban célszerű az “everywhere” opciót használni, hogy a CUPS ellenőrizni tudja az eszköz URI-ját (az előző `-v` opció), hogy automatikusan meghatározza a nyomtatóhoz használandó PPD fájlt. A modern helyzetekben a CUPS egyszerűen az IPP-t használja az alábbiakban ismertetett módon.

Mint korábban említettük, a legjobb, ha hagyjuk, hogy a CUPS automatikusan meghatározza, hogy melyik PPD-fájlt használja egy adott nyomtatóhoz. Az `lpinfo` parancs azonban használható a helyileg telepített PPD-fájlok lekérdezésére, hogy lássuk, melyek állnak rendelkezésre. Csak adjuk meg a telepíteni kívánt nyomtató `--make-and-model` kapcsolóját és az `-m` kapcsolót:

```
$ lpinfo --make-and-model "HP Envy 4510" -m
hplip:0/ppd/hplip/HP/hp-envy_4510_series-hpijs.ppd HP Envy 4510 Series hpijs, 3.17.10
hplip:1/ppd/hplip/HP/hp-envy_4510_series-hpijs.ppd HP Envy 4510 Series hpijs, 3.17.10
hplip:2/ppd/hplip/HP/hp-envy_4510_series-hpijs.ppd HP Envy 4510 Series hpijs, 3.17.10
drv:///hpcups.crv/hp-envy_4510_series.ppd HP Envy 4510 Series, hpcups 3.17.10
everywhere IPP Everywhere
```

Vegyük figyelembe, hogy az `lpinfo` parancs elavult (deprecated). A példában látható, hogy felsorolja, milyen nyomtatóillesztő-programfájlokat használhat egy nyomtató.

WARNING

A CUPS jövőbeli verziói már nem tartalmaznak elavult drivereket, helyettük az IPP (*Internet Printing Protocol*) és a szabványos fájlformátumok használatára összpontosítanak. Az előző parancs kimenete ezt az `everywhere IPP Everywhere` nyomtatási képességgel szemlélteti. Az IPP ugyanazokat a feladatokat tudja elvégezni, mint amire a nyomtatóillesztő-programot használják. Az IPP a CUPS webes felületéhez hasonlóan a 631-es hálózati portot használja a TCP protokollal.

Az alapértelmezett nyomtatót az `lpoptions` paranccsal lehet beállítani. Így, ha a nyomtatási feladatok többségét (vagy mindegyikét) egy adott nyomtatóra küldjük, akkor az `lpoptions` paranccsal megadott nyomtató lesz az alapértelmezett. Csak adjuk meg a nyomtatót a `-d` kapcsolóval együtt:

```
$ lpoptions -d ENVY-4510
```

Nyomtatók menedzselése

A nyomtató telepítése után a rendszergazda a webes felület segítségével kezelheti a nyomtatóhoz rendelkezésre álló beállításokat. A nyomtató kezelésének közvetlenebb módja az `lpadmin`

parancs használata.

Az egyik lehetőség a nyomtató megosztásának lehetővé tétele a hálózaton. Ezt a `printer-is-shared` opcióval lehet elérni, illetve a `-p` opció és a nyomtató megadásával:

```
$ sudo lpadmin -p FRONT-DESK -o printer-is-shared=true
```

A rendszergazda úgy is beállíthatja a nyomtatót, hogy csak bizonyos felhasználóktól fogadjon el nyomtatási feladatokat, az egyes felhasználókat vesszővel elválasztva:

```
$ sudo lpadmin -p FRONT-DESK -u allow:carol,frank,grace
```

Fordítva, meg is lehet tagadni csak bizonyos felhasználók számára a hozzáférést egy adott nyomtatási:

```
$ sudo lpadmin -p FRONT-DESK -u deny:dave
```

A felhasználói csoportok arra is használhatók, hogy engedélyezzék vagy megtagadják a hozzáférést egy nyomtató várólistájához, feltéve, hogy a csoport neve előtt egy “kukac” (@) karakter áll:

```
$ sudo lpadmin -p FRONT-DESK -u deny:@sales,@marketing
```

A nyomtató rendelkezhet error policyval (hibaházirend) is, ha egy feladat nyomtatásával kapcsolatban problémák merülnek fel. A házirendek használatával a nyomtatási feladatot meg lehet szakítani (`abort-job`), vagy egy későbbi időpontban újra meg lehet próbálni kinyomtatni (`retry-job`). További házirendek közé tartozik a nyomtató azonnali leállítása hiba esetén (`stop-printer`), valamint a hiba észlelése utáni azonnali újbóli próbálkozás lehetősége (`retry-current-job`). Íme egy példa, ahol a nyomtató policyje úgy van beállítva, hogy a nyomtatási feladatot megszakítja, ha a `FRONT-DESK` nyomtatón hiba lép fel:

```
$ sudo lpadmin -p FRONT-DESK -o printer-error-policy=abort-job
```

A parancs használatával kapcsolatos további részletekért olvassuk el az `lpadmin` parancs man oldalait az `lpadmin(8)` címen!

Nyomtatási feladatok megadása

Sok asztali alkalmazás lehetővé teszi a nyomtatási feladatok elküldését egy menüpontból vagy a `Ctrl` + `p` billentyűparancs segítségével. Ha olyan Linux rendszerben találjunk magunkat, amely nem használ asztali környezetet, akkor a hagyományos LPD/LPR parancsok segítségével továbbra is küldhetünk fájlokat nyomtatóra.

Az `lpr` ("line printer remote") paranccsal nyomtatási feladatot küldhetünk egy nyomtató várólistájára. A parancs legalapvetőbb formájában csak egy fájlnévre és az `lpr` parancsra van szükség:

```
$ lpr report.txt
```

A fenti parancs a `report.txt` fájlt a rendszer alapértelmezett nyomtatási sorába küldi (ahogyan azt a `/etc/cups/printers.conf` fájl azonosítja).

Ha egy CUPS-ban több nyomtató van telepítve, akkor az `lpstat` paranccsal ki lehet listázni a rendelkezésre álló nyomtatókat a `-p` kapcsolóval, és a `-d` kapcsolóval meg lehet nézni, hogy melyik az alapértelmezett nyomtató:

```
$ lpstat -p -d
printer FRONT-DESK is idle.  enabled since Mon 03 Aug 2020 10:33:07 AM EDT
printer PostScript_oc0303387803 disabled since Sat 07 Mar 2020 08:33:11 PM EST -
    reason unknown
printer ENVY-4510 is idle.  enabled since Fri 31 Jul 2020 10:08:31 AM EDT
system default destination: ENVY-4510
```

Tehát a példánkban a `report.txt` fájl az `ENVY-4510` nyomtatóra kerül elküldésre, mivel ez az alapértelmezett. Ha a fájlt egy másik nyomtatón szeretnénk kinyomtatni, adjuk meg a nyomtatót a `-P` kapcsolóval együtt:

```
$ lpr -P FRONT-DESK report.txt
```

Amikor egy nyomtatási feladatot elküldünk a CUPS-nek, a daemon megfejti, hogy melyik backend a legalkalmasabb a feladat kezelésére. A CUPS különböző nyomtatóillesztőket, szűrőket, hardverport-monitorokat és egyéb szoftvereket használhat a dokumentum megfelelő megjelenítéséhez. Előfordulhat, hogy a dokumentumot nyomtató felhasználónak módosítania kell a dokumentum nyomtatásának *módjának* kinyomtatását. Ezt a feladatot sok grafikus alkalmazás meglehetősen megkönnyíti. Vannak parancssori opciók is, amelyekkel megváltoztatható a

dokumentum nyomtatási módja. Amikor egy nyomtatási feladatot a parancssoron keresztül küldünk el, a `-o` kapcsoló (az "`opciók" szó rövidítése) használható bizonyos kifejezésekkel együtt a dokumentum nyomtatási elrendezésének beállításához. Az alábbiakban a gyakran használt kapcsolók rövid listája látható:

`landscape`

A dokumentumot úgy nyomtatja ki, hogy az oldalt az óramutató járásával megegyező irányban 90 fokkal elforgatja. Az `orientation-requested=4` kapcsolóval ugyanezt az eredményt érjük el.

`two-sided-long-edge`

A nyomtató a dokumentumot portré módban nyomtatja ki a papír mindkét oldalára, feltéve, hogy a nyomtató támogatja ezt a funkciót.

`two-sided-short-edge`

A nyomtató a dokumentumot landscape (fektetett) módban nyomtatja ki a papír mindkét oldalára, feltéve, hogy a nyomtató támogatja ezt a funkciót.

`media`

A nyomtató a megadott méretben nyomtatja ki a dokumentumot. A nyomtatási feladathoz rendelkezésre álló méretek a nyomtatótól függnek, de az alábbiakban felsoroljuk a leggyakoribbakat:

Méret	**Cél**
`A4`	ISO A4
`Letter`	US Letter
`Legal`	US Legal
`DL`	ISO DL Envelope
`COM10`	US #10 Envelope

`collate`

A kinyomtatott dokumentumok csoportosítása. Ez akkor hasznos, ha egy többoldalas dokumentumot többször is kinyomtatunk, mivel ekkor az egyes dokumentumok összes oldala sorrendben kerül kinyomtatásra. Állítsuk ezt a kapcsolót `true`-ra, hogy engedélyezzük, vagy `false`-ra, hogy letiltsuk.

`page-ranges`

Ezzel a kapcsolóval egyetlen nyomtatandó oldal vagy egy dokumentumból kinyomtatandó

oldalak egy adott csoportja választható ki. Például: a nyomtatáshoz szükséges oldalak száma: `-o page-ranges=5-7,9,15`. Ez az 5., 6. és 7. oldalt, majd a 9. és 15. oldalt nyomtatná ki.

`fit-to-page`

Úgy nyomtatja ki a dokumentumot, hogy a fájl a papírra méretezett legyen. Ha a nyomtatandó fájlban nincs információ az oldalméretről, előfordulhat, hogy a nyomtatási feladatot helytelenül méretezi, és a dokumentum egyes részei nem férnek rá a lapra, vagy a dokumentum túl kicsi lesz a lapon.

`outputorder`

A dokumentum nyomtatása `fordított` vagy `normál` sorrendben, hogy a nyomtatás az első oldallal kezdődjön. Ha egy nyomtató az oldalakat lefelé fordítva nyomtatja, akkor az alapértelmezett sorrend `-o outputorder=normal`, míg azok a nyomtatók, amelyek az oldalakat felfelé fordítva nyomtatják, `-o outputorder=reverse` sorrendben nyomtatnak.

A fenti opciókból kiindulva a következő példaparancsot állíthatjuk össze:

```
$ lpr -P ACCOUNTING-LASERJET -o landscape -o media=A4 -o two-sided-short-edge finance-
report.pdf
```

Egy dokumentumból egynél több példány nyomtatható ki a number kapcsoló használatával a következő formátumban: `-#N`, ahol az `N` egyenlő a nyomtatandó példányok számával. Íme egy példa a collate kapcsolóval, ahol egy jelentés hét példányát kell kinyomtatni az alapértelmezett nyomtatón:

```
$ lpr -#7 -o collate=true status-report.pdf
```

Az `lpr` parancs mellett az `lp` parancs is használható. Az `lpr` paranccsal használható kapcsolók közül sokan használhatók az `lp` paranccsal is, de van néhány különbség. Mindenképpen nézzük meg az `lp(1)` man oldalt! Az alábbiakban bemutatjuk, hogyan futtathatjuk az előző `lpr` parancsot az `lp` parancs szintaxisával, miközben a `-d` opcióval megadjuk a célnyomtatót is:

```
$ lp -d ACCOUNTING-LASERJET -n 7 -o collate=true status-report.pdf
```

Nyomtatási feladatok menedzselése

Amint korábban említettük, a nyomtatási sorba beküldött minden egyes nyomtatási feladat kap egy feladatazonosítót a CUPS-tól. A felhasználó az `lpq` paranccsal megtekintheti az általa

beküldött nyomtatási feladatokat. Az `-a` kapcsoló megadásával a CUPS telepítés által kezelt összes nyomtató nomtatási sorát megjeleníti:

```
$ lpq -a
Rank        Owner         Job       File(s)                 Total Size
1st         carol         20        finance-report.pdf      5072 bytes
```

A korábban használt `lpstat` parancsnak van egy olyan kapcsolója is, amellyel a nyomtatási listákat lehet megtekinteni. Az `-o` kapcsoló önmagában megmutatja az összes nyomtatási várólistát, vagy megadhatjuk a lista nevét is:

```
$ lp -o
ACCOUNTING-LASERJET-4                  carol        19456   Wed 05 Aug 2020 04:29:44 PM EDT
```

A nyomtatási feladat azonosítója elé kerül annak a sornak a neve, ahová a feladatot küldték, majd a feladatot elküldő felhasználó neve, a fájl mérete és a küldés időpontja.

Ha egy nyomtatási feladat elakad a nyomtatón, vagy a felhasználó szeretné törölni a nyomtatási feladatát, használhatjuk az `lprm` parancsot az `lpq` parancsból kapott munkaazonosítóval együtt:

```
$ lprm 20
```

A nyomtatási sorban lévő összes feladat egyszerre törölhető, ha csak egy kötőjelet (`-`) adunk meg:

```
$ lprm -
```

Alternatívaként a CUPS `cancel` parancsát is használhatja a felhasználó az aktuális nyomtatási feladat leállítására:

```
$ cancel
```

Egy adott nyomtatási feladatot a nyomtató nevével előtagolt feladat azonosítójával lehet törölni:

```
$ cancel ACCOUNTING-LASERJET-20
```

Egy nyomtatási feladatot át is lehet helyezni egyik nyomtatási sorból a másikba. Ez gyakran hasznos, ha egy nyomtató nem reagál, vagy ha a nyomtatandó dokumentum egy másik nyomtatón

elérhető funkciókat igényel. Vegyük figyelembe, hogy ehhez az eljáráshoz általában emelt jogosultságú felhasználóra van szükség. Az előző példában szereplő nyomtatási feladatot a `FRONT-DESK` nyomtató várólistájába helyezhetjük át:

```
$ sudo lpmove ACCOUNTING-LASERJET-20 FRONT-DESK
```

Nyomtatók eltávolítása

Egy nyomtató eltávolításához gyakran hasznos lehet, ha először felsoroljuk a CUPS szolgáltatás által jelenleg kezelt összes nyomtatót. Ezt az `lpstat` paranccsal lehet megtenni:

```
$ lpstat -v
device for FRONT-DESK: socket://192.168.150.24
device for ENVY-4510: socket://192.168.150.25
device for PostScript_oc0303387803: ///dev/null
```

A `-v` kapcsoló nem csak a nyomtatókat sorolja fel, hanem azt is, hogy hol (és hogyan) vannak csatlakoztatva. Jó gyakorlat, ha először elutasítjuk a nyomtatóra érkező új feladatokat, és megadjuk az okát, hogy a nyomtató miért nem használható. Ezt a következőkkel lehet megtenni:

```
$ sudo cupsreject -r "Printer to be removed" FRONT-DESK
```

Vegyük észre a `sudo` használatát, mivel ehhez a feladathoz emelt jogosultságú felhasználóra van szükség!

A nyomtató eltávolításához az `lpadmin` parancsot használjuk az `-x` kapcsolóval:

```
$ sudo lpadmin -x FRONT-DESK
```

Gyakorló feladatok

1. Egy új nyomtatót telepítettek az `office-mgr` nevű helyi munkaállomásra. Milyen paranccsal lehet ezt a nyomtatót alapértelmezettként beállítani ezen a munkaállomáson?

2. Melyik parancs és kapcsolója használható annak meghatározására, hogy milyen nyomtatók állnak rendelkezésre a munkaállomásról történő nyomtatáshoz?

3. A `cancel` parancs segítségével hogyan távolíthatunk el egy olyan nyomtatási feladatot, aminek 15 az ID-je és ami az `office-mgr` nevű nyomtató sorában ragadt?

4. A nyomtatási feladatot olyan nyomtatóra küldtük, amiben nincs elég papír a teljes fájl kinyomtatásához. Milyen parancsot használhatunk ahhoz, hogy a `FRONT-DESK` nyomtatón nyomtatásra váró 2. azonosítójú nyomtatási feladatot átrakjuk az `ACCOUNTING-LASERJET` nyomtató nyomtatási sorába?

Gondolkodtató feladatok

A disztribúció csomagkezelőjének segítségével telepítsük a `cups` és a `printer-driver-cups-pdf` csomagokat! Vegyük figyelembe, hogy ha Red Hat alapú disztribúciót használunk (például Fedora), akkor a CUPS PDF-illesztőprogram neve `cups-pdf`. Telepítsük a `cups-client` csomagot is a System V típusú nyomtatási parancsok használatához! Ezeket a csomagokat arra fogjuk használni, hogy gyakoroljuk egy CUPS nyomtató kezelését anélkül, hogy fizikailag telepítenénk egy valódi nyomtatót.

1. Ellenőrizzük, hogy a CUPS daemon fut-e, majd azt is, hogy a PDF nyomtató engedélyezve és alapértelmezettre van-e állítva!

2. Futtassunk egy parancsot, amely kinyomtatja az `/etc/services` fájlt! Most már kell, hogy legyen egy `PDF` nevű mappánk a home mappában.

3. Használjunk egy olyan parancsot, amely csak a nyomtatót tiltja le, majd futtassunk egy másik parancsot, amely minden állapotinformációt megjelenít, hogy ellenőrizzük, hogy a PDF nyomtató le van-e tiltva! Ezután próbáljuk meg kinyomtatni az `/etc/fstab` fájl másolatát. Mi történik?

4. Most próbáljuk meg elküldeni az `/etc/fstab` fájl egy példányát a PDF nyomtatónak! Mi történik?

5. Vonjuk vissza a nyomtatási feladatot, majd távolítsuk el a PDF nyomtatót!

Összefoglalás

A CUPS daemon egy széles körben használt platform a helyi és távoli nyomtatókra történő nyomtatáshoz. Bár felváltja a régi LPD protokollt, mégis biztosítja az eszközök visszafelé kompatibilitását.

A leckében az alábbi fájlokat és parancsokat tárgyaltuk:

`/etc/cups/cupsd.conf`

A CUPS szolgáltatás elsődleges konfigurációs fájlja. Ez a fájl szabályozza a CUPS webes felületéhez való hozzáférést is.

`/etc/printcap`

Az LPD által használt legacy fájl, amely minden egyes, a rendszerhez csatlakoztatott nyomtatóhoz tartalmaz egy sort.

`/etc/cups/printers.conf`

A CUPS által a nyomtatóval kapcsolatos információkhoz használt konfigurációs fájl.

A CUPS webes felülete, amely alapértelmezett telepítés esetén a `http://localhost:631` címen található. Ne feledjük, hogy a webes felület alapértelmezett hálózati portja a 631/TCP!

Az alábbi legacy LPD/LPR parancsokról szintén volt szó:

`lpadmin`

Nyomtatók és nyomtatóosztályok telepítésére és eltávolítására szolgál.

`lpoptions`

A nyomtató beállításainak megjelenítésére és a nyomtató beállításainak módosítására szolgál.

`lpstat`

A CUPS telepítéshez csatlakoztatott nyomtatók állapotinformációinak megjelenítésére szolgál.

`lpr`

Nyomtatási feladatok elküldése a nyomtatási sorba.

`lp`

Nyomtatási feladatok elküldése a nyomtatási sorba.

`lpq`

A nyomtatási sorban lévő nyomtatási feladatok felsorolása.

`lprm`

Nyomtatási feladatok törlése azonosító szerint. A feladatok azonosítóját az `lpq` parancs kimenetéből lehet megtudni.

`cancel`

Az `lprm` parancs alternatívája a nyomtatási feladatok törlésére azok azonosítója alapján.

Mindenképpen olvassuk el a következő man oldalakat a cups különböző eszközeiről és segédprogramjairól: `lpadmin(8)`, `lpoptions(1)`, `lpr(1)`, `lpq(1)`, `lprm(1)`, `cancel(1)`, `lpstat(1)`, `cupsenable(8)` és `cupsaccept(8)`! Az online súgó dokumentációjának áttekintése a `http://localhost:631/help` címen szintén ajánlott.

Válaszok a gyakorló feladatokra

1. Egy új nyomtatót telepítettek az `office-mgr` nevű helyi munkaállomásra. Milyen paranccsal lehet ezt a nyomtatót alapértelmezettként beállítani ezen a munkaállomáson?

```
$ lpoptions -d office-mgr
```

2. Melyik parancs és kapcsolója használható annak meghatározására, hogy milyen nyomtatók állnak rendelkezésre a munkaállomásról történő nyomtatáshoz?

```
$ lpstat -p
```

   A `-p` kapcsoló felsorolja az összes elérhető nyomtatót és azt, hogy engedélyezve vannak-e.

3. A `cancel` parancs segítségével hogyan távolíthatunk el egy olyan nyomtatási feladatot, aminek 15 az ID-je és ami az `office-mgr` nevű nyomtató sorában ragadt?

```
$ cancel office-mgr-15
```

4. A nyomtatási feladatot olyan nyomtatóra küldtük, amiben nincs elég papír a teljes fájl kinyomtatásához. Milyen parancsot használhatunk ahhoz, hogy a `FRONT-DESK` nyomtatón nyomtatásra váró 2. azonosítójú nyomtatási feladatot átrakjuk az `ACCOUNTING-LASERJET` nyomtató nyomtatási sorába?

```
$ sudo lpmove FRONT-DESK-2 ACCOUNTING-LASERJET
```

Válaszok a gondolkodtató feladatokra

A disztribúció csomagkezelőjének segítségével telepítsük a `cups` és a `printer-driver-cups-pdf` csomagokat! Vegyük figyelembe, hogy ha Red Hat alapú disztribúciót használunk (például Fedora), akkor a CUPS PDF-illesztőprogram neve `cups-pdf`. Telepítsük a `cups-client` csomagot is a System V típusú nyomtatási parancsok használatához! Ezeket a csomagokat arra fogjuk használni, hogy gyakoroljuk egy CUPS nyomtató kezelését anélkül, hogy fizikailag telepítenénk egy valódi nyomtatót.

1. Ellenőrizzük, hogy a CUPS daemon fut-e, majd azt is, hogy a PDF nyomtató engedélyezve és alapértelmezettre van-e állítva!

   A PDF nyomtató elérhetőségének és státuszának ellenőrzésére az egyik lehetőségünk például az alábbi parancs futtatása:

   ```
   $ lpstat -p -d

   printer PDF is idle.  enabled since Thu 25 Jun 2020 02:36:07 PM EDTi

   system default destination: PDF
   ```

2. Futtassunk egy parancsot, amely kinyomtatja az `/etc/services` fájlt! Most már kell, hogy legyen egy `PDF` nevű mappánk a home mappában.

   ```
   $ lp -d PDF /etc/services
   ```

   Mostantól a PDF mappában lesz a fájl PDF változata.

3. Használjunk egy olyan parancsot, amely csak a nyomtatót tiltja le, majd futtassunk egy másik parancsot, amely minden állapotinformációt megjelenít, hogy ellenőrizzük, hogy a PDF nyomtató le van-e tiltva! Ezután próbáljuk meg kinyomtatni az `/etc/fstab` fájl másolatát. Mi történik?

   ```
   $ sudo cupsdisable PDF
   ```

   Ez letiltja a nyomtatót.

   Ezután futtassuk az `lpstat -t` parancsot, hogy teljes képet kapjunk a nyomtató állapotáról. Ennek az alábbi kimenethez hasonlóan kell kinéznie:

```
$ scheduler is running

system default destination: PDFi

device for PDF: cups-pdf:/

PDF accepting requests since Wed 05 Aug 2020 04:19:15 PM EDTi

printer PDF disabled since Wed 05 Aug 2020 04:19:15 PM EDT -

Paused
```

4. Most próbáljuk meg elküldeni az `/etc/fstab` fájl egy példányát a PDF nyomtatónak! Mi történik?

   Az `lp -d PDF /etc/fstab` parancs kiadása után meg kell kapnunk a feladat azonosítóját. Ha azonban megnézzük a PDF mappát a home mappánkban, az új fájl nincs ott. Ezután az `lpstat -o` paranccsal ellenőrizhetjük a nyomtatási sort, és ott megtalálhatjuk a feladatot.

5. Vonjuk vissza a nyomtatási feladatot, majd távolítsuk el a PDF nyomtatót!

   Az előző `lp` parancs kimenete alapján töröljük a feladatot a `cancel` paranccsal. Például:

   ```
   $ cancel PDF-4
   ```

   Majd futtassuk az `lpstat -o` parancsot, hogy meggyőződjünk arról, hogy valóban törölve lett a feladat.

   Töröljük a PDF nyomtatót az alábbival: `sudo lpadmin -x PDF`, majd így győződjünk meg arról, hogy a nyomtatót törlésre került: `lpstat -a`.

Linux Professional Institute

Témakör 109: Hálózati alapok

109.1 Az internetprotokollok alapjai

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 102, Objective 109.1

Súlyozás

4

Kulcsfontosságú ismeretek

- A hálózati maszkok és a CIDR jelölés megértésének bemutatása
- A privát és a nyilvános "dotted quad" IP-címek közötti különbségek ismerete
- Az általános TCP és UDP portok és szolgáltatások ismerete (20, 21, 22, 23, 25, 53, 80, 110, 123, 139, 143, 161, 162, 389, 443, 465, 514, 636, 993, 995)
- Az UDP, a TCP és az ICMP különbségeinek és főbb jellemzőinek ismerete
- Az IPv4 és az IPv6 közötti főbb különbségek ismerete
- Az IPv6 alapvető jellemzőinek ismerete

A használt fájlok, kifejezések és segédprogramok listája

- `/etc/services`
- IPv4, IPv6
- Subnetting
- TCP, UDP, ICMP

109.1 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	109 Hálózati alapok
Fejezet:	109.1 Az internetprotokollok alapjai
Lecke:	1/2

Bevezetés

A TCP/IP (*Transmission Control Protocol/Internet Protocol*) a számítógépek közötti kommunikáció lehetővé tételére szolgáló protokollok halmaza. A neve ellenére több protokollból áll, mint például IP, TCP, UDP, ICMP, DNS, SMTP, ARP, stb.

IP (Internetprotokoll)

Az IP az a protokoll, amely a host logikai címzéséért felelős, lehetővé téve a csomagok egyik hostról a másikra történő továbbítását. Ehhez a hálózat minden egyes eszközéhez egyedi IP-cím tartozik, és egy eszközhöz több cím is rendelhető.

Az IP protokoll 4. verziójában, amelyet általában IPv4-nek neveznek, a cím egy 32 bitből álló, 4 darab 8 bites csoportra osztott, decimális formában ábrázolt, úgynevezett “dotted quad” csoportból áll. Például:

Bináris formátum (8 bit 4 csoportban)

```
11000000.10101000.00001010.00010100
```

Decimális formátum

`192.168.10.20`

Az IPv4-ben az egyes oktettek értékei 0-tól 255-ig terjedhetnek, ami bináris formátumban a 11111111 értéknek felel meg.

Címosztályok

Elméletben az IP-címeket osztályok szerint különítik el, amelyeket az első oktett tartománya határoz meg az alábbi táblázatban látható módon:

Osztály	Első oktett	Tartomány	Példa
A	1-126	`1.0.0.0` – `126.255.255.255`	`10.25.13.10`
B	128-191	`128.0.0.0` – `191.255.255.255`	`141.150.200.1`
C	192-223	`192.0.0.0` – `223.255.255.255`	`200.178.12.242`

Publikus és privát IP-k

Amint korábban említettük, a kommunikációhoz a hálózaton minden eszközhöz legalább egy egyedi IP-címet kell rendelni. Ha azonban a világon minden, az internetre csatlakozó eszköz egyedi IP-címmel rendelkezne, nem lenne elég IP-cím (v4) mindenki számára. Emiatt kerültek meghatározásra a *privát* IP-címek.

A privát IP-címek olyan IP-címtartományok, amelyeket vállalatok, intézmények, otthonok stb. belső (privát) hálózataiban való használatra tartanak fenn. Ugyanazon a hálózaton belül az IP-címek használata egyedi marad. Azonban ugyanaz a privát IP-cím bármelyik privát hálózaton belül használható.

Így az interneten az adatforgalom nyilvános IP-címeket használ, amelyek felismerhetők és az interneten keresztül továbbíthatók, míg a privát hálózatokon belül ezeket a fenntartott IP-tartományokat használják. A router feladata a forgalom átirányítása a privát hálózatból a nyilvános hálózatba és fordítva.

A privát IP-k osztályok szerint elkülönített tartományai az alábbi táblázatban láthatók:

Osztály	Első oktett	Tartomány	Privát IP-k
A	1-126	`1.0.0.0` – `126.255.255.255`	`10.0.0.0` – `10.255.255.255`
B	128-191	`128.0.0.0` – `191.255.255.255`	`172.16.0.0` – `172.31.255.255`
C	192-223	`192.0.0.0` – `223.255.255.255`	`192.168.0.0` – `192.168.255.255`

A decimális formátum binárissá konvertálása

A téma szempontjából fontos tudni, hogyan kell az IP-címeket bináris és decimális formátumba konvertálni.

A decimális formátumról a binárisra való átváltás egymást követő 2-vel való osztással történik. Példaként a 105-ös értéket a következő lépésekkel alakítjuk át:

1. A 105 osztása 2-vel:

```
105/2
Hányados = 52
Maradék = 1
```

2. A hányadosok szekvenciális osztása 2-vel, amíg a hányados nem lesz 1:

```
52/2
Maradék = 0
Hányados = 26
```

```
26/2
Maradék = 0
Hányados = 13
```

```
13/2
Maradék = 1
Hányados = 6
```

```
6/2
```

```
Maradék = 0
Hányados = 3
```

```
3/2
Maradék = 1
Hányados = 1
```

3. Csoportosítsa az utolsó hányadost az összes osztás maradékával:

```
1101001
```

4. Töltsük fel 0-val balról, amíg a 8 bitet el nem érjük:

```
01101001
```

5. Végül azt kapjuk, hogy a 105-ös érték decimálisan egyenlő a 01101001 értékkel binárisan.

A bináris formátum decimálissá konvertálása

Ebben a példában a `10110000` bináris értéket fogjuk használni.

1. Minden bithez kettős alaphatványú érték tartozik. A hatványok 0-tól kezdődnek, és jobbról balra növekszenek. Ebben a példában a következők lesznek:

1	0	1	1	0	0	0	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

2. Ha a bit 1, akkor hozzárendeljük a megfelelő hatványértéket, ha a bit 0, akkor az eredmény 0.

1	0	1	1	0	0	0	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	0	32	16	0	0	0	0

3. Adjuk össze az értékeket:

   128 + 32 + 16 = 176

4. Tehát a 10110000 bináris érték egyenlő a 176 decimális értékkel.

Netmask

A hálózati maszk (vagy *netmask*) az IP-címmel együtt annak meghatározására szolgál, hogy az IP melyik része képviseli a hálózatot és melyik a hostokat. Formátuma megegyezik az IP-címmel, azaz 32 bit van 4 db 8-as csoportban. Például:

Decimális	Bináris	CIDR
`255.0.0.0`	`11111111.00000000.00000000.00000000`	`8`
`255.255.0.0`	`11111111.11111111.00000000.00000000`	`16`
`255.255.255.0`	`11111111.11111111.11111111.00000000`	`24`

A `255.255.0.0.0` maszkot használva példaként, ez azt jelzi, hogy a hozzá tartozó IP-ben az első 16 bit (2 első decimális) a hálózatot/alhálózatot azonosítja, az utolsó 16 bit pedig a hálózaton belüli hostok egyedi azonosítására szolgál.

A fent említett CIDR (*Classless Inter-Domain Routing*) egy egyszerűsített maszkjelöléshez kapcsolódik, amely a hálózathoz/alhálózathoz tartozó bitek számát (1) jelzi. Ezt a jelölést általában a decimális formátum helyettesítésére használják, például `/24` a `255.255.255.0` helyett.

Érdekes megjegyezni, hogy minden IP-osztálynak van egy szabványos maszkja, az alábbiak szerint:

Osztály	Első oktett	Tartomány	Alapértelmezett maszk
A	1-126	`1.0.0.0` – `126.255.255.255`	`255.0.0.0 / 8`
B	128-191	`128.0.0.0` – `191.255.255.255`	`255.255.0.0 / 16`
C	192-223	`192.0.0.0` – `223.255.255.255`	`255.255.255.0 / 24`

Ez a minta azonban nem jelenti azt, hogy mindig ezt a maszkot fogjuk használni. Bármilyen maszkot használhatunk bármilyen IP-címhez, ahogyan azt alább látni fogjuk.

Lássunk néhány példát az IP-k és a maszkok használatára:

```
192.168.8.12 / 255.255.255.0 / 24
```

Tartomány

`192.168.8.0` - `192.168.8.255`

Hálózati cím

`192.168.8.0`

Broadcast cím

`192.168.8.255`

Hostok

`192.168.8.1` - `192.168.8.254`

Ebben az esetben az IP-cím első 3 számjegye (az első 24 bit) határozza meg a hálózatot, az utolsó számjegy pedig a hostok címét, vagyis a hálózat tartománya a `192.168.8.0`-tól a `192.168.8.255` -ig terjed.

Most már két fontos fogalmat megismertünk: minden hálózat/alhálózat rendelkezik 2 fenntartott címmel, a tartomány első címét *hálózati címnek* nevezzük. Ebben az esetben a `192.168.8.0` az, amely magát a hálózatot/alhálózatot azonosítja. A tartomány utolsó címe a *broadcast address*, ebben az esetben a `192.168.8.255`. Ezt a címet arra használják, hogy ugyanazt az üzenetet (csomagot) elküldjék az adott hálózaton/alhálózaton lévő összes IP-hostnak.

A hálózati és broadcast címeket a hálózaton lévő gépek nem használhatják. Ezért az effektíven konfigurálható IP-címek listája a `192.168.8.1`-től a `192.168.8.254`-ig terjed.

Most egy példa ugyanarra az IP címre, de más maszkkal:

```
192.168.8.12 / 255.255.0.0 / 16
```

Tartomány

`192.168.0.0` - `192.168.255.255`

Hálózati cím

`192.168.0.0`

Broadcast cím

`192.168.255.255`

Hostok

`192.168.0.1` – `192.168.255.254`

Nézzük meg, hogy a különböző maszkok hogyan változtatják meg az azonos hálózaton/alhálózaton belüli IP-k tartományát!

A hálózatok maszkok szerinti felosztása nem korlátozódik az alapértelmezett értékekre (8, 16, 24). Tetszés szerinti felosztást hozhatunk létre, biteket adva vagy töröve a hálózat azonosításakor, létrehozva az új alhálózatokat.

Például:

```
11111111.11111111.11111111.00000000 = 255.255.255.0 = 24
```

Ha a fenti hálózatot 2 részre akarjuk osztani, csak adjunk hozzá egy újabb bitet a hálózat azonosítójához a maszkban, így:

```
11111111.11111111.11111111.10000000 = 255.255.255.128 = 25
```

Az alábbi alhálózataink vannak:

```
192.168.8.0   - 192.168.8.127
192.168.8.128 - 192.168.8.255
```

Ha tovább növeljük a hálózat felosztását:

```
11111111.11111111.11111111.11000000 = 255.255.255.192 = 26
```

Ezt kapjuk:

```
192.168.8.0   - 192.168.8.63
192.168.8.64  - 192.168.8.127
192.168.8.128 - 192.168.8.191
192.168.8.192 - 192.168.8.255
```

Vegyük figyelembe, hogy minden alhálózatban lesznek fenntartott hálózati (a tartományban az első) és broadcast (a tartományban az utolsó) címek, tehát minél jobban fel van osztva a hálózat, annál kevesebb IP-címet tudnak hatékonyan használni a hostok.

A hálózati és a broadcast cím azonosítása

Az IP-cím és a maszk segítségével azonosítani tudjuk a hálózati címet és a broadcast címet, és így meghatározhatjuk a hálózat/alhálózat IP-tartományát.

A hálózati címet az IP-cím és a maszk bináris formátumú "logikai AND" segítségével kapjuk meg. Nézzük meg például a `192.168.8.12` IP és a `255.255.255.192` maszk használatát!

Ha decimális formátumból bináris formátumba konvertáljuk őket, ahogy korábban láttuk, a következőket kapjuk:

```
11000000.10101000.00001000.00001100 (192.168.8.12)
11111111.11111111.11111111.11000000 (255.255.255.192)
```

A "logikai AND" (és) miatt 1 and 1 = 1, 0 and 0 = 0, 1 and 0 = 0, tehát:

```
11000000.10101000.00001000.00001100 (192.168.8.12)
11111111.11111111.11111111.11000000 (255.255.255.192)
11000000.10101000.00001000.00000000
```

Tehát az alhálózat hálózati címe `192.168.8.0`.

Ahhoz, hogy megkapjuk a broadcast címet, azt a hálózati címet kell használnunk, ahol a host címhez tartozó összes bit 1:

```
11000000.10101000.00001000.00000000 (192.168.8.0)
11111111.11111111.11111111.11000000 (255.255.255.192)
11000000.10101000.00001000.00111111
```

A broadcast cím tehát `192.168.8.63`.

Összegzésképp:

```
192.168.8.12 / 255.255.255.192 / 26
```

Tartomány

`192.168.8.0` - `192.168.8.63`

Hálózati cím

`192.168.8.0`

Broadcast cím

`192.168.8.63`

Hostok

`192.168.8.1` – `192.168.8.62`

Alapértelmezett útvonal

Mint eddig láttuk, az azonos logikai hálózaton/alhálózaton belüli gépek közvetlenül az IP protokollon keresztül kommunikálhatnak egymással.

De nézzük az alábbi példát:

Hálózat 1

`192.168.10.0/24`

Hálózat 2

`192.168.200.0/24`

Ebben az esetben a `192.168.10.20` gép nem tud közvetlenül csomagot küldeni a `192.168.200.100` gépnek, mivel különböző logikai hálózatokon vannak.

A kommunikáció lehetővé tételéhez egy routert (vagy routerek csoportját) szükséges használni. Ebben a konfigurációban a routert átjárónak (gateway) is nevezhetjük, mivel átjárót biztosít két hálózat között. Ez az eszköz mindkét hálózathoz hozzáfér, mivel mindkét hálózat IP-jeivel van konfigurálva. Például `192.168.10.1` és `192.168.200.1`, és ezért közvetítő szerepet tölt be a kommunikációban.

Ennek engedélyezéséhez a hálózat minden egyes hostján be kell állítani az úgynevezett *default route*-t (alapértelmezett útvonal). Az alapértelmezett útvonal azt az IP-t jelöli, ahová minden olyan csomagot küldeni kell, amelynek a célpontja olyan IP, amely nem része a host logikai hálózatának.

A fenti példában a `192.168.10.0/24` hálózaton lévő gépek alapértelmezett útvonala a `192.168.10.1` IP lesz, ami a router/gateway IP címe, míg a `192.168.200.0/24` hálózaton lévő gépek alapértelmezett útvonala a `192.168.200.1`.

Az alapértelmezett útvonal arra is szolgál, hogy egy privát hálózat (LAN) gépei egy routeren keresztül hozzáférjenek az internethez (WAN).

Gyakorló feladatok

1. A `172.16.30.230` IP és a `255.255.255.224` netmask használatával azonosítsuk:

A netmask CIDR-jelölését	
A hálózati címet	
A broadcast címet	
Az ebben az alhálózatban a hostok számára használható IP-k számát	

2. Melyik beállítás szükséges egy hoston ahhoz, hogy engedélyezze az IP-kommunikációt egy másik logikai hálózatban lévő hosttal?

Gondolkodtató feladatok

1. Miért nem tartoznak az A, B vagy C IP-címosztályok közé a `127`-el kezdődő és a `224` utáni IP-címtartományok?

2. Az IP csomag egyik nagyon fontos mezője a TTL (*Time To Live*). Mi a funkciója ennek a mezőnek, és hogyan működik?

3. Mi a NAT funkciója és mikor használjuk?

 | |

Összefoglalás

Ebben a leckében az IPv4 protokoll főbb fogalmairól volt szó, amely a hálózaton lévő hostok közötti kommunikációjáért felelős.

Tanulmányoztuk azokat a fő műveleteket is, amelyeket a szakembernek ismernie kell ahhoz, hogy az IP-ket különböző formátumokba konvertálja, és képes legyen elemezni, valamint elvégezni a hálózatok és alhálózatok logikai konfigurációit.

Az alábbi témákról volt szó:

- IP címek osztályai
- Publikus és privát IP-k
- Hogyan konvertáljunk IP-ket decimálisról bináris formátumra, és vice versa
- A hálózati maszk (netmask)
- Hogyan azonosítsuk a hálózati és a broadcast címeket az IP-ből és a netmaskból
- Alapértelmezett útvonal (default route)

Válaszok a gyakorló feladatokra

1. A `172.16.30.230` IP és a `255.255.255.224` netmask használatával azonosítsuk:

A netmask CIDR-jelölését	`27`
A hálózati címet	`172.16.30.224`
A broadcast címet	`172.16.30.255`
Az ebben az alhálózatban a hostok számára használható IP-k számát	`30`

2. Melyik beállítás szükséges egy hoston ahhoz, hogy engedélyezze az IP-kommunikációt egy másik logikai hálózatban lévő hosttal?

   Alapértelmezett útvonal (default route)

Válaszok a gondolkodtató feladatokra

1. Miért nem tartoznak az A, B vagy C IP-címosztályok közé a `127`-el kezdődő és a `224` utáni IP-címtartományok?

   A `127`-el kezdődő tartomány a tesztekhez és a folyamatok közötti belső kommunikációhoz használt loopback címek számára van fenntartva, mint például a `127.0.0.1` cím. Ezenkívül a `224` feletti címeket sem hostcímként, hanem multicast és egyéb célokra használják.

2. Az IP csomag egyik nagyon fontos mezője a TTL (*Time To Live*). Mi a funkciója ennek a mezőnek, és hogyan működik?

   A TTL a csomag élettartamát határozza meg. Ezt egy számlálóval valósítják meg, amelyben a forrásnál meghatározott kezdeti értéket csökkentik minden egyes gatewayben/routerben, amelyen a csomag áthalad (ezt “hop”-nak is nevezik). Ha ez a számláló eléri a 0 értéket, a csomag elvetésre kerül.

3. Mi a NAT funkciója és mikor használjuk?

   A NAT (Network Address Translation, *hálózati címfordítás*) funkció lehetővé teszi, hogy a belső hálózaton lévő, privát IP-címeket használó hostok úgy férjenek hozzá az internethez, mintha közvetlenül csatlakoznának hozzá, a gatewayen használt nyilvános IP-címmel.

109.1 Lecke 2

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	109 Hálózati alapok
Fejezet:	109.1 Az internetprotokollok alapjai
Lecke:	2/2

Bevezetés

Az alfejezet elején láttuk, hogy a TCP/IP stack különböző protokollokból áll. Eddig az IP protokollt tanulmányoztuk, amely lehetővé teszi a gépek közötti kommunikációt IP-címeken, maszkokon, útvonalakon stb. keresztül.

Ahhoz, hogy egy host hozzáférhessen egy másik hoston elérhető szolgáltatáshoz, a hálózati réteg IP-címzési protokollján kívül szükség van egy szállítási rétegbeli protokollra is, például a TCP és UDP protokollokra.

Ezek a protokollok ezt a kommunikációt hálózati portokon keresztül bonyolítják le. Tehát a forrás- és cél-IP meghatározásán kívül a forrás- és célportokat is használni kell a szolgáltatás eléréséhez.

A portot egy 16 bites mező azonosítja, így a lehetséges portok száma 65 535-re korlátozódik. A szolgáltatások (célállomás) az 1-től 1023-ig terjedő portokat használják, amelyeket *privilegizált portoknak* (privileged ports) nevezünk, mivel root hozzáféréssel rendelkeznek a rendszerhez. A kapcsolat forrása az 1024 és 65 535 közötti portok tartományát használja, amelyeket *nem privilegizált portoknak* (non-privileged ports) vagy scoket-portoknak neveznek.

Az egyes szolgáltatástípusok által használt portokat az IANA (*Internet Assigned Numbers*

Authority) szabványosítja és ellenőrzi. Ez azt jelenti, hogy bármely rendszer esetén a 22-es port az SSH szolgáltatásé, a 80-as port a HTTP szolgáltatásé és így tovább.

Az alábbi táblázat tartalmazza a főbb szolgáltatásokat és a hozzájuk tartozó portokat.

Port	**Szolgáltatás**
20	FTP (adat)
21	FTP (kontroll)
22	SSH (Secure Socket Shell)
23	Telnet (Távoli, titkosítás nélküli kapcsolat)
25	SMTP (Simple Mail Transfer Protocol), E-mailek küldése
53	DNS (Domain Name System)
80	HTTP (Hypertext Transfer Protocol)
110	POP3 (Post Office Protocol), E-mailek fogadása
123	NTP (Network Time Protocol)
139	Netbios
143	IMAP (Internet Message Access Protocol), E-mailekhez való hozzáférés
161	SNMP (Simple Network Management Protocol)
162	SNMPTRAP, SNMP értesítések
389	LDAP (Lightweight Directory Access Protocol)
443	HTTPS (Secure HTTP)
465	SMTPS (Secure SMTP)
514	RSH (Remote Shell)
636	LDAPS (Secure LDAP)
993	IMAPS (Secure IMAP)
995	POP3S (Secure POP3)

Linux rendszereken a szabványos szolgáltatási portok a `/etc/services` fájlban vannak felsorolva.

A kívánt célport azonosítása egy kapcsolaton belül az IPv4-cím után a `:` (kettőspont) karakterrel

történik. Így amikor a `200.216.10.15` IP-host által kiszolgált HTTPS-szolgáltatáshoz akar hozzáférni, a kliensnek a kérést a `200.216.10.15:443` célportra kell küldenie.

A fent felsorolt és az összes többi szolgáltatás a megkövetelt jellemzőknek megfelelő szállítási protokollt használ, amelyek közül a TCP és az UDP a legfontosabbak.

Transmission Control Protocol (TCP)

A TCP egy kapcsolatorientált szállítási protokoll. Ez azt jelenti, hogy a kliens a socket porton keresztül, a szolgáltatás pedig a szolgáltatás szabványos portján keresztül létesít kapcsolatot. A protokoll feladata az összes csomag megfelelő kézbesítésének biztosítása, a csomagok integritásának és sorrendjének ellenőrzése, beleértve a hálózati hibák miatt elveszett csomagok újbóli továbbítását is.

Így az alkalmazásnak nem kell megvalósítania ezt az adatáramlás-szabályozást, mivel azt a TCP protokoll már garantálja.

User Datagram Protocol (UDP)

Az UDP kapcsolatot hoz létre a kliens és a szolgáltatás között, de nem ellenőrzi a kapcsolat adatátvitelét. Más szóval nem ellenőrzi, hogy a csomagok nem vesztek-e el, vagy rendben vannak-e stb. A szükséges ellenőrzések megvalósítása az alkalmazás feladata.

Mivel kevesebb az ellenőrzés, az UDP jobb teljesítményt tesz lehetővé az adatáramlásban, ami bizonyos típusú szolgáltatások esetében fontos.

Internet Control Message Protocol (ICMP)

Az ICMP egy hálózati szintű protokoll a TCP/IP stackben, és fő funkciója a hálózati elemek elemzése és ellenőrzése, lehetővé téve például:

- A forgalomszabályozást
- Az elérhetetlen célpontok detektálását
- Az útvonal átirányítást
- A távoli hostok státuszának ellenőrzését

Ezt a protokollt használja a `ping` parancs, amelyet egy másik alfejezetben fogunk tanulmányozni.

IPv6

Eddig az IP protokoll 4-es verziójáról, azaz az IPv4-ről volt szó. Ez volt a szabványos verzió, amelyet minden hálózati és internetes környezetben használtak. Ennek azonban megvannak a maga korlátai, különösen a rendelkezésre álló címek számát illetően, és mivel már most is valóság, hogy minden eszköz valamilyen módon csatlakozik az internethez (lsd. IoT), egyre inkább elterjedt az IP-protokoll 6-os verziójának használata, amelyet általában IPv6-nak hívnak.

Az IPv6 egy sor változást, új implementációt és funkciót, valamint magának a címnek az új megjelenítését hozza magával.

Minden IPv6-cím 128 bitből áll, 8 darab 16 bites csoportra osztva, amelyeket hexadecimális értékekkel ábrázolnak.

Például:

```
2001:0db8:85a3:08d3:1319:8a2e:0370:7344
```

Rövidítések

Az IPv6 bizonyos helyzetekben meghatározza a címek lerövidítésének módját. Nézzük a következő címet:

```
2001:0db8:85a3:0000:0000:0000:0000:7344
```

Az első lehetőség, hogy a `0000` stringeket csak `0`-ra redukáljuk, ami a következőt eredményezi:

```
2001:0db8:85a3:0:0:0:0:7344
```

Ezenkívül a `0` értékű csoportos stringek elhagyhatók az alábbiak szerint:

```
2001:0db8:85a3::7344
```

Ez utóbbi rövidítésfajta azonban csak egyszer szerepelhet a címben. Például:

```
2001:0db8:85a3:0000:0000:1319:0000:7344

2001:0db8:85a3:0:0:1319:0:7344
```

```
2001:0db8:85a3::1319:0:7344
```

IPv6 címtípusok

Az IPv6 a címeket 3 típusba sorolja:

Unicast

Egyetlen hálózati interfészt azonosít. Alapértelmezés szerint a bal oldali 64 bit a hálózatot, a jobb oldali 64 bit pedig az interfészt azonosítja.

Multicast

Hálózati interfészek egy csoportját azonosítja. A multicast címre küldött csomagot az összes olyan interfészre elküldi, amelyik az adott csoporthoz tartozik. Bár hasonló, nem összetévesztendő a broadcasttel, amely nem létezik az IPv6 protokollban.

Anycast

Ez is azonosítja az interfészek egy csoportját a hálózaton, de az *anycast* címre továbbított csomagot csak egy címre fogja továbbítani a csoportból, nem pedig mindenkinek.

Az IPv4 és az IPv6 közötti különbségek

A címen kívül számos más különbség is kimutatható az IP 4. és 6. verziója között. Íme néhány ezek közül:

- A szolgáltatási portok ugyanazokat a szabványokat és protokollokat követik (TCP, UDP), a különbség csak az IP és a portkészlet megjelenítésében van. Az IPv6-ban az IP-címet `[]` (szögletes zárójellel) kell levédeni:

  IPv4

  `200.216.10.15:443`

  IPv6

  `[2001:0db8:85a3:08d3:1319:8a2e:0370:7344]:443`

- Az IPv6 nem pontosan úgy valósítja meg a broadcast funkciót, ahogyan az az IPv4-ben létezik. Azonban ugyanaz az eredmény érhető el, ha a csomagot az `ff02::1` címre küldjük, így a helyi hálózat összes hostjához eljut. Kicsit ahhoz hasonló, mintha az IPv4-en a `224.0.0.1`-t használnánk multicasting célállomásként.
- Az SLAAC (*Stateless Address Autoconfiguration*) funkció révén az IPv6 hostok képesek az önkonfigurációra.

- Az IPv4 TTL (*Time to Live*) mezőjét az IPv6 fejlécében a “Hop Limit” mező váltotta fel.
- Minden IPv6-interfésznek van egy helyi címe, az úgynevezett lokális-link (link-local) cím, amelynek előtagja `fe80::/10`.
- Az IPv6 a *Neighbor Discovery Protocol* (NDP) protokollt valósítja meg, amely hasonló az IPv4 által használt ARP-hez, de sokkal több funkcióval rendelkezik.

Gyakorló feladatok

1. Melyik az SMTP protokoll alapértelmezett portja?

2. Hány különböző port található meg a rendszerben?

3. Melyik szállítási protokoll biztosítja, hogy minden csomagot megfelelően kézbesítsenek, ellenőrizve a csomagok integritását és sorrendjét?

4. Milyen típusú IPv6-cím használható egy csomag elküldésére az összes olyan interfésznek, amely egy hostcsoporthoz tartozik?

Gondolkodtató feladatok

1. Említsünk 4 példát olyan szolgáltatásokra, amelyek alapértelmezés szerint a TCP protokollt használják!

2. Mi a neve annak a mezőnek az IPv6 fejléc csomagjában, amely ugyanazt a TTL erőforrást implementálja az IPv4-ben?

3. Milyen információkat képes felfedezni a Neighbor Discovery Protocol (NDP)?

Összefoglalás

Ez a lecke a TCP/IP stackben használt főbb szállítási protokollokkal és szolgáltatásokkal foglalkozott.

Egy másik fontos téma volt az IP protokoll 6-os verziója, beleértve az IPv6 címeket és a főbb különbségeket az IPv4-hez képest.

Az alábbi témákról volt szó:

- A portszámok és a szolgáltatások közötti összefüggés
- TCP (Transmission Control Protocol)
- UDP (User Datagram Protocol)
- ICMP (Internet Control Message Protocol)
- Az IPv6 cím és a rövidítésének módjai
- IPv6 címtípusok
- Az IPv4 és az IPv6 közötti fő különbségek

Válaszok a gyakorló feladatokra

1. Melyik az SMTP protokoll alapértelmezett portja?

   25

2. Hány különböző port található meg a rendszerben?

   65535

3. Melyik szállítási protokoll biztosítja, hogy minden csomagot megfelelően kézbesítsenek, ellenőrizve a csomagok integritását és sorrendjét?

   TCP

4. Milyen típusú IPv6-cím használható egy csomag elküldésére az összes olyan interfésznek, amely egy hostcsoporthoz tartozik?

   Multicast

Válaszok a gondolkodtató feladatokra

1. Említsünk 4 példát olyan szolgáltatásokra, amelyek alapértelmezés szerint a TCP protokollt használják!

   FTP, SMTP, HTTP, POP3, IMAP, SSH

2. Mi a neve annak a mezőnek az IPv6 fejléc csomagjában, amely ugyanazt a TTL erőforrást implementálja az IPv4-ben?

   Hop Limit

3. Milyen információkat képes felfedezni a Neighbor Discovery Protocol (NDP)?

   Az NDP képes különböző információkat beszerezni a hálózatból, beleértve más csomópontokat, duplikált címeket, útvonalakat, DNS-szervereket, gatewayeket stb.

109.2 Perzisztens hálózatkonfiguráció

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 102, Objective 109.2

Súlyozás

4

Kulcsfontosságú ismeretek

- Az alapvető TCP/IP hostkonfiguráció megértése
- Ethernet és wi-fi hálózati konfiguráció beállítása a NetworkManager segítségével
- A systemd-networkd ismerete

A használt fájlok, kifejezések és segédprogramok listája

- `/etc/hostname`
- `/etc/hosts`
- `/etc/nsswitch.conf`
- `/etc/resolv.conf`
- `nmcli`
- `hostnamectl`
- `ifup`
- `ifdown`

109.2 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	109 Hálózati alapok
Fejezet:	109.2 Perzisztens hálózatkonfiguráció
Lecke:	1/2

Bevezetés

Bármely TCP/IP-hálózatban minden csomópontnak úgy kell konfigurálnia a hálózati adapterét, hogy az megfeleljen a hálózati követelményeknek, különben nem tudnak egymással kommunikálni. Ezért a rendszergazdának meg kell adnia az alapkonfigurációt, hogy az operációs rendszer minden indításkor képes legyen beállítani a megfelelő hálózati interfészt, valamint azonosítani magát és a hálózat alapvető jellemzőit.

A hálózati beállítások nem függnek az operációs rendszerektől, de ez utóbbiaknak saját módszerük van a beállítások tárolására és alkalmazására. A Linux rendszerek az `/etc` mappában található egyszerű szöveges fájlokban tárolt konfigurációkra támaszkodnak a hálózati kapcsolat létrehozásához a rendszerindítás során. Érdemes tudni, hogyan vannak használva ezek a fájlok, hogy elkerüljük a helyi hibás konfiguráció miatti kapcsolatvesztést.

A Network Interface

A *network interface* (hálózati interfész) az a kifejezés, amellyel az operációs rendszer a rendszerhez csatlakoztatott hálózati hardverrel, például egy ethernet vagy wi-fi eszközzel való együttműködésre konfigurált kommunikációs csatornára utal. Kivételt képez ezalól a *loopback*

interfész, amelyet az operációs rendszer akkor használ, ha kapcsolatot kell létesítenie önmagával, de a hálózati interfész fő célja, hogy olyan útvonalat biztosítson, amelyen keresztül helyi adatokat lehet küldeni és távoli adatokat fogadni. Ha a hálózati interfész nincs megfelelően konfigurálva, akkor az operációs rendszer nem tud kommunikálni a hálózat más gépeivel.

A legtöbb esetben a megfelelő interfészbeállítások vagy alapértelmezés szerint vannak megadva, vagy az operációs rendszer telepítése során testre szabhatók. Mindazonáltal ezeket a beállításokat gyakran ellenőrizni vagy akár módosítani kell, ha a kommunikáció nem működik megfelelően, vagy ha az interfész viselkedése testreszabást igényel.

Számos Linux-parancs létezik a rendszerben lévő hálózati interfészek listázására, de nem mindegyik érhető el minden disztribúcióban. Az `ip` parancs azonban a minden Linux disztribúcióhoz mellékelt hálózati eszközök alapkészletének része, és használható a hálózati interfészek listázására. Az interfészek megjelenítésére szolgáló teljes parancs az `ip link show`:

```
$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
default qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp3s5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT
group default qlen 1000
        link/ether 00:16:3e:8d:2b:5b brd ff:ff:ff:ff:ff:ff
```

Ha elérhető, az `nmcli device` parancs is használható:

```
$ nmcli device
DEVICE      TYPE      STATE      CONNECTION
enp3s5      ethernet  connected  Gigabit Powerline Adapter
lo          loopback  unmanaged  --
```

A példákban bemutatott parancsok nem módosítanak semmilyen beállítást a rendszerben, így egy nem privilegizált felhasználó is végrehajthatja őket. Mindkét parancs két hálózati interfészt sorol fel: `lo` (a loopback interfész) és `enp3s5` (egy ethernet interfész).

A Linuxot futtató asztali számítógépek és laptopok általában két vagy három előre definiált hálózati interfésszel rendelkeznek: egy a loopback virtuális interfészhez, a többi pedig a rendszer által talált hálózati hardverhez van rendelve. A Linuxot futtató szerverek és hálózati eszközök viszont több tucat hálózati interfésszel rendelkezhetnek, de ugyanazok az elvek vonatkoznak mindegyikre. Az operációs rendszer által biztosított absztrakció lehetővé teszi a hálózati interfészek beállítását ugyanazon módszerek alkalmazásával, függetlenül a mögöttes hardvertől.

Az interfész mögöttes hardverének részleteinek ismerete azonban hasznos lehet ahhoz, hogy jobban megértsük, mi történik, amikor a kommunikáció nem a várt módon működik. Egy olyan rendszerben, ahol számos hálózati interfész áll rendelkezésre, nem biztos, hogy egyértelmű, hogy melyik felel meg például a wi-fi-nek, és melyik az ethernetnek. Ezért a Linux olyan interfész elnevezési konvenciót használ, amely segít azonosítani, hogy melyik hálózati interfész melyik eszköznek és portnak felel meg.

Interfészek nevei

A régebbi Linux disztribúciók az ethernet hálózati interfészeket `eth0`, `eth1` stb. néven nevezték el, aszerint a sorrend szerint, ahogy a kernel azonosítja az eszközöket. A vezeték nélküli interfészek neve `wlan0`, `wlan1` stb. volt. Ez az elnevezési konvenció azonban nem tisztázza azt a kédést, hogy például az `eth0` interfészhez melyik konkrét ethernet-port tartozik. Attól függően, hogy a hardvert hogyan észlelte a rendszer, még az is előfordulhatott, hogy két hálózati interfész nevet cserélt egy újraindítás után.

Ennek a kétértelműségnek a kiküszöbölésére az újabb Linux-rendszerek kiszámítható elnevezési konvenciót alkalmaznak a hálózati interfészekre, szorosabb kapcsolatot teremtve az interfész neve és a mögöttes hardverkapcsolat között.

A systemd elnevezési sémát használó Linux-disztribúciókban minden interfész neve egy kétkarakteres előtaggal kezdődik, amely az interfész típusát jelöli:

`en`
: Ethernet

`ib`
: InfiniBand

`sl`
: Serial line IP (slip)

`wl`
: Wireless local area network (WLAN)

`ww`
: Wireless wide area network (WWAN)

Az operációs rendszer a következő szabályokat használja a hálózati interfészek elnevezésére és számozására a magasabb prioritástól az alacsonyabb prioritásig:

1. Az interfész elnevezése a BIOS vagy a beágyazott eszközök firmware-je által megadott index után, pl. `eno1`.
2. Az interfész elnevezése a PCI express slot indexe után, ahogyan azt a BIOS vagy a firmware megadja, pl. `ens1`.
3. Az interfész elnevezése a megfelelő buszon lévő címe után, pl. `enp3s5`.
4. Az interfész elnevezése az interfész MAC-címe után, pl. `enx78e7d1ea46da`.
5. Az interfész elnevezése a legacy konvenciót használva, pl. `eth0`.

Helyes az a feltételezés, hogy például az `enp3s5` hálózati interfész azért kapta ezt a nevet, mert nem illett az első két elnevezési módszerhez, ezért helyette a megfelelő busz és slot címét használta a rendszer. A `03:05.0` eszközcím, amelyet az `lspci` parancs kimenetén találunk, elárulja, hogy mi a a társított eszköz:

```
$ lspci | fgrep Ethernet
03:05.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL-8110SC/8169SC Gigabit
Ethernet (rev 10)
```

A hálózati interfészeket maga a Linux kernel hozza létre, de számos parancsot lehet használni a velük való interakcióra. Általában a konfiguráció automatikusan történik, és nincs szükség a beállítások kézi módosítására. Ennek ellenére az interfész nevével meg lehet mondani a kernelnek, hogy szükség esetén hogyan járjon el a konfigurálásban.

Az interfészek menedzselése

Az évek során számos programot fejlesztettek ki a Linux kernel által biztosított hálózati funkciókkal való együttműködésre. Bár a régi `ifconfig` parancs még mindig használható egyszerű interfész-konfigurációk és lekérdezések elvégzésére, mára már elavult, mivel a nem Ethernet interfészek korlátozottan támogatottak. Az `ifconfig` parancsot felváltotta az `ip` parancs, amely a TCP/IP interfészek számos más aspektusát, például útvonalakat és tunneleket (alagutakat) képes kezelni.

Az `ip` parancs számos lehetősége túl sok lehet a legtöbb hétköznapi feladathoz, ezért vannak segédparancsok, amelyek megkönnyítik a hálózati interfészek aktiválását és konfigurálását. Az `ifup` és `ifdown` parancsok az `/etc/network/interfaces` fájlban található interfészdefiníciók alapján konfigurálhatók. Bár ezek a parancsok manuálisan is előhívhatók, általában a rendszer indításakor automatikusan végrehajtódnak.

Az `ifup` és `ifdown` által kezelt összes hálózati interfésznek szerepelnie kell az `/etc/network/interfaces` fájlban. A fájlban használt formátum egyszerű: az `auto` szóval

kezdődő sorok az `ifup -a` kapcsolóval történő végrehajtása esetén felhozandó fizikai interfészek azonosítására szolgálnak. Az interfész nevének ugyanabban a sorban kell követnie az `auto` szót. Az összes `auto` jelzésű interfész a rendszer indításkor a listában szereplő sorrendben jelenik meg.

WARNING

Az `ifup` és `ifdown` által használt hálózati konfigurációs módszerek nem minden Linux disztribúcióban szabványosak. A CentOS például az interfészbeállításokat az `/etc/sysconfig/network-scripts/` mappában lévő egyedi fájlokban tárolja, és az ott használt konfigurációs formátum némileg eltér az `/etc/network/interfaces`-ben használttól.

Az interfész tényleges konfigurációja egy másik sorba kerül, amely az `iface` szóval kezdődik, majd az interfész neve, az interfész által használt címcsalád neve és az interfész konfigurálásához használt módszer neve következik. A következő példa a `lo` (loopback) és az `enp3s5` interfészek alapvető konfigurációs fájlját mutatja be:

```
auto lo
iface lo inet loopback

auto enp3s5
iface enp3s5 inet dhcp
```

A címcsaládnak a TCP/IP hálózathoz `inet`-nek kell lennie, de támogatott az IPX hálózat (`ipx`) és az IPv6 hálózat (`inet6`) is. A loopback interfészek a `loopback` konfigurációs metódust használják. A `dhcp` módszerrel az interfész a hálózat DHCP-szervere által biztosított IP-beállításokat használja. A példakonfiguráció beállításai lehetővé teszik az `ifup` parancs végrehajtását az `enp3s5` interfész nevet argumentumként használva:

```
# ifup enp3s5
Internet Systems Consortium DHCP Client 4.4.1
Copyright 2004-2018 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/enp3s5/00:16:3e:8d:2b:5b
Sending on   LPF/enp3s5/00:16:3e:8d:2b:5b
Sending on   Socket/fallback
DHCPDISCOVER on enp3s5 to 255.255.255.255 port 67 interval 4
DHCPOFFER of 10.90.170.158 from 10.90.170.1
DHCPREQUEST for 10.90.170.158 on enp3s5 to 255.255.255.255 port 67
DHCPACK of 10.90.170.158 from 10.90.170.1
bound to 10.90.170.158 -- renewal in 1616 seconds.
```

Ebben a példában az `enp3s5` interfészhez a `dhcp` módszert választottuk, így az `ifup` parancs egy DHCP kliensprogramot hívott meg az IP-beállítások DHCP-szervertől való megszerzéséhez. Hasonlóképpen, az `ifdown enp3s5` parancs az interfész kikapcsolására is használható.

DHCP-szerverrel nem rendelkező hálózatokban ehelyett a `statikus` módszert lehet használni, és az IP-beállításokat kézzel kell megadni az `/etc/network/interfaces` állományban. Például:

```
iface enp3s5 inet static
    address 192.168.1.2/24
    gateway 192.168.1.1
```

A `statikus` módszert használó interfészeknek nincs szükségük megfelelő `auto` direktívára, mivel mindig akkor kerülnek elő, amikor a hálózati hardvert észlelésre kerül.

Ha ugyanannak az interfésznek egynél több `iface` bejegyzése van, akkor az összes konfigurált cím és beállítás érvényesül az adott interfész megjelenítésekor. Ez hasznos lehet, ha ugyanazon az interfészen kell konfigurálni az IPv4 és IPv6 címeket, valamint ha egyetlen interfészen kell több azonos típusú címet konfigurálni.

Lokális és távoli nevek

A működő TCP/IP beállítás csak az első lépés a teljes hálózati használhatóság felé. Amellett, hogy a rendszer képes a hálózat csomópontjait IP-címük alapján azonosítani, az ember számára könnyebben érthető nevekkel is azonosítani kell őket.

A név, amivel a rendszer azonosítja magát, testreszabható, és jó gyakorlat, ha ezt megtesszük, még akkor is, ha a gép nem csatlakozik semmilyen hálózathoz. A helyi név gyakran megegyezik a gép hálózati nevével, de ez nem feltétlenül igaz. Ha az `/etc/hostname` fájl létezik, az operációs rendszer az első sor tartalmát használja helyi névként, ezt követően egyszerűen *hostname*-ként (hostnévként). Az `/etc/hostname` mappában lévő `#` karakterrel kezdődő sorok nem kerülnek figyelembe vételre.

Az `/etc/hostname` fájl közvetlenül is szerkeszthető, de a gép hostneve a `hostnamectl` paranccsal is megadható. Ha a `set-hostname` alparanccsal együtt adjuk meg, a `hostnamectl` parancs a megadott nevet veszi argumentumként, és beírja az `/etc/hostname` állományba:

```
# hostnamectl set-hostname storage
# cat /etc/hostname
storage
```

Az `/etc/hostname`-ben definiált hostnév a *statikus* hostnév, azaz az a név, amely a rendszer hostnevének inicializálásához használatos a rendszer indításakor. A statikus hostnév egy 64 karakter hosszúságú, szabadon választott string lehet. Ajánlott azonban, hogy csak ASCII kisbetűs karakterekből álljon, szóközök és pontok nélkül. A DNS-domainnevek címkéinek megengedett formátumára kell korlátozódnia, még ha ez nem is szigorú követelmény.

A `hostnamectl` parancs használatával a statikus hostnév mellett két másik típusú hostnevet is beállíthatunk:

Pretty hostname

A statikus hostnévvel ellentétben a pretty hostnév mindenféle speciális karaktert tartalmazhat. Használható a gép leíróbb nevének beállítására, pl. “LAN Megosztott Tarhely”:

```
# hostnamectl --pretty set-hostname "LAN Megosztott Tarhely"
```

Transient hostname

Akkor használatos, ha a statikus hostnév nincs beállítva, vagy ha ez az alapértelmezett `localhost` név. Az átmeneti hostnév általában az egyéb automatikus konfigurációkkal együtt beállított név, de a `hostnamectl` paranccsal is módosítható, pl.

```
# hostnamectl --transient set-hostname generic-host
```

Ha sem a `--pretty`, sem a `--transient` kapcsolót nem használjuk, akkor mindhárom hostnévtípus a megadott névre lesz beállítva. Ha a statikus hostnevet akarjuk beállítani, de a pretty és transient neveket nem, akkor a `--static` kapcsolót kell használnunk. Minden esetben csak a statikus hostnév kerül tárolásra az `/etc/hostname` fájlban. A `hostnamectl` paranccsal különböző leíró és azonosító információk is megjeleníthetők a futó rendszerről:

```
$ hostnamectl status
     Static hostname: storage
     Pretty hostname: LAN Shared Storage
  Transient hostname: generic-host
           Icon name: computer-server
             Chassis: server
          Machine ID: d91962a957f749bbaf16da3c9c86e093
             Boot ID: 8c11dcab9c3d4f5aa53f4f4e8fdc6318
    Operating System: Debian GNU/Linux 10 (buster)
              Kernel: Linux 4.19.0-8-amd64
        Architecture: x86-64
```

Ez a `hostnamectl` parancs alapértelmezett művelete, így a `status` alparancs elhagyható.

Ami a távoli hálózati csomópontok nevét illeti, az operációs rendszer két alapvető módot alkalmazhat a nevek és az IP-címek megfeleltetésére: helyi forrást, vagy egy távoli szervert használ a nevek IP-re való lefordítására és fordítva. A módszerek kiegészíthetik egymást és prioritási sorrendjüket a *Name Service Switch* konfigurációs fájlban határozzuk meg: `/etc/nsswitch.conf`. Ezt a fájlt a rendszer és az alkalmazások arra használják, hogy ne csak a név-IP megfeleltetések forrásait határozzák meg, hanem azt is, hogy mely forrásokból szerezzenek be névszolgáltatási információkat egy sor kategóriában, az úgynevezett *adatbázisokban* (database).

A *hosts* adatbázis nyomon követi az hostnevek és a hostszámok közötti megfeleltetést. Az `/etc/nsswitch.conf` állományban a `hosts` kezdetű sor határozza meg a hozzárendelések biztosításáért felelős szolgáltatásokat:

```
hosts: files dns
```

Ebben a példában a `files` és a `dns` a szolgáltatásnevek, amelyek meghatározzák, hogyan fog működni a hostnevek keresési folyamata. A rendszer először a helyi fájlokban keres egyezéseket, majd a DNS-szolgáltatástól kér egyezést.

A hosts adatbázis helyi fájlja az `/etc/hosts`, egy egyszerű szöveges fájl, amely IP-címeket társít hostnevekhez, IP-címenként egy sor formában, pl.:

```
127.0.0.1 localhost
```

A *127.0.0.0.1* IP-cím a loopback interfész alapértelmezett címe, ezért kapcsolódik a *localhost* névhez.

Lehetőség van arra is, hogy opcionális aliasokat kössünk ugyanahhoz az IP-címhez. Az aliasok alternatív írásmódokat, rövidebb hostneveket biztosíthatnak, és a sor végén kell hozzáadni őket, például:

```
192.168.1.10 foo.mydomain.org foo
```

Az `/etc/hosts` fájl formázási szabályai a következők:

- A bejegyzés mezőit tetszőleges számú üres és/vagy tabulátor karakter választja el egymástól.
- A # karaktertől a sor végéig terjedő szöveg megjegyzésnek minősül, és figyelmen kívül hagyjuk.

- A hostnevek csak alfanumerikus karaktereket, mínuszjeleket és pontokat tartalmazhatnak.
- A hostneveknek alfanumerikus karakterrel kell kezdődniük és alfanumerikus karakterrel kell végződniük.

Az IPv6-címek az `/etc/hosts` állományba is felvehetők. A következő bejegyzés az IPv6 loopback címre vonatkozik:

```
::1 localhost ip6-localhost ip6-loopback
```

A `files` szolgáltatási specifikációt követően a `dns` specifikáció azt mondja a rendszernek, hogy kérjen egy DNS-szolgáltatást a kívánt név/IP társításhoz. Az ezért a módszerért felelős rutinokat *resolver*-nek (feloldó) nevezik, és a konfigurációs fájl az `/etc/resolv.conf`. Az alábbi példa egy általános `/etc/resolv.conf` állományt mutat be, amely a Google nyilvános DNS-szervereinek bejegyzéseit tartalmazza:

```
nameserver 8.8.4.4
nameserver 8.8.8.8
```

A példában látható módon a `nameserver` kulcsszó a DNS-szerver IP-címét jelöli. Csak egy névszerver szükséges, de legfeljebb három névszerver is megadható. A továbbiak tartalékként lesznek használva. Ha nincs névszerver bejegyzés, akkor alapértelmezés szerint a helyi gépen lévő névszerver lesz használatban.

A resolver beállítható úgy, hogy automatikusan hozzáadja a domaint a nevekhez, mielőtt a névszerveren lekérdezné azokat. Például:

```
nameserver 8.8.4.4
nameserver 8.8.8.8
domain mydomain.org
search mydomain.net mydomain.com
```

A `domain` bejegyzés a `mydomain.org`-ot állítja be helyi domainnévként, így az ezen a domainen belüli nevekre vonatkozó lekérdezések a helyi domainhez viszonyított rövid neveket használhatják. A `search` bejegyzésnek hasonló a célja, de itt a rövid név megadása esetén megpróbálandó domainek listáját fogadja el. Alapértelmezés szerint csak a helyi domainnevet tartalmazza.

Gyakorló feladatok

1. Milyen parancsokkal lehet listázni a rendszerben lévő hálózati adaptereket?

2. Milyen típusú az a hálózati adapter, amelynek az interfész neve `wlo1`?

3. Milyen szerepet játszik az `/etc/network/interfaces` fájl a rendszerindítás során?

4. Az `/etc/network/interfaces`-ben melyik bejegyzés konfigurálja az `eno1` interfészt, hogy IP-beállításait DHCP-vel kapja meg?

Gondolkodtató feladatok

1. Hogyan lehetne a `hostnamectl` parancsot arra használni, hogy csak a helyi gép *statikus* hostnevét változtassuk meg `firewall`-ra?

2. Milyen adatokat lehet módosítani a hostneveken kívül a `hostnamectl` paranccsal?

3. Az `/etc/hosts`-ban melyik bejegyzés társítja a `firewall` és a `router` nevet a `10.8.0.1` IP-hez?

4. Hogyan lehetne módosítani az `/etc/resolv.conf` fájlt, hogy minden DNS-kérést az `1.1.1.1` címre küldjön?

Összefoglalás

Ez a lecke a helyi hálózati konfiguráció tartós módosítását mutatja be a szabványos Linux fájlok és parancsok segítségével. A Linux elvárja, hogy a TCP/IP-beállítások meghatározott helyeken legyenek, és szükség lehet ezek megváltoztatására, ha az alapértelmezett beállítások nem megfelelőek. A lecke a következő témákat járja körül:

- Hogyan azonosítja a Linux a hálózati interfészeket.
- Az interfész aktiválása a rendszerindítás során és az alapvető IP-konfiguráció.
- Hogyan társítja az operációs rendszer a neveket a hostokhoz.

A bemutatott koncepciók, parancsok és procedúrák az alábbiak voltak:

- Interfész elnevezési konvenciók.
- Hálózati interfészek listázása az `ip` és `nmcli` segítségével.
- Interfészek aktiválása az `ifup` és `ifdown` segítségével.
- A `hostnamectl` parancs és az `/etc/hostname` fájl.
- Az `/etc/nsswitch.conf`, `/etc/hosts` és `/etc/resolv.conf` fájlok.

Válaszok a gyakorló feladatokra

1. Milyen parancsokkal lehet listázni a rendszerben lévő hálózati adaptereket?

   Az `ip link show`, `nmcli device` és a legacy `ifconfig` parancsokkal.

2. Milyen típusú az a hálózati adapter, amelynek az interfész neve `wlo1`?

   A név `wl`-el kezdődik, tehát egy vezeték nélküli (wireless) LAN adapter.

3. Milyen szerepet játszik az `/etc/network/interfaces` fájl a rendszerindítás során?

   Az `ifup` parancs által használt konfigurációkat tartalmazza a megfelelő interfészek aktiválásához a rendszerindítás során.

4. Az `/etc/network/interfaces`-ben melyik bejegyzés konfigurálja az `eno1` interfészt, hogy IP-beállításait DHCP-vel kapja meg?

   Az `iface eno1 inet dhcp` sor.

Válaszok a gondolkodtató feladatokra

1. Hogyan lehetne a `hostnamectl` parancsot arra használni, hogy csak a helyi gép *statikus* hostnevét változtassuk meg `firewall`-ra?

   A `--static` kapcsolóval: `hostnamectl --static set-hostname firewall`.

2. Milyen adatokat lehet módosítani a hostneveken kívül a `hostnamectl` paranccsal?

   A `hostnamectl` beállíthatja a helyi gép alapértelmezett ikonját, az eszköz (chassis) típusát, a helyet és a telepítési környezetet is.

3. Az `/etc/hosts`-ban melyik bejegyzés társítja a `firewall` és a `router` nevet a `10.8.0.1` IP-hez?

   A `10.8.0.1 firewall router` sor.

4. Hogyan lehetne módosítani az `/etc/resolv.conf` fájlt, hogy minden DNS-kérést az `1.1.1.1` címre küldjön?

   A `nameserver 1.1.1.1` használatával a névszerver bejegyzésnél.

109.2 Lecke 2

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	109 Hálózati alapok
Fejezet:	109.2 Perzisztens hálózatkonfiguráció
Lecke:	2/2

Bevezetés

A Linux gyakorlatilag minden olyan hálózati technológiát támogat, amelyet a szerverek, konténerek, virtuális gépek, asztali számítógépek és mobileszközök összekapcsolására használnak. Az ilyen hálózati csomópontok közötti kapcsolatok dinamikusak és heterogének lehetnek, így megfelelő kezelést igényelnek a bennük futó operációs rendszer részéről.

A múltban a forgalmazók saját, egyedi megoldásokat fejlesztettek ki a dinamikus hálózati infrastruktúra kezelésére. Ma már az olyan eszközök, mint a *NetworkManager* és a *systemd* olyan, átfogóbb és integráltabb funkciókat biztosítanak, amelyek minden egyedi igényt kielégítenek.

NetworkManager

A legtöbb Linux-disztribúció a *NetworkManager* szolgáltatás daemonját használja a rendszer hálózati kapcsolatainak konfigurálására és vezérlésére. A NetworkManager célja, hogy a hálózati konfigurációt a lehető legegyszerűbbé és automatikusabbá tegye. DHCP használata esetén például a NetworkManager gondoskodik az útvonalváltozásokról, az IP-címek lekéréséről és szükség esetén a DNS-szerverek helyi listájának frissítéséről. Ha vezetékes és vezeték nélküli kapcsolat is rendelkezésre áll, a NetworkManager alapértelmezés szerint a vezetékes kapcsolatot részesíti

előnyben. A NetworkManager mindig megpróbál legalább egy kapcsolatot aktívan tartani, amikor csak lehetséges.

NOTE

A DHCP (*Dynamic Host Configuration Protocol*) segítségével a kérés általában a hálózati adapteren keresztül érkezik, amint létrejön a kapcsolat a hálózattal. A hálózaton aktív DHCP-szerver ezután válaszol a beállításokkal (IP-cím, hálózati maszk, alapértelmezett útvonal stb.), amelyeket a kérvényezőnek az IP-protokollon keresztül történő kommunikációhoz használnia kell.

Alapértelmezés szerint a NetworkManager daemon vezérli az `/etc/network/interfaces` fájlban nem említett hálózati interfészeket. Ezt azért teszi, hogy ne zavarja az esetlegesen szintén jelenlévő egyéb konfigurációs módszereket, így csak a felügyelet nélküli interfészeket módosítja.

A NetworkManager szolgáltatás a háttérben fut root jogosultságokkal, és elindítja a rendszer online tartásához szükséges műveleteket. A közönséges felhasználók hálózati kapcsolatokat hozhatnak létre és módosíthatnak olyan kliensalkalmazásokkal, amelyek — bár maguk nem rendelkeznek root jogosultságokkal --, képesek kommunikálni a mögöttes szolgáltatással a kért műveletek végrehajtása érdekében.

A NetworkManager kliensalkalmazások mind parancssori, mind grafikus környezetben elérhetők. Az utóbbi esetben a kliensalkalmazás az asztali környezet tartozékaként jelenik meg (olyan nevek alatt, mint *nm-tray*, *network-manager-gnome*, *nm-applet* vagy *plasma-nm*), és általában az asztali sáv sarkában lévő jelzőikonon keresztül vagy a rendszerkonfigurációs segédprogramból érhető el.

A parancssorban maga a NetworkManager két kliensprogramot biztosít: `nmcli` és `nmtui`. Mindkét program ugyanazokkal az alapfunkciókkal van ellátva, de az `nmtui` egy curses-alapú interfésszel rendelkezik, míg az `nmcli` egy átfogóbb parancs, amely scriptekben is használható. Az `nmcli` a NetworkManager által ellenőrzött összes hálózathoz kapcsolódó tulajdonságot *objects* nevű kategóriákba sorolja:

`general`
: A NetworkManager általános állapota és műveletei.

`networking`
: Általános hálózati vezérlés.

`radio`
: A NetworkManager rádiós kapcsolói.

`connection`
: A NetworkManager kapcsolatai.

`device`

A NetworkManager által kezelt eszközök.

`agent`

A NetworkManager secret agent vagy polkit agent.

`monitor`

A NetworkManager változások figyelése.

Az objektum neve az `nmcli` parancs fő argumentuma. Ha például a rendszer általános kapcsolati állapotát szeretnénk megjeleníteni, akkor a `general` objektumot kell megadni argumentumként:

```
$ nmcli general
STATE      CONNECTIVITY  WIFI-HW  WIFI     WWAN-HW  WWAN
connected  full          enabled  enabled  enabled  enabled
```

A `STATE` oszlop azt mutatja meg, hogy a rendszer csatlakozik-e a hálózathoz vagy sem. Ha a kapcsolat külső hibás konfiguráció vagy hozzáférési korlátozások miatt korlátozott, akkor a `CONNECTIVITY` oszlop nem a `teljes` (full) kapcsolódási állapotot jelzi. Ha a `CONNECTIVITY` oszlopban megjelenik a `Portal`, az azt jelenti, hogy a csatlakozási folyamat befejezéséhez további hitelesítési lépésekre van szükség (általában a webböngészőn keresztül). A fennmaradó oszlopok a vezeték nélküli kapcsolatok állapotát jelzik (ha van ilyen), akár `WIFI`, akár `WWAN` (Wide Wireless Area Network, azaz mobilhálózatok). A `HW` utótag azt jelzi, hogy az állapot a hálózati eszköznek felel meg, nem pedig a rendszer hálózati kapcsolatának, vagyis azt, hogy a hardver engedélyezve vagy letiltva van-e az energiatakarékosság érdekében.

Az objektum argumentumon kívül az `nmcli`-nek egy parancs argumentumra is szüksége van a végrehajtáshoz. A `status` parancsot használja alapértelmezés szerint, ha nincs parancsargumentum, így az `nmcli general` parancsot valójában `nmcli general status`-ként értelmezi.

Ha a hálózati adapter közvetlenül a hozzáférési ponthoz van csatlakoztatva kábelen keresztül, akkor aligha szükséges bármilyen intézkedést tenni, de a vezeték nélküli hálózatok további interakciót igényelnek az új tagok felvételéhez. Az `nmcli` megkönnyíti a csatlakozási folyamatot, és elmenti a beállításokat a jövőbeni automatikus csatlakozás érdekében, ezért nagyon hasznos laptopok vagy más mobileszközök esetében.

A wi-fi-hez való csatlakozás előtt, célszerű először listázni a helyileg elérhető hálózatokat. Ha a rendszer rendelkezik működő wi-fi adapterrel, akkor az `device` objektum azt használja a rendelkezésre álló hálózatok keresésére az `nmcli device wifi list` parancs esetén:


```
$ nmcli device wifi list
IN-USE  BSSID              SSID        MODE   CHAN  RATE        SIGNAL  BARS  SECURITY
        90:F6:52:C5:FA:12  Hypnotoad   Infra  11    130 Mbit/s  67      ▂▄▆_  WPA2
        10:72:23:C7:27:AC  Jumbao      Infra  1     130 Mbit/s  55      ▂▄__  WPA2
        00:1F:33:33:E9:BE  NETGEAR     Infra  1     54 Mbit/s   35      ▂▄__  WPA1 WPA2
        A4:33:D7:85:6D:B0  AP53        Infra  11    130 Mbit/s  32      ▂▄__  WPA1 WPA2
        98:1E:19:1D:CC:3A  Bruma       Infra  1     195 Mbit/s  22      ▂___  WPA1 WPA2
```

A legtöbb felhasználó valószínűleg az `SSID` oszlopban szereplő nevet fogja használni a kívánt hálózat azonosítására. Például az `nmcli` parancs ismét a `device` objektummal tud csatlakozni a `Hypnotoad` nevű hálózathoz:

```
$ nmcli device wifi connect Hypnotoad
```

Ha a parancsot grafikus környezetben, terminál emulátoron belül hajtjuk végre, akkor egy párbeszédpanel jelenik meg, amely a hálózat passphrase-t (egy jelszó jellegű hosszabb szöveg) kéri. Ha a parancsot a szöveges konzolban hajtjuk végre, a jelszót a többi argumentummal együtt adhatjuk meg:

```
$ nmcli device wifi connect Hypnotoad password MyPassword
```

Ha a wi-fi hálózat elrejti az SSID nevét, az `nmcli` a `hidden yes` argumentummal még mindig tud csatlakozni hozzá:

```
$ nmcli device wifi connect Hypnotoad password MyPassword hidden yes
```

Ha a rendszerben egynél több wi-fi adapter van, akkor a használandó adaptert az `ifname` kapcsolóval lehet megjelölni. Példa a `wlo1` nevű adapterrel való csatlakozára:

```
$ nmcli device wifi connect Hypnotoad password MyPassword ifname wlo1
```

Miután a kapcsolat létrejött, a NetworkManager a megfelelő SSID után nevezi el (ha wi-fi-ről van szó), és megtartja a jövőbeli csatlakozásokhoz. A kapcsolatok nevei és UUID-ik az `nmcli connection show` paranccsal listázhatók:

```
$ nmcli connection show
NAME                UUID                                  TYPE      DEVICE
Ethernet            53440255-567e-300d-9922-b28f0786f56e  ethernet  enp3s5
```

```
tun0                cae685e1-b0c4-405a-8ece-6d424e1fb5f8  tun       tun0
Hypnotoad           6fdec048-bcc5-490a-832b-da83d8cb7915  wifi      wlo1
4G                  a2cf4460-0cb7-42e3-8df3-ccb927f2fd88  gsm       --
```

Az egyes kapcsolatok típusa — ami lehet `ethernet`, `wifi`, `tun`, `gsm`, `bridge`, stb. — valamint az eszköz is megjelenítésre kerül, amelyhez kapcsolódnak. Egy adott kapcsolaton történő művelet végrehajtásához meg kell adni annak nevét vagy UUID-jét. Például a `Hypnotoad` kapcsolat megszüntetéséhez:

```
$ nmcli connection down Hypnotoad
Connection 'Hypnotoad' successfully deactivated
```

Hasonlóképpen, az `nmcli connection up Hypnotoad` parancs használható a kapcsolat felállítására, mivel a NetworkManager már elmentette azt. Az interfész neve is használható az újrakapcsolódáshoz, de ebben az esetben a `device` objektumot kell használni helyette:

```
$ nmcli device disconnect wlo2
Device 'wlo1' successfully disconnected.
```

Az interfész neve a kapcsolat újbóli létrehozásához is használható:

```
$ nmcli device connect wlo2
Device 'wlo1' successfully activated with '833692de-377e-4f91-a3dc-d9a2b1fcf6cb'.
```

Vegyük figyelembe, hogy a kapcsolat UUID-je minden egyes alkalommal változik, amikor a kapcsolat létrejön, ezért a következetesség érdekében célszerű a nevét használni.

Ha a vezeték nélküli adapter rendelkezésre áll, de nem használjuk, akkor az energiatakarékosság érdekében kikapcsolható. Ezúttal a *radio* objektumot kell átadni az `nmcli`-nek:

```
$ nmcli radio wifi off
```

Természetesen a vezeték nélküli eszköz újra bekapcsolható az `nmcli radio wifi on` paranccsal.

A kapcsolatok létrehozását követően a jövőben nem lesz szükség kézi beavatkozásra, mivel a NetworkManager azonosítja a rendelkezésre álló ismert hálózatokat, és automatikusan csatlakozik hozzájuk. Szükség esetén a NetworkManager rendelkezik olyan bővítményekkel, amelyekkel bővíthetők a funkciói, ilyen például a VPN-kapcsolatok támogatása.

systemd-networkd

A systemd-t futtató rendszerek opcionálisan használhatják a beépített daemonjait a hálózati kapcsolat kezelésére: a `systemd-networkd`-t a hálózati interfészek vezérlésére, a `systemd-resolved`-t pedig a helyi névfeloldás kezelésére. Ezek a szolgáltatások visszafelé kompatibilisek a régebbi Linux konfigurációs módszerekkel, de különösen a hálózati interfészek konfigurálása rendelkezik olyan tulajdonságokkal, amelyeket érdemes ismerni.

A systemd-networkd által a hálózati interfészek beállításához használt konfigurációs fájlok a következő három mappa valamelyikében találhatók:

`/lib/systemd/network`
: A rendszer hálózati mappája.

`/run/systemd/network`
: A volatilis futásidejű hálózati mappa.

`/etc/systemd/network`
: A helyi adminisztrációs hálózati mappa.

A fájlok feldolgozása lexikográfiai sorrendben történik, ezért ajánlott a nevüket számokkal kezdeni, hogy a sorrend könnyebben olvasható és beállítható legyen.

Az `/etc`-ben található fájlok rendelkeznek a legmagasabb prioritással, míg a `/run`-ban lévők elsőbbséget élveznek az azonos nevű `/lib`-ben lévő állományokkal szemben. Ez azt jelenti, hogy ha a különböző mappákban lévő konfigurációs fájlok azonos nevűek, akkor a systemd-networkd figyelmen kívül hagyja a kisebb prioritású fájlokat. A fájlok ilyen módon történő szétválasztása egy módja annak, hogy az interfész beállításait az eredeti fájlok módosítása nélkül megváltoztassuk: a `/etc/systemd/network`-ben elhelyezett módosítások felülírhatják a `/lib/systemd/network`-ben lévő beállításokat.

Az egyes konfigurációs fájlok célja az utótagtól (suffix) függ. A `.netdev` végződésű fájlneveket a systemd-networkd virtuális hálózati eszközök, például *bridge* vagy *tun* eszközök létrehozására használja. A systemd-networkd automatikusan felismeri és konfigurálja a hálózati eszközöket, amint azok megjelennek — valamint figyelmen kívül hagyja a más módon már konfigurált eszközöket --, így a legtöbb esetben nincs szükség ezeknek a fájloknak a hozzáadására.

A legfontosabb utótag a `.network`. Az ezzel az utótaggal ellátott fájlok hálózati címek és útvonalak beállítására használhatók. A többi konfigurációs fájltípushoz hasonlóan a fájl neve határozza meg a feldolgozás sorrendjét. A hálózati interfész, amelyre a konfigurációs fájl hivatkozik, az állományon belül a `[Match]` szakaszban kerül meghatározásra.

Például az `/etc/systemd/network/30-lan.network` fájlban az `[Match]` szakaszban a `Name=enp3s5` bejegyzéssel lehet kiválasztani az `enp3s5` ethernet hálózati interfészt:

```
[Match]
Name=enp3s5
```

A szóközzel elválasztott nevek listája is elfogadható, hogy egyszerre több hálózati interfésznek is megfeleljen ugyanazzal a fájllal. A nevek tartalmazhatnak shell-stílusú globokat, mint például `en*`. Más bejegyzések különböző illesztési szabályokat adnak meg, például egy hálózati eszköz kiválasztása a MAC-címe alapján így néz ki:

```
[Match]
MACAddress=00:16:3e:8d:2b:5b
```

Az eszköz beállításai a fájl `[Network]` szakaszában találhatók. Egy egyszerű statikus hálózati konfigurációhoz csak az `Address` és a `Gateway` bejegyzésekre van szükség:

```
[Match]
MACAddress=00:16:3e:8d:2b:5b

[Network]
Address=192.168.0.100/24
Gateway=192.168.0.1
```

Ha statikus IP-címek helyett a DHCP protokollt szeretnénk használni, akkor a `DHCP` bejegyzést kell használni:

```
[Match]
MACAddress=00:16:3e:8d:2b:5b

[Network]
DHCP=yes
```

A systemd-networkd szolgáltatás megpróbálja lekérni a hálózati interfész IPv4 és IPv6 címeit is. Csak az IPv4 esetén a `DHCP=ipv4` értéket kell használni. Hasonlóképpen, a `DHCP=ipv6` figyelmen kívül hagyja az IPv4 beállításokat, és csak a megadott IPv6 címet használja.

A jelszóval védett vezeték nélküli hálózatokat is konfigurálhatja a systemd-networkd, de a hálózati adaptert már hitelesíteni kell a hálózatban, mielőtt a systemd-networkd konfigurálni tudná. A

hitelesítést a *WPA supplicant* végzi, amely egy jelszóval védett hálózatokhoz való hálózati adapterek konfigurálására szolgáló program.

Az első lépés a hitelesítő fájl létrehozása a `wpa_passphrase` paranccsal:

```
# wpa_passphrase MyWifi > /etc/wpa_supplicant/wpa_supplicant-wlo1.conf
```

Ez a parancs átveszi a `MyWifi` vezeték nélküli hálózat passphrase-jét a szabványos bemenetről, és a hash-jét az `/etc/wpa_supplicant/wpa_supplicant-wlo1.conf`-ban tárolja. Vegyük figyelembe, hogy a fájlnévnek tartalmaznia kell a vezeték nélküli interfész megfelelő nevét, ezért szerepel benne a `wlo1`.

A systemd menedzser beolvassa a WPA passphrase-ket tartalmazó fájlokat az `/etc/wpa_supplicant/` állományban, és létrehozza a megfelelő szolgáltatást a WPA supplicant futtatásához és az interfész üzembe helyezéséhez. A példában létrehozott passphrase-es fájlhoz tartozik egy megfelelő szolgáltatásegység, a `wpa_supplicant@wlo1.service`. A `systemctl start wpa_supplicant@wlo1.service` parancs a vezeték nélküli adaptert társítja a távoli hozzáférési ponthoz. A `systemctl enable wpa_supplicant@wlo1.service` paranccsal a társítás automatikus lesz a rendszerindításkor.

Végül, az `/etc/systemd/network/`-ben meg kell lennie a `wlo1` interfészhez tartozó `.network` fájlnak, mivel a systemd-networkd ezt fogja használni az interfész konfigurálásához, amint a WPA supplicant befejezi a hozzáférési ponthoz való társítást.

Gyakorló feladatok

1. Mit jelent a `Portal` szó a `CONNECTIVITY` oszlopban az `nmcli general status` parancs kimenetében?

2. Hogyan tud egy közönséges felhasználó egy konzolterminálban az `nmcli` paranccsal csatlakozni a `MyWifi` vezeték nélküli hálózathoz, amelyet a `MyPassword` jelszó véd?

3. Milyen paranccsal kapcsolhatjuk be a vezeték nélküli adaptert, ha azt az operációs rendszer korábban letiltotta?

4. Az egyéni konfigurációs fájlokat melyik mappában kell elhelyezni, amikor a systemd-networkd kezeli a hálózati interfészeket?

Gondolkodtató feladatok

1. Hogyan kell futtatnia a felhasználónak az `nmcli` parancsot, ha törölni szeretné a `Hotel Internet` nevű, nem használt kapcsolatot?

2. A NetworkManager rendszeresen vizsgálja a wi-fi hálózatokat, és az `nmcli device wifi list` parancs csak a legutóbbi vizsgálat során talált hozzáférési pontokat listázza. Hogyan kell használni az `nmcli` parancsot, hogy a NetworkManager azonnal új szkennelést hajtson végre az összes elérhető hozzáférési pont megtalálásának érdekében?

3. Milyen `name` bejegyzést kell használni a systemd-networkd konfigurációs fájl `[Match]` szakaszában, hogy az az összes ethernet interfésznek megfeleljen?

4. Hogyan kell végrehajtani a `wpa_passphrase` parancsot, hogy az argumentumként megadott passphrase-t használja, és ne a standard bemenetet?

Összefoglalás

Ez a lecke a Linuxban a heterogén és dinamikus hálózati kapcsolatok kezelésére használt általános eszközöket tárgyalja. Bár a legtöbb konfigurációs módszer nem igényel felhasználói beavatkozást, néha mégis szükség van rá, és az olyan eszközök, mint a *NetworkManager* és a *systemd-networkd* minimálisra csökkenthetik a gondokat. A lecke a következő témákat járja körül:

- Hogyan integrálódik a NetworkManager és a systemd-networkd a rendszerbe.
- Hogyan léphet kapcsolatba a felhasználó a NetworkManagerrel és a systemd-networkd-vel.
- A NetworkManager és a systemd-networkd alapvető interfész-konfigurációja.

A bemutatott koncepciók, parancsok és procedúrák az alábbiak voltak:

- NetworkManager kliensparancsok: `nmtui` és `nmcli`.
- Vezeték nélküli hálózatok keresése és csatlakoztatása az `nmcli` megfelelő parancsaival.
- Tartós wi-fi hálózati kapcsolatok a systemd-networkd használatával.

Válaszok a gyakorló feladatokra

1. Mit jelent a `Portal` szó a `CONNECTIVITY` oszlopban az `nmcli general status` parancs kimenetében?

   Ez azt jelenti, hogy a csatlakozási folyamat befejezéséhez további hitelesítési lépésekre van szükség (általában a webböngészőn keresztül).

2. IHogyan tud egy közönséges felhasználó egy konzolterminálban az `nmcli` paranccsal csatlakozni a `MyWifi` vezeték nélküli hálózathoz, amelyet a `MyPassword` jelszó véd?

   Egy szöveges terminálban a parancs:

   ```
   $ nmcli device wifi connect MyWifi password MyPassword
   ```

3. Milyen paranccsal kapcsolhatjuk be a vezeték nélküli adaptert, ha azt az operációs rendszer korábban letiltotta?

   ```
   $ nmcli radio wifi on
   ```

4. Az egyéni konfigurációs fájlokat melyik mappában kell elhelyezni, amikor a systemd-networkd kezeli a hálózati interfészeket?

   A helyi hálózati adminisztrációs mappában: `/etc/systemd/network`.

Válaszok a gondolkodtató feladatokra

1. Hogyan kell futtatnia a felhasználónak az `nmcli` parancsot, ha törölni szeretné a `Hotel Internet` nevű, nem használt kapcsolatot?

   ```
   $ nmcli connection delete "Hotel Internet"
   ```

2. A NetworkManager rendszeresen vizsgálja a wi-fi hálózatokat, és az `nmcli device wifi list` parancs csak a legutóbbi vizsgálat során talált hozzáférési pontokat listázza. Hogyan kell használni az `nmcli` parancsot, hogy a NetworkManager azonnal új szkennelést hajtson végre az összes elérhető hozzáférési pont megtalálásának érdekében?

   A root felhasználó tudja futtatni a `nmcli device wifi rescan` parancsot.

3. Milyen `name` bejegyzést kell használni a systemd-networkd konfigurációs fájl `[Match]` szakaszában, hogy az az összes ethernet interfésznek megfeleljen?

   A `name=en*` bejegyzést, mivel az `en` az ethernet interfészek előtagja a Linuxban, és a systemd-networkd elfogadja a shell-szerű globokat.

4. Hogyan kell végrehajtani a `wpa_passphrase` parancsot, hogy az argumentumként megadott passphrase-t használja, és ne a standard bemenetet?

   A jelszót közvetlenül az SSID után kell megadni, például `wpa_passphrase MyWifi MyPassword`.

109.3 Alapvető hálózati hibaelhárítás

Hivatkozás az LPI célkitűzésres

LPIC-1 version 5.0, Exam 102, Objective 109.3

Súlyozás

4

Kulcsfontosságú ismeretek

- Hálózati interfészek manuális konfigurálása, beleértve a hálózati interfészek konfigurációjának megtekintését és módosítását az iproute2 használatával
- Útválasztás manuális konfigurálása, beleértve a routing table-ök megtekintését és módosítását, valamint az alapértelmezett útvonal beállítását az iproute2 használatával
- A hálózati konfigurációval kapcsolatos problémák elhárítása
- A régebbi net-tools parancsok ismerete

A használt fájlok, kifejezések és segédprogramok listája

- `ip`
- `hostname`
- `ss`
- `ping`
- `ping6`
- `traceroute`
- `traceroute6`
- `tracepath`
- `tracepath6`

- `netcat`
- `ifconfig`
- `netstat`
- `route`

109.3 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	109 Hálózati alapok
Fejezet:	109.3 Alapvető hálózati hibaelhárítás
Lecke:	1/2

Bevezetés

A Linux nagyon rugalmas és hatékony hálózati képességekkel rendelkezik. Valójában a Linux-alapú operációs rendszereket gyakran használják az általános hálózati eszközökön, beleértve a drága kereskedelmi berendezéseket is. A Linux hálózatépítés önmagában is egy tanúsítvány lehetne. Ezt szem előtt tartva ez a lecke csak néhány alapvető konfigurációs és hibaelhárítási eszközzel foglalkozik.

Mielőtt belekezdenénk ebbe a leckébe, mindenképpen olvassuk el az internetprotokollokról és a tartós hálózati konfigurációról szóló leckéket. Ebben a leckében az IPv4 és IPv6 hálózatok konfigurálásához és hibaelhárításához szükséges eszközökkel foglalkozunk.

Bár nem hivatalos célúak, a *packet sniffers* (csomagelemzők), mint például a `tcpdump` hasznos hibaelhárító eszközök. Lehetővé teszik a hálózati interfészre érkező vagy onnan kimenő csomagok megtekintését és rögzítését. Az olyan eszközök segítségével, mint a *hex viewers* és a *protocol analyzers* ezeket a csomagokat részletesebben lehet megtekinteni, mint ahogyan azt a csomagelemzők általában lehetővé teszik. Nem árt, ha legalább nagyjából ismerjük az ilyen programokat.

Az `ip` parancs

Az `ip` parancs egy meglehetősen új segédprogram, amellyel szinte bármit megtekinthetünk és beállíthatunk, ami a hálózati konfigurációkkal kapcsolatos. Ez a lecke az 'ip' leggyakrabban használt alparancsaival foglalkozik, de ezzel is csak a felszínt kapargatjuk. Ha elolvassuk a dokumentációt, akkor sokkal hatékonyabban tudjuk használni.

Az `ip` minden egyes alparancsának megvan a maga man oldala. Az 'ip' man oldal `SEE ALSO` szakaszában ezek listája található meg:

```
$ man ip
...
SEE ALSO
       ip-address(8), ip-addrlabel(8), ip-l2tp(8), ip-link(8), ip-maddress(8),
       ip-monitor(8), ip-mroute(8), ip-neighbour(8), ip-netns(8), ip-
       ntable(8), ip-route(8), ip-rule(8), ip-tcp_metrics(8), ip-token(8), ip-
       tunnel(8), ip-xfrm(8)
       IP Command reference ip-cref.ps
...
```

Ahelyett, hogy minden alkalommal megnéznénk, amikor szükségünk van a man oldalra, egyszerűen csak adjuk hozzá a `-` és az alparancs nevét az `ip`-hez, pl. `man ip-route`!

Egy másik információforrás a súgó funkció. A beépített súgó megtekintéséhez írjuk be a `help` szót az alparancs után:

```
$ ip address help
Usage: ip address {add|change|replace} IFADDR dev IFNAME [ LIFETIME ]
                                                      [ CONFFLAG-LIST ]
       ip address del IFADDR dev IFNAME [mngtmpaddr]
       ip address {save|flush} [ dev IFNAME ] [ scope SCOPE-ID ]
                            [ to PREFIX ] [ FLAG-LIST ] [ label LABEL ] [up]
       ip address [ show [ dev IFNAME ] [ scope SCOPE-ID ] [ master DEVICE ]
                         [ type TYPE ] [ to PREFIX ] [ FLAG-LIST ]
                         [ label LABEL ] [up] [ vrf NAME ] ]
       ip address {showdump|restore}
IFADDR := PREFIX | ADDR peer PREFIX
...
```

A netmask és az útvonalválasztás felülvizsgálata

Az IPv4 és az IPv6 úgynevezett útválasztott (routed) vagy útválasztható (routable) protokollok. Ez azt jelenti, hogy úgy tervezték őket, hogy a hálózat tervezői számára lehetővé tegyék a forgalomáramlás szabályozását. Az Ethernet nem routolható protokoll. Ez azt jelenti, hogy ha sok eszközt csatlakoztatnánk egymáshoz csak az Ethernet segítségével, akkor nagyon keveset tehetnénk a hálózati forgalom irányításának érdekében. A forgalom irányítását célzó bármilyen intézkedés a jelenlegi routolható és útválasztó protokollokhoz hasonlóan végződne.

Az útválasztó protokollok lehetővé teszik a hálózattervezők számára a hálózatok szegmentálását a csatlakoztatható eszközök feldolgozási igényeinek csökkentése, a redundancia biztosítása és a forgalom kezelése érdekében.

Az IPv4 és IPv6 címek két részből állnak. Az első bitcsoport alkotja a hálózati részt, míg a második bitcsoport a host részt. A hálózati részt alkotó bitek számát a *netmask* (más néven *subnet mask*) határozza meg. Néha *prefix hossznak* (prefix length) is nevezik. Bárhogy is hívják, ez azon bitek száma, amelyet a gép a cím hálózati részeként kezel. Az IPv4 esetében ez néha pontokkal elválasztott decimálisokkal van megadva.

Az alábbiakban egy IPv4-es példát mutatunk be. Vegyük észre, hogy a bináris számjegyek megtartják helyértéküket az oktettekben, még akkor is, ha azt a netmaskokkal osztjuk fel!

```
192.168.130.5/20

          192      168      130      5
          11000000 10101000 10000010 00000101

20 bit = 11111111 11111111 11110000 00000000

Hálózat = 192.168.128.0
Host    = 2.5
```

A cím hálózati részét az IPv4 vagy IPv6 gépek arra használják, hogy a routing table-ben (útválasztási táblázatban) megnézzék, hogy egy csomagot melyik interfészen kell elküldeni. Amikor egy IPv4 vagy IPv6 routingot engedélyező host olyan csomagot kap, amely nem magának a hostnak szól, megpróbálja a célállomás hálózati részét a routing table-ben szereplő hálózathoz rendelni. Ha talál megfelelő bejegyzést, akkor a csomagot a routing table-ben megadott célállomásra küldi. Ha nem talál bejegyzést, és van alapértelmezett útvonal konfigurálva, akkor az alapértelmezett útvonalra küldi a csomagot. Ha nem talál bejegyzést, és nincs konfigurálva alapértelmezett útvonal, a csomagot elveti.

Interfész konfigurálása

Két eszközzel fogunk foglalkozni, amelyeket a hálózati interfész konfigurálásához használhatunk: `ifconfig` és `ip`. Az `ifconfig` program, bár még mindig széles körben használatos, már elavult eszköznek számít, és nem biztos, hogy elérhető az újabb rendszereken.

TIP Az újabb Linux disztribúciókban a `net-tools` csomag telepítése biztosítja a régi hálózati parancsokat.

Egy interfész konfigurálása előtt először tudnunk kell, hogy milyen interfészek állnak rendelkezésre. Ezt többféleképpen is megnézhetjük, az egyik lehetőség az `ifconfig -a` kapcsolójának használata:

```
$ ifconfig -a
```

Egy másik lehetőség az `ip`. Néha láthatunk példákat az `ip addr`, `ip a`, néhányat pedig az `ip address` használatával. Ezek szinonimák. Hivatalosan az alparancs az `ip address`. Ez azt jelenti, hogy ha meg akarjuk nézni a man oldalt, akkor a `man ip-address`-t kell használnunk, nem pedig a `man ip-addr`-t.

A `ip` parancs `link` alparancsa felsorolja a konfigurálható interfészkapcsolatokat:

```
$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT
group default qlen 1000
    link/ether 08:00:27:54:18:57 brd ff:ff:ff:ff:ff:ff
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT
group default qlen 1000
    link/ether 08:00:27:ab:11:3e brd ff:ff:ff:ff:ff:ff
```

Feltételezve, hogy a `sys` fájlrendszer mountolva van, a `/sys/class/net` fájlrendszer tartalmát is ki tudjuk listázni:

```
$ ls /sys/class/net
enp0s3  enp0s8  lo
```

Root felhasználóként kell bejelentkeznünk, ha szeretnénk egy interfészt az `ifconfig` paranccsal

konfigurálni, vagy a `sudo` segítségével, root jogosultsággal kell futtatnunk a parancsot. Nézzük az alábbi példát:

```
# ifconfig enp1s0 192.168.50.50/24
```

Az `ifconfig` Linux verziója rugalmasan kezeli az alhálózati maszk megadásának módját:

```
# ifconfig eth2 192.168.50.50 netmask 255.255.255.0
# ifconfig eth2 192.168.50.50 netmask 0xffffff00
# ifconfig enp0s8 add 2001:db8::10/64
```

Vegyük észre, hogy az IPv6 esetében az `add` kulcsszót használtuk! Ha az IPv6-cím előtt nem írjuk be az `add` szót, hibaüzenetet kapunk.

A következő parancs az `ip`-vel konfigurál egy interfészt:

```
# ip addr add 192.168.5.5/24 dev enp0s8
# ip addr add 2001:db8::10/64 dev enp0s8
```

Az `ip`-vel a parancs ugyanaz IPv4 és IPv6 esetén is.

Alacsonyszintű konfiguráció

Az `ip link` parancs alacsonyszintű interfész- vagy protokollbeállítások, például VLAN-ok, ARP vagy MTU-k konfigurálására, illetve egy interfész letiltására szolgál.

Az `ip link` gyakori feladata egy interfész letiltása vagy engedélyezése. Ez az `ifconfig` segítségével is elvégezhető:

```
# ip link set dev enp0s8 down
# ip link show dev enp0s8
3: enp0s8: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast state DOWN mode DEFAULT group
default qlen 1000
    link/ether 08:00:27:ab:11:3e brd ff:ff:ff:ff:ff:ff
# ifconfig enp0s8 up
# ip link show dev enp0s8
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT
group default qlen 1000
    link/ether 08:00:27:ab:11:3e brd ff:ff:ff:ff:ff:ff
```

Néha szükség lehet egy interfész MTU-jának beállítására. Az interfészek engedélyezéséhez/tiltásához hasonlóan ezt is az `ifconfig` vagy az `ip link` segítségével lehet elvégezni:

```
# ip link set enp0s8 mtu 2000
# ip link show dev enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 2000 qdisc pfifo_fast state UP mode DEFAULT
group default qlen 1000
    link/ether 08:00:27:54:53:59 brd ff:ff:ff:ff:ff:ff
# ifconfig enp0s3 mtu 1500
# ip link show dev enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT
group default qlen 1000
    link/ether 08:00:27:54:53:59 brd ff:ff:ff:ff:ff:ff
```

A routing table

A `route`, `netstat -r` és `ip route` parancsok mind használhatók a routing table megtekintésére. Ha módosítani szeretnénk az útvonalakat, akkor a `route` vagy az `ip route` parancsot kell használni. Az alábbiakban láthatunk példákat a routing table megtekintésére:

```
$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         10.0.2.2        0.0.0.0         UG        0 0          0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
192.168.150.0   0.0.0.0         255.255.255.0   U         0 0          0 enp0s8
$ ip route
default via 10.0.2.2 dev enp0s3 proto dhcp metric 100
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
192.168.150.0/24 dev enp0s8 proto kernel scope link src 192.168.150.200
$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.0.2.2        0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
192.168.150.0   0.0.0.0         255.255.255.0   U     0      0        0 enp0s8
```

Vegyük észre, hogy nincs IPv6-ra vonatkozó kimenet! Ha meg szeretnénk tekinteni az IPv6-os routing table-t, akkor a `route -6`, `netstat -6r` és `ip -6 route` parancsokat kell használnunk!

```
$ route -6
Kernel IPv6 routing table
Destination                    Next Hop                   Flag Met Ref Use If
2001:db8::/64                  [::]                       U    256 0      0 enp0s8
fe80::/64                      [::]                       U    100 0      0 enp0s3
2002:a00::/24                  [::]                       !n   1024 0      0 lo
[::]/0                         2001:db8::1                UG   1   0      0 enp0s8
localhost/128                  [::]                       Un   0   2     84 lo
2001:db8::10/128               [::]                       Un   0   1      0 lo
fe80::a00:27ff:fe54:5359/128   [::]                       Un   0   1      0 lo
ff00::/8                       [::]                       U    256 1      3 enp0s3
ff00::/8                       [::]                       U    256 1      6 enp0s8
```

A `netstat -r6` példáját kihagyjuk, mert a kimenete ugyanaz, mint a `route -6` esetén. A fenti `route` parancs néhány kimenete magától értetődő. A `Flag` oszlop az útvonalról ad némi információt. Az `U` flag azt jelzi, hogy az útvonal működik. A `!` azt jelenti, hogy az útvonal elutasítva, azaz a `!` jelzővel ellátott útvonalat nem fogjuk használni. Az `n` azt jelenti, hogy az útvonal nincs gyorsítótárazva. A kernel a gyorsabb keresés érdekében az összes ismert útvonaltól elkülönítve tárolja az útvonalak gyorsítótárát. A `G` flag egy gatewayt jelöl. A `Metric` vagy `Met` oszlopot a kernel nem használja. Ez a céltól való adminisztratív távolságra utal, amit az routing protokollok használnak a dinamikus útvonalak meghatározásához. A `Ref` oszlop a hivatkozások száma, vagyis az útvonal használatának a gyakorisága. A `Metric` oszlophoz hasonlóan ezt sem használja a Linux kernel. A `Use` oszlop az útvonal keresések (lookup) számát mutatja.

A `netstat -r` kimenetében az `MSS` az adott útvonalon keresztül futó TCP-kapcsolatok maximális szegmensméretét jelzi. A `Window` oszlop a defualt TCP ablakméretet mutatja. Az `irtt` a csomagok ezen az útvonalon történő oda-vissza utazási idejét mutatja.

Az `ip route` és az `ip -6 route` kimenete a következőképpen néz ki:

1. Úticél.
2. Választható cím, majd az interfész.
3. Az útvonal hozzáadásához használt routing protokoll.
4. Az útvonal hatóköre. Ha ez nincs megadva, akkor vagy globális hatókör, vagy gateway.
5. Az útvonal metrikája. A dinamikus routing protokollok ezt használják az útvonal költségének meghatározására. Ezt a legtöbb rendszer nem használja.
6. Ha IPv6 útvonalról van szó, az RFC4191 szerinti útvonalpreferencia.

Nézzünk meg néhány példát, hogy érthetőbb legyen:

IPv4 Példa

```
default via 10.0.2.2 dev enp0s3 proto dhcp metric 100
```

1. Az úticél a default route.
2. A gateway cím `10.0.2.2`, ami az `enp0s3` interfészen keresztül érhető el.
3. A routing table-be a DHCP által került be.
4. Scope nincs, így az globális.
5. Az út költsége `100`.
6. Nincs IPv6 útpreferencia.

IPv6 Példa

```
fc0::/64 dev enp0s8 proto kernel metric 256 pref medium
```

1. Az úticél `fc0::/64`.
2. Az `enp0s8` interfészen keresztül érhető el.
3. A kernel automatikusan adta hozzá.
4. Scope nincs, tehát globális.
5. Az út költsége `256`.
6. Az IPv6 preferencia `medium`.

Útvonalak menedzselése

Az útvonalakat a `route` vagy az `ip route` használatával lehet kezelni. Az alábbiakban látható egy példa egy útvonal hozzáadására és eltávolítására a `route` parancs használatával. A `route` parancsban az IPv6 esetén a `-6` kapcsolót kell használni:

```
# ping6 -c 2 2001:db8:1::20
connect: Network is unreachable
# route -6 add 2001:db8:1::/64 gw 2001:db8::3
# ping6 -c 2 2001:db8:1::20
PING 2001:db8:1::20(2001:db8:1::20) 56 data bytes
64 bytes from 2001:db8:1::20: icmp_seq=1 ttl=64 time=0.451 ms
64 bytes from 2001:db8:1::20: icmp_seq=2 ttl=64 time=0.438 ms
# route -6 del 2001:db8:1::/64 gw 2001:db8::3
# ping6 -c 2 2001:db8:1::20
```

```
connect: Network is unreachable
```

Lentebb látható ugyanaz a példa, az `ip route` parancs használatával:

```
# ping6 -c 2 2001:db8:1:20
connect: Network is unreachable
# ip route add 2001:db8:1::/64 via 2001:db8::3
# ping6 -c 2 2001:db8:1:20
PING 2001:db8:1::20(2001:db8:1::20) 56 data bytes
64 bytes from 2001:db8:1::20: icmp_seq=2 ttl=64 time=0.529 ms
64 bytes from 2001:db8:1::20: icmp_seq=2 ttl=64 time=0.438 ms
# ip route del 2001:db8:1::/64 via 2001:db8::3
# ping6 -c 2 2001:db8:1::20
connect: Network is unreachable
```

Gyakorló feladatok

1. Mely parancsokkal listázhatjuk a hálózati interfészeket?

2. Hogyan lehet ideiglenesen letiltani egy interfészt? Hogyan lehet újra engedélyezni?

3. Az alábbiak közül melyik fogadható el IPv4 alhálózati maszkként?

`0.0.0.255`	
`255.0.255.0`	
`255.252.0.0`	
`/24`	

4. Mely parancsokkal tudjuk ellenőrizni az alapértelmezett útvonalat?

5. Hogyan adhatunk hozzá egy második IP-címet egy interfészhez?

Gondolkodtató feladatok

1. Az `ip` melyik alparancsa használható vlan címkézés (tagging) konfigurálására?

2. Hogyan lehet konfigurálni egy alapértelmezett útvonalat?

3. Hogyan kaphatunk részletes információkat az `ip neighbour` parancsról? Mi történik, ha önmagában futtatjuk?

4. Hogyan készíthetünk biztonsági másolatot az routing table-ről? Hogyan állíthatjuk vissza?

5. Melyik `ip` alparancs használható a spanning tree (feszítőfa) beállításainak konfigurálására?

Összefoglalás

A hálózatot általában a rendszer indítási parancsfájljai vagy egy segédprogram, például a NetworkManager konfigurálja. A legtöbb disztribúció rendelkezik olyan eszközökkel, amelyek szerkesztik az indítóscript konfigurációs fájljait. A részletekért nézzük meg a disztribúció dokumentációját!

A hálózat manuális konfigurálása lehetővé teszi a hatékonyabb hibaelhárítást. Hasznos a minimalista környezetekben, például biztonsági mentésekből való visszaállításhoz vagy új hardverre való áttéréshez.

Az ebben a szakaszban tárgyalt segédprogramok több funkcióval rendelkeznek, mint amennyit ez a lecke tartalmaz. Érdemes átfutni mindegyik man oldalát, hogy megismerkedjünk a rendelkezésre álló lehetőségekkel. Az `ss` és az `ip` parancsok a dolgok modern módját jelentik, míg a többi tárgyalt eszköz, bár még mindig használatban van, már elavultnak számít.

A legjobb módja annak, hogy megismerkedjünk a tárgyalt eszközökkel — a gyakorlás. Egy szerény mennyiségű RAM-mal rendelkező számítógéppel virtuális gépek segítségével virtuális hálózati laboratóriumot hozhatunk létre, amelyen gyakorolhatunk. Három virtuális gép már elég a felsorolt eszközökkel való megismerkedéshez.

A leckében használt parancsok:

`ifconfig`

Hálózati interfészek konfigurálására és állapotuk áttekintésére használt, legacy segédprogram.

`ip`

Modern és sokoldalú segédprogram a hálózati interfészek konfigurálására és állapotuk áttekintésére.

`netstat`

Az aktuális hálózati kapcsolatok és útvonal-információk megtekintésére szolgáló legacy parancs.

`route`

A routing table megtekintésére vagy módosítására használt legacy parancs.

Válaszok a gyakorló feladatokra

1. Milyen parancsokkal lehet listázni a rendszerben lévő hálózati adaptereket?

   Az alábbi parancsok közül bármelyikkel:

   `ip link`, `ifconfig -a`, vagy `ls /sys/class/net`

2. Hogyan lehet ideiglenesen letiltani egy interfészt? Hogyan lehet újra engedélyezni?

   Használhatjuk az `ifconfig` vagy az `ip link` parancsokat:

   `ifconfig`:

   ```
   $ ifconfig wlan1 down
   $ ifconfig wlan1 up
   ```

   `ip link`:

   ```
   $ ip link set wlan1 down
   $ ip link set wlan1 up
   ```

3. Az alábbiak közül melyik fogadható el IPv4 alhálózati maszkként?
   - `255.252.0.0`
   - `/24`

     A többi felsorolt maszk érvénytelen, mert nem választja szét a címet tisztán két részre, ahol az első rész a hálózatot, a második pedig a hosthelyet határozza meg. A maszk bal oldali bitjei mindig `1`, a jobb oldali bitek pedig mindig `0` értékűek.

4. Mely parancsokkal tudjuk ellenőrizni az alapértelmezett útvonalat?

   Használhatjuk a `route`, `netstat -r`, vagy `ip route` parancsokat:

   ```
   $ route
   Kernel IP routing table
   Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
   default         server          0.0.0.0         UG    600    0        0 wlan1
   192.168.1.0     0.0.0.0         255.255.255.0   U     600    0        0 wlan1
   $ netstat -r
   ```

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         server          0.0.0.0         UG        0 0          0 wlan1
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 wlan1
$ ip route
default via 192.168.1.20 dev wlan1 proto static metric 600
192.168.1.0/24 dev wlan1 proto kernel scope link src 192.168.1.24 metric 600
```

5. Hogyan adhatunk hozzá egy második IP-címet egy interfészhez?

   Használhatjuk az `ip address` vagy `ifconfig` parancsokat. Ne feledjük, hogy az `ifconfig` egy legacy eszköz:

   ```
   $ ip addr add 172.16.15.16/16 dev enp0s9 label enp0s9:sub1
   ```

   A `label enp0s9:sub1` parancs egy aliast ad az `enp0s9` parancshoz. Ha nem használjuk a legacy `ifconfig` parancsot, akkor ezt kihagyhatjuk. Ebben az esetben a parancs továbbra is működni fog, de az imént hozzáadott cím nem fog megjelenni az `ifconfig` kimenetében.

   Az `ifconfig`-ot is használhatjuk:

   ```
   $ ifconfig enp0s9:sub1 172.16.15.16/16
   ```

Válaszok a gondolkodtató feladatokra

1. Az `ip` melyik alparancsa használható vlan címkézés (tagging) konfigurálására?

   Van egy `vlan` kapcsoló, amit használhatunk. Az alábbi példában egy hálózati interfészt vlan 20-szal címkézünk fel.

   ```
   # ip link add link enp0s9 name enp0s9.20 type vlan id 20
   ```

2. Hogyan lehet konfigurálni egy alapértelmezett útvonalat?

   A `route` vagy az `ip route` használatával:

   ```
   # route add default gw 192.168.1.1
   # ip route add default via 192.168.1.1
   ```

3. Hogyan kaphatunk részletes információkat az `ip neighbour` parancsról? Mi történik, ha önmagában futtatjuk?

   A man oldalának elolvasásával:

   ```
   $ man ip-neigbour
   ```

   Megjeleníti az ARP cache-t:

   ```
   $ ip neighbour
   10.0.2.2 dev enp0s3 lladdr 52:54:00:12:35:02 REACHABLE
   ```

4. Hogyan készíthetünk biztonsági másolatot az routing table-ről? Hogyan állíthatjuk vissza?

   Az alábbi példa bemutatja a biztonsági másolat készítését és a helyreállítást is:

   ```
   # ip route save > /root/routes/route_backup
   # ip route restore < /root/routes/route_backup
   ```

5. Melyik `ip` alparancs használható a spanning tree (feszítőfa) beállításainak konfigurálására?

   A vlan beállítások menedzseléséhez hasonlóan, az `ip link` a `bridge` típus használatával konfigurálhatja a feszítőfát. Az alábbi példa egy virtuális interfész hozzáadását mutatja 50-es

STP prioritással:

```
# ip link add link enp0s9 name enp0s9.50 type bridge priority 50
```

109.3 Lecke 2

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	109 Hálózati alapok
Fejezet:	109.3 Alapvető hálózati hibaelhárítás
Lecke:	2/2

Bevezetés

A Linux alapú operációs rendszerek számos eszközzel rendelkeznek a hálózati problémák elhárításához. Ez a lecke a leggyakoribbak közül ismertet néhányat. Ezen a ponton már tisztában kell lennünk az OSI vagy más hálózati rétegmodellekkel, az IPv4 vagy IPv6 címzéssel, valamint a routing és a switching alapjaival.

A hálózati kapcsolat tesztelésének legjobb módja az, ha megpróbáljuk használni az alkalmazást. Ha ez nem működik, rengeteg eszköz áll rendelkezésre a probléma diagnosztizálásához.

Kapcsolatok tesztelés a `ping` paranccsal

A `ping` és a `ping6` parancsokkal ICMP echo requestet küldhetünk egy IPv4 vagy IPv6 címre. Ez kis mennyiségű adatot küld a célcímre. Ha a célcím elérhető, akkor az ICMP echo válaszüzenetet küld vissza a feladónak ugyanazokkal az adatokkal, amelyeket a feladó küldött:

```
$ ping -c 3 192.168.50.2
PING 192.168.50.2 (192.168.50.2) 56(84) bytes of data.
64 bytes from 192.168.50.2: icmp_seq=1 ttl=64 time=0.525 ms
```

```
64 bytes from 192.168.50.2: icmp_seq=2 ttl=64 time=0.419 ms
64 bytes from 192.168.50.2: icmp_seq=3 ttl=64 time=0.449 ms

--- 192.168.50.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 0.419/0.464/0.525/0.047 ms
```

```
$ ping6 -c 3 2001:db8::10
PING 2001:db8::10(2001:db8::10) 56 data bytes
64 bytes from 2001:db8::10: icmp_seq=1 ttl=64 time=0.425 ms
64 bytes from 2001:db8::10: icmp_seq=2 ttl=64 time=0.480 ms
64 bytes from 2001:db8::10: icmp_seq=3 ttl=64 time=0.725 ms

--- 2001:db8::10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.425/0.543/0.725/0.131 ms
```

A `-c` kapcsoló a küldendő csomagok számának megadására szolgál. Ha kihagyjuk ezt az opciót, a `ping` és a `ping6` addig küldi a csomagokat, amíg le nem állítjuk, általában a `Ctrl` + `C` billentyűkombinációval.

Csak azért, mert nem tudunk pingelni egy hostot, az még nem jelenti azt, hogy nem tudunk csatlakozni hozzá. Sok szervezet tűzfalai vagy router hozzáférési vezérlési listái mindent blokkolnak, kivéve a rendszerek működéséhez szükséges minimumot. Ez magában foglalja az ICMP kéréseket és válaszokat is. Mivel ezek a csomagok tetszőleges adatokat tartalmazhatnak, egy ügyes támadó felhasználhatja őket adatok kiszivárogtatására.

Útvonalak követése

A `traceroute` és `traceroute6` programok segítségével megnézhetjük, hogy egy csomag milyen útvonalon jut el a célállomásáig. Ez úgy történik, hogy a rendszer több csomagot küld a célállomásra, és minden egyes csomaggal növeli az IP header *Time-To-Live* (TTL) mezőjét. Az útvonal mentén minden router egy TTL túllépett ICMP üzenettel válaszol:

```
$ traceroute 192.168.1.20
traceroute to 192.168.1.20 (192.168.1.20), 30 hops max, 60 byte packets
 1  10.0.2.2 (10.0.2.2)  0.396 ms  0.171 ms  0.132 ms
 2  192.168.1.20 (192.168.1.20)  2.665 ms  2.573 ms  2.573 ms
$ traceroute 192.168.50.2
traceroute to 192.168.50.2 (192.168.50.2), 30 hops max, 60 byte packets
 1  192.168.50.2 (192.168.50.2)  0.433 ms  0.273 ms  0.171 ms
```

```
$ traceroute6 2001:db8::11
traceroute to 2001:db8::11 (2001:db8::11), 30 hops max, 80 byte packets
 1  2001:db8::11 (2001:db8::11)  0.716 ms  0.550 ms  0.641 ms
$ traceroute 2001:db8::11
traceroute to 2001:db8::11 (2001:db8::11), 30 hops max, 80 byte packets
 1  2001:db8::10 (2001:db8::11)  0.617 ms  0.461 ms  0.387 ms
$ traceroute net2.example.net
traceroute to net2.example.net (192.168.50.2), 30 hops max, 60 byte packets
 1  net2.example.net (192.168.50.2)  0.533 ms  0.529 ms  0.504 ms
$ traceroute6 net2.example.net
traceroute to net2.example.net (2001:db8::11), 30 hops max, 80 byte packets
 1  net2.example.net (2001:db8::11)  0.738 ms  0.607 ms  0.304 ms
```

Alapértelmezés szerint a `traceroute` 3 UDP csomagot küld a 33434-es portra, és minden egyes csomagküldéskor növeli ezt a számot. A parancs kimenetében minden sor egy-egy router interfész, amelyen a csomag áthalad. A kimenet egyes soraiban feltüntetett idők az egyes csomagok oda-vissza utazási idejét jelentik. Az IP-cím a kérdéses router interfészének a címe. Ha a `traceroute` képes rá, akkor a router interfészének DNS-nevét használja. Néha az idő helyett `*` látható. Ez azt jelenti, hogy a `traceroute` nem kapta meg az adott csomaghoz tartozó TTL túllépés üzenetét. Amikor ezt kezdjük látni, az gyakran azt jelzi, hogy az utolsó válasz az útvonal utolsó ugrása.

`root`-ként az `-I` kapcsolóval a `traceroute`-t úgy állíthatjuk be, hogy UDP csomagok helyett ICMP kéréseket használjon. Ez gyakran hatékonyabb, mint az UDP, mivel a célállomás nagyobb valószínűséggel válaszol az ICMP kérésre, mint az UDP csomagra:

```
# traceroute -I learning.lpi.org
traceroute to learning.lpi.org (208.94.166.201), 30 hops max, 60 byte packets
 1  047-132-144-001.res.spectrum.com (47.132.144.1)  9.764 ms  9.702 ms  9.693 ms
 2  096-034-094-106.biz.spectrum.com (96.34.94.106)  8.389 ms  8.481 ms  8.480 ms
 3  dtr01hlrgnc-gbe-4-15.hlrg.nc.charter.com (96.34.64.172)  8.763 ms  8.775 ms  8.770 ms
 4  acr01mgtnnc-vln-492.mgtn.nc.charter.com (96.34.67.202)  27.080 ms  27.154 ms  27.151 ms
 5  bbr01gnvlsc-bue-3.gnvl.sc.charter.com (96.34.2.112)  31.339 ms  31.398 ms  31.395 ms
 6  bbr01aldlmi-tge-0-0-0-13.aldl.mi.charter.com (96.34.0.161)  39.092 ms  38.794 ms  38.821
ms
 7  prr01ashbva-bue-3.ashb.va.charter.com (96.34.3.51)  34.208 ms  36.474 ms  36.544 ms
 8  bx2-ashburn.bell.ca (206.126.236.203)  53.973 ms  35.975 ms  38.250 ms
 9  tcore4-ashburnbk_0-12-0-0.net.bell.ca (64.230.125.190)  66.315 ms  65.319 ms  65.345 ms
10  tcore4-toronto47_2-8-0-3.net.bell.ca (64.230.51.22)  67.427 ms  67.502 ms  67.498 ms
11  agg1-toronto47_xe-7-0-0_core.net.bell.ca (64.230.161.114)  61.270 ms  61.299 ms  61.291
ms
12  dis4-clarkson16_5-0.net.bell.ca (64.230.131.98)  61.101 ms  61.177 ms  61.168 ms
```

```
13  207.35.12.142 (207.35.12.142)  70.009 ms  70.069 ms  59.893 ms
14  unassigned-117.001.centrilogic.com (66.135.117.1)  61.778 ms  61.950 ms  63.041 ms
15  unassigned-116.122.akn.ca (66.135.116.122)  62.702 ms  62.759 ms  62.755 ms
16  208.94.166.201 (208.94.166.201)  62.936 ms  62.932 ms  62.921 ms
```

Egyes szervezetek blokkolják az ICMP kéréseket és válaszokat. Ennek megkerülésére használhatjuk a TCP-t. Egy ismert nyitott TCP-port használatával garantálhatjuk azt, hogy a célállomás válaszoljon. A TCP használatához a `-T` kapcsolót kell használnunk, valamint a `-p` kapcsolóval meg kell adnunk a portot. Az ICMP echo kérésekhez hasonlóan ehhez is `root`-nak kell lennünk:

```
# traceroute -m 60 -T -p 80  learning.lpi.org
traceroute to learning.lpi.org (208.94.166.201), 60 hops max, 60 byte packets
 1  * * *
 2  096-034-094-106.biz.spectrum.com (96.34.94.106)  12.178 ms  12.229 ms  12.175 ms
 3  dtr01hlrgnc-gbe-4-15.hlrg.nc.charter.com (96.34.64.172)  12.134 ms  12.093 ms  12.062 ms
 4  acr01mgtnnc-vln-492.mgtn.nc.charter.com (96.34.67.202)  31.146 ms  31.192 ms  31.828 ms
 5  bbr01gnvlsc-bue-3.gnvl.sc.charter.com (96.34.2.112)  39.057 ms  46.706 ms  39.745 ms
 6  bbr01aldlmi-tge-0-0-0-13.aldl.mi.charter.com (96.34.0.161)  50.590 ms  58.852 ms  58.841
ms
 7  prr01ashbva-bue-3.ashb.va.charter.com (96.34.3.51)  34.556 ms  37.892 ms  38.274 ms
 8  bx2-ashburn.bell.ca (206.126.236.203)  38.249 ms  36.991 ms  36.270 ms
 9  tcore4-ashburnbk_0-12-0-0.net.bell.ca (64.230.125.190)  66.779 ms  63.218 ms tcore3-
ashburnbk_100ge0-12-0-0.net.bell.ca (64.230.125.188)  60.441 ms
10  tcore4-toronto47_2-8-0-3.net.bell.ca (64.230.51.22)  63.932 ms  63.733 ms  68.847 ms
11  agg2-toronto47_xe-7-0-0_core.net.bell.ca (64.230.161.118)  60.144 ms  60.443 ms agg1-
toronto47_xe-7-0-0_core.net.bell.ca (64.230.161.114)  60.851 ms
12  dis4-clarkson16_5-0.net.bell.ca (64.230.131.98)  67.246 ms dis4-clarkson16_7-
0.net.bell.ca (64.230.131.102)  68.404 ms dis4-clarkson16_5-0.net.bell.ca (64.230.131.98)
67.403 ms
13  207.35.12.142 (207.35.12.142)  66.138 ms  60.608 ms  64.656 ms
14  unassigned-117.001.centrilogic.com (66.135.117.1)  70.690 ms  62.190 ms  61.787 ms
15  unassigned-116.122.akn.ca (66.135.116.122)  62.692 ms  69.470 ms  68.815 ms
16  208.94.166.201 (208.94.166.201)  61.433 ms  65.421 ms  65.247 ms
17  208.94.166.201 (208.94.166.201)  64.023 ms  62.181 ms  61.899 ms
```

A `ping`-hez hasonlóan a `traceroute`-nak is megvannak a maga korlátai. A tűzfalak és a routerek blokkolhatják a `traceroute`-ból küldött vagy a `traceroute`-ba visszaküldött csomagokat. Ha van `root` hozzáférésünk, vannak olyan lehetőségek, amelyek segítségével pontos eredményeket kaphatunk.

MTU-k feltárása a `tracepath` segítségével

A `tracepath` parancs hasonló a `traceroute` parancshoz, a különbség az, hogy az útvonal mentén követi a *Maximum Transmission Unit* (MTU) méretét. Az MTU vagy egy hálózati interfészen konfigurált beállítás, vagy a legnagyobb protokolladategység hardveres korlátozása, amelyet az interfész továbbítani vagy fogadni tud. A `tracepath` program ugyanúgy működik, mint a `traceroute`, azaz minden csomaggal növeli a TTL értéket. A különbség az, hogy egy nagyon nagy UDP adatcsomagot küld. Szinte elkerülhetetlen, hogy az adatcsomag nagyobb legyen, mint az útvonal mentén a legkisebb MTU-val rendelkező eszköz. Amikor a csomag eléri ezt az eszközt, az eszköz általában egy elérhetetlen célcsomaggal válaszol. Az ICMP elérhetetlen célállomás csomagnak (destination unreachable packet) van egy mezője annak a kapcsolatnak az MTU-jára, amelyen a csomagot elküldené, ha képes lenne rá. A `tracepath` ezután minden további csomagot ezzel a mérettel küld:

```
$ tracepath 192.168.1.20
 1?: [LOCALHOST]                                         pmtu 1500
 1:  10.0.2.2                                              0.321ms
 1:  10.0.2.2                                              0.110ms
 2:  192.168.1.20                                          2.714ms reached
     Resume: pmtu 1500 hops 2 back 64
```

A `traceroute`-tól eltérően IPv6 esetén kifejezetten a `tracepath6`-ot kell használni:

```
$ tracepath 2001:db8::11
tracepath: 2001:db8::11: Address family for hostname not supported
$ tracepath6 2001:db8::11
 1?: [LOCALHOST]                      0.027ms pmtu 1500
 1:  net2.example.net                                      0.917ms reached
 1:  net2.example.net                                      0.527ms reached
     Resume: pmtu 1500 hops 1 back 1
```

A kimenet hasonló a `traceroute`-hoz. A `tracepath` előnye, hogy az utolsó sorban a legkisebb MTU-t adja meg a teljes kapcsolaton. Ez hasznos lehet olyan kapcsolatok hibaelhárításánál, amelyek nem tudják kezelni a töredékeket.

Az előző hibaelhárítási eszközökhöz hasonlóan itt is fennáll annak a lehetősége, hogy az eszközök blokkolják a csomagokat.

Tetszőleges kapcsolatok létrehozása

Az `nc` program, más néven netcat, tetszőleges adatokat küldhet vagy fogadhat TCP vagy UDP hálózati kapcsolaton keresztül. A következő példákból világos lesz a működése.

Íme egy példa egy listener beállítására az `1234` porton:

```
$ nc -l 1234
LPI Example
```

Az `LPI Example` kimenete az alábbi példa után jelenik meg, amely egy netcat feladót állít be, hogy csomagokat küldjön a `net2.example.net` címre az `1234` porton. Az `-l` kapcsolóval megadhatjuk, hogy az `nc` ne küldjön, hanem fogadjon adatokat:

```
$ nc net2.example.net 1234
LPI Example
```

A `Ctrl` + `C` billentyűparanccsal bármilyen rendszeren megszakíthatjuk a kapcsolatot.

A Netcat IPv4 és IPv6 címekkel egyaránt működik, valamint TCP és UDP protokollal is. Még egy primitív távoli shell beállítására is használható.

WARNING

Vegyük figyelembe, hogy az `nc` nem minden telepítése támogatja az `-e` kapcsolót! Mindenképpen meg kell néznünk a telepítéshez tartozó man oldalakat, ahol biztonsági információkat találhatunk erről a kapcsolóról, valamint a távoli rendszeren történő parancsvégrehajtás alternatív módszereiről.

```
$ hostname
net2
$ nc -u -e /bin/bash -l 1234
```

Az `-u` kapcsoló az UDP-t jelöli. Az `-e` arra utasítja a netcat-et, hogy mindent, amit kap, küldjön el az őt követő futtatható program standard bemenetére — ez ebben a példában a `/bin/bash`.

```
$ hostname
net1
$ nc -u net2.example.net 1234
hostname
net2
```

```
pwd
/home/emma
```

Észrevettük, hogy a `hostname` parancs kimenete megegyezik a hallgató hostéval, a `pwd` parancs pedig egy mappát ír ki?

Jelenlegi kapcsolatok és hallgatók megtekintése

A `netstat` és az `ss` programokkal megtekinthetjük az aktuális hallgatók (listeners) és kapcsolatok állapotát. Az `ifconfig` programhoz hasonlóan a `netstat` is egy régi eszköz. Mind a `netstat`, mind az `ss` hasonló kimenettel és beállításokkal rendelkezik. Íme néhány kapcsoló, amely mindkét programban elérhető:

`-a`
: Megjeleníti az összes socketet.

`-l`
: Megjeleníti az összes hallgató socketet.

`-p`
: Megjeleníti a kapcsolathoz tartozó folyamatot.

`-n`
: Megakadályozza a névkeresést mind a portok, mind a címek esetében.

`-t`
: Megjeleníti a TCP kapcsolatokat.

`-u`
: Megjeleníti az UDP kapcsolatokat.

Az alábbi példák a két program általánosan használt beállításkészletének kimenetét mutatják:

```
# netstat -tulnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program
name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      892/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      1141/master
tcp6       0      0 :::22                   :::*                    LISTEN      892/sshd
tcp6       0      0 ::1:25                  :::*                    LISTEN      1141/master
```

```
udp        0      0 0.0.0.0:68              0.0.0.0:*                           692/dhclient
# ss -tulnp
# ss -tulnp
Netid  State      Recv-Q Send-Q     Local Address:Port                    Peer
Address:Port
udp    UNCONN     0      0                     :68                                  *:
users:(("dhclient",pid=693,fd=6))
tcp    LISTEN     0      128                   :22                                  *:
users:(("sshd",pid=892,fd=3))
tcp    LISTEN     0      100            127.0.0.1:25                                 :
users:(("master",pid=1099,fd=13))
tcp    LISTEN     0      128                [::]:22                              [::]:*
users:(("sshd",pid=892,fd=4))
tcp    LISTEN     0      100               [::1]:25                              [::]:*
users:(("master",pid=1099,fd=14))
```

A `Recv-Q` oszlop a socket által fogadott, de a programnak át nem adott csomagok száma. A `Send-Q` oszlop azoknak a csomagoknak a száma, amelyeket a socket elküldött, de a vevő nem nyugtázta. A többi oszlop magától értetődő.

Gyakorló feladatok

1. Milyen parancs(ok)kal tudnánk ICMP echo requestet küldeni a `learning.lpi.org` címre?

2. Hogyan tudnánk meghatározni a `8.8.8.8`-hoz vezető útvonalat?

3. Melyik parancs mutatja meg, hogy a 80-as TCP-porton hallgat-e valamilyen folyamat?

4. Hogyan lehet kideríteni, hogy melyik folyamat hallgat egy porton?

5. Hogyan lehetne meghatározni egy hálózati útvonal maximális MTU-ját?

Gondolkodtató feladatok

1. Hogyan használhatjuk a netcat-et HTTP-kérés küldésére egy webszervernek?

2. Mik lehetnek az okai annak, ha egy host pingelése sikertelen?

3. Nevezzünk meg egy eszközt, amellyel láthatjuk, hogy a hálózati csomagok elérnek vagy elhagynak egy Linux-hostot?

4. Hogyan kényszeríthetjük a `traceroute`-t, hogy másik interfészt használjon?

5. Lehetséges, hogy a `traceroute` jelentse az MTU-kat?

Összefoglalás

A hálózatot általában a rendszer indítási parancsfájljai vagy egy segédprogram, például a NetworkManager konfigurálja. A legtöbb disztribúció rendelkezik olyan eszközökkel, amelyek szerkesztik az indítóscript konfigurációs fájljait. A részletekért nézzük meg a disztribúció dokumentációját!

A hálózat manuális konfigurálása lehetővé teszi a hatékonyabb hibaelhárítást. Hasznos a minimalista környezetekben, például biztonsági mentésekből való visszaállításhoz vagy új hardverre való áttéréshez.

Az ebben a szakaszban tárgyalt segédprogramok több funkcióval rendelkeznek, mint amennyit ez a lecke tartalmaz. Érdemes átfutni mindegyik man oldalát, hogy megismerkedjünk a rendelkezésre álló lehetőségekkel. Az `ss` és az `ip` parancsok a dolgok kezelésének modern módját jelentik, míg a többi tárgyalt eszköz, bár még mindig használatban van, már elavultnak számít.

A legjobb módja annak, hogy megismerkedjünk a tárgyalt eszközökkel — a gyakorlás. Egy szerény mennyiségű RAM-mal rendelkező számítógéppel virtuális gépek segítségével virtuális hálózati laboratóriumot hozhatunk létre, amelyen gyakorolhatunk. Három virtuális gép már elég a felsorolt eszközökkel való megismerkedéshez.

A leckében használt parancsok:

`ping` and `ping6`

ICMP-csomagokat továbbít egy egy távoli host részére, ezzel tesztelve a hálózati kapcsolat elérhetőségét.

`traceroute` and `traceroute6`

Egy hálózaton keresztülvezető útvonal nyomon követésére szolgál a hálózat összeköttetésének meghatározásához.

`tracepath` and `tracepath6`

A hálózaton keresztüli útvonal nyomon követésére, valamint az MTU méretének meghatározására szolgál az útvonal mentén.

`nc`

Tetszőleges kapcsolatok létrehozására szolgál a hálózaton a kapcsolat tesztelésére, valamint a hálózaton elérhető szolgáltatások és eszközök lekérdezésére.

`netstat`

Legacy parancs a rendszer nyitott hálózati kapcsolatainak és statisztikáinak meghatározására.

`ss`

Modern parancs a rendszer nyitott hálózati kapcsolatainak és statisztikáinak meghatározására.

Válaszok a gyakorló feladatokra

1. Milyen parancs(ok)kal tudnánk ICMP echo requestet küldeni a `learning.lpi.org` címre?

   A `ping` vagy a `ping6` használatával:

   ```
   $ ping learning.lpi.org
   ```

   vagy

   ```
   $ ping6 learning.lpi.org
   ```

2. Hogyan tudnánk meghatározni a `8.8.8.8`-hoz vezető útvonalat?

   A `tracepath` vagy `traceroute` parancsokkal.

   ```
   $ tracepath 8.8.8.8
   ```

   vagy

   ```
   $ traceroute 8.8.8.8
   ```

3. Melyik parancs mutatja meg, hogy a 80-as TCP-porton hallgat-e valamilyen folyamat?

   Az `ss`:

   ```
   $ ss -ln | grep ":80"
   ```

   A `netstat`:

   ```
   $ netstat -ln | grep ":80"
   ```

   Bár nem szerepel a vizsga követelményei között, használhatjuk az `lsof`-t is:

   ```
   # lsof -Pi:80
   ```

4. Hogyan lehet kideríteni, hogy melyik folyamat hallgat egy porton?

Ennek megvalósítására sok lehetőség van. Használhatjuk az `lsof`-ot úgy, mint az előző válaszban, kicserélve a port számát. Használhatjuk a `netstat`-ot vagy az `ss`-t a `-p` kapcsolóval. Ne feledjük, hogy a `netstat` egy legacy eszköz.

```
# netstat -lnp | grep ":22"
```

A `netstat` esetén használható kapcsolók működnek az `ss`-el is:

```
# ss -lnp | grep ":22"
```

5. Hogyan lehetne meghatározni egy hálózati útvonal maximális MTU-ját?

   A `tracepath` paranccsal:

```
$ tracepath somehost.example.com
```

Válaszok a gondolkodtató feladatokra

1. Hogyan használhatjuk a netcat-et HTTP-kérés küldésére egy webszervernek?

   A HTTP-request sorának, az esetleges fejléceknek és egy üres sornak a terminálba történő beírásával:

   ```
   $ nc learning.lpi.org 80
   GET /index.html HTTP/1.1
   HOST: learning.lpi.org

   HTTP/1.1 302 Found
   Location: https://learning.lpi.org:443/index.html
   Date: Wed, 27 May 2020 22:54:46 GMT
   Content-Length: 5
   Content-Type: text/plain; charset=utf-8

   Found
   ```

2. Mik lehetnek az okai annak, ha egy host pingelése sikertelen?

   Számos lehetőség van. Lássunk néhány példát:

   - A távoli host leállt.
   - Egy ACL router blokkolja a pingelést.
   - A távoli host tűzfala blokkolja a pinget.
   - Lehet, hogy helytelen hostnevet vagy címet használunk.
   - A névfeloldás helytelen címet ad vissza.
   - A gép hálózati konfigurációja helytelen.
   - A gép tűzfala blokkolja a pinget.
   - A távoli host hálózati konfigurációja helytelen.
   - A gépünk interfésze(i) nincs(enek) összeköttetésben.
   - A távoli gép interfésze(i) nincs(ek) összeköttetésben.
   - A köztünk és a távoli gép közötti hálózati komponens, például switch, kábel vagy router már nem működik.

3. Nevezzünk meg egy eszközt, amellyel láthatjuk, hogy a hálózati csomagok elérnek vagy elhagynak egy Linux-hostot?

A `tcpdump` és a `wireshark` is használható.

4. Hogyan kényszeríthetjük a `traceroute`-t, hogy másik interfészt használjon?

   Az `-i` kapcsoló használatával:

```
$ traceroute -i eth2 learning.lpi.org
traceroute -i eth2 learning.lpi.org
traceroute to learning.lpi.org (208.94.166.201), 30 hops max, 60 byte packets
...
```

5. Lehetséges, hogy a `traceroute` jelentse az MTU-kat?

   Igen, az `--mtu` kapcsolóval:

```
# traceroute -I --mtu  learning.lpi.org
traceroute to learning.lpi.org (208.94.166.201), 30 hops max, 65000 byte packets
 1  047-132-144-001.res.spectrum.com (47.132.144.1)  9.974 ms F=1500  10.476 ms  4.743 ms
 2  096-034-094-106.biz.spectrum.com (96.34.94.106)  8.697 ms  9.963 ms  10.321 ms
...
```

109.4 Kliensoldali DNS konfigurációja

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 102, Objective 109.4

Súlyozás

2

Kulcsfontosságú ismeretek

- Távoli DNS-szerverek lekérdezése
- Helyi névfeloldás beállítása és távoli DNS-szerverek használata
- A névfeloldás sorrendjének módosítása
- A névfeloldással kapcsolatos hibák elhárítása
- A `systemd-resolved` ismerete

A használt fájlok, kifejezések és segédprogramok listája

- `/etc/hosts`
- `/etc/resolv.conf`
- `/etc/nsswitch.conf`
- `host`
- `dig`
- `getent`

109.4 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	109 Hálózati alapok
Fejezet:	109.4 Kliensoldali DNS konfigurációja
Lecke:	1 of 1

Bevezetés

Ez a lecke a kliensoldali névfeloldás konfigurálásával és néhány CLI névfeloldó eszköz használatával foglalkozik.

Az IP-címek, UID-k, GID-k és a minden máshoz tartozó egyéb számok megjegyzése és karbantartása nem kivitelezhető. A névfeloldó szolgáltatások a könnyen megjegyezhető neveket számokra fordítják le és fordítva. Ez a lecke a hostnevek feloldására összpontosít, de hasonló folyamat zajlik a felhasználói nevek, csoportnevek, portszámok és számos más név esetében is.

A névfeloldás folyamata

A neveket számokká feloldó programok szinte mindig a szabványos C könyvtár által biztosított függvényeket használják, ami Linux rendszereken a GNU projekt glibc-je. Ezek a függvények először is elolvassák az `/etc/nsswitch.conf` fájlt az adott névtípus feloldására vonatkozó utasításokért. Ez a lecke a hostnév feloldására összpontosít, de ugyanez a folyamat más típusú nevek feloldása esetén is alkalmazható. Miután a folyamat beolvasta az `/etc/nsswitch.conf` állományt, a megadott módon megkeresi a nevet. Mivel az `/etc/nsswitch.conf` támogatja a bővítményeket, a következő lépés bármi lehet. Miután a függvény befejezte a név vagy szám

keresését, az eredményt visszaadja a hívó folyamatnak.

DNS osztályok

A DNS három rekordosztállyal rendelkezik: IN, HS és CH. Ebben a leckében minden DNS-lekérdezés IN típusú lesz. Az IN osztály a TCP/IP stacket használó internetcímek számára készült. A CH a ChaosNet, egy rövid életű hálózati technológia, amelyet már nem használnak. A HS osztály a Hesiod jelölése. A Hesiod olyan dolgok tárolására szolgál, mint a jelszó és a csoportbejegyzések a DNS-ben. A Hesiod túlmutat ennek a leckének a keretein.

Az `/etc/nsswitch.conf` értelmezése

Ezen fájl megismerésének a legjobb módja a man oldal elolvasása, amely a Linux man-pages projekt része. Ez a legtöbb rendszeren elérhető, a `man nsswitch.conf` paranccsal, de itt is megtalálható: https://man7.org/linux/man-pages/dir_section_5.html

Az alábbiakban látható egy egyszerű példa a `/etc/nsswitch.conf` állományra a man oldalról:

```
passwd:         compat
group:          compat
shadow:         compat

hosts:          dns [!UNAVAIL=return] files
networks:       nis [NOTFOUND=return] files
ethers:         nis [NOTFOUND=return] files
protocols:      nis [NOTFOUND=return] files
rpc:            nis [NOTFOUND=return] files
services:       nis [NOTFOUND=return] files
# This is a comment. It is ignored by the resolution functions.
```

A fájl oszlopokba van rendezve. A bal szélső oszlop a névadatbázis típusa. A többi oszlop a resolver függvények által a névkereséshez használandó módszerek. A módszereket balról jobbra haladva a függvények követik. A `[]`-el jelölt oszlopok arra szolgálnak, hogy a közvetlenül balra lévő oszlophoz némi korlátozott feltételes logikát adjanak.

Tegyük fel, hogy egy folyamat megpróbálja feloldani a `learning.lpi.org` hostnevet. Ez egy megfelelő C könyvtárhívást (valószínűleg `gethostbyname`) indítana. Ez a függvény ezután beolvassa az `/etc/nsswitch.conf` állományt. Mivel a folyamat egy hostnevet keres, a `hosts` kezdetű sort fogja megtalálni. Ezután megpróbálja a DNS-t használni a név feloldásához. A következő oszlop, a `[!UNAVAIL=return]` azt jelenti, hogy ha a szolgáltatás *nem* nem-elérhető, akkor ne próbálkozzon a következő forrással, azaz ha a DNS elérhető, akkor ne próbálja meg

feloldani a hostnevet, még akkor sem, ha a névszerverek nem képesek rá. Ha a DNS nem elérhető, akkor folytassa a következő forrással. Ebben az esetben a következő forrás a `files`.

Ha egy oszlopot `[result=action]` formátumban látunk, az azt jelenti, hogy ha a bal oldali oszlopban a resolver keresése `result`, akkor `action` történik. Ha a `result` előtt egy `!` áll, az azt jelenti, hogy ha az eredmény nem `result`, akkor az `action`-t kell végrehajtani. A lehetséges eredmények és műveletek leírását lsd. a man oldalon.

Ha egy folyamat megpróbálna feloldani egy portszámot egy szolgáltatásnévhez, akkor a `services` sort olvasná. Az első felsorolt forrás a NIS. A NIS a *Network Information Service* rövidítése (néha arany oldalakként is emlegetik). Ez egy régi szolgáltatás, amely lehetővé tette az olyan dolgok központi kezelését, mint például a felhasználók, de az alacsony biztonsági szintje miatt ma már ritkán használják. A következő oszlop a `[NOTFOUND=return]`, ami azt jelenti, hogy ha a keresés teljesült, de a szolgáltatást nem találtuk meg, akkor hagyjuk abba a keresést. Ha az előbb említett feltétel nem áll fenn, használjuk a helyi fájlokat.

Minden, ami a "#"-től jobbra van, kommentnek minősül, és figyelmen kívül marad.

Az `/etc/resolv.conf` fájl

Az `/etc/resolv.conf` fájlt a DNS-en keresztüli hostfeloldás konfigurálására használják. Néhány disztribúcióban vannak olyan indítóscriptek, daemonok és egyéb eszközök, amelyek ebbe a fájlba írnak. Tartsuk ezt szem előtt, amikor kézzel szerkesztjük ezt a fájlt! Ebben az esetben ellenőrizzük a disztribúció és a hálózati konfigurációs eszközök dokumentációját. Néhány eszköz, például a Network Manager hagy egy megjegyzést a fájlban, amely tudatja, hogy a kézi módosítások felülíródnak.

A `/etc/nsswitch.conf` fájlhoz hasonlóan ehhez a fájlhoz is tartozik egy man oldal. Ez a `man resolv.conf` paranccsal vagy a https://man7.org/linux/man-pages/man5/resolv.conf.5.html címen érhető el.

A fájlformátum meglehetősen egyszerű. A bal oldali oszlopban a `name` opció található. A többi oszlop ugyanabban a sorban az opció értékét tartalmazza.

A leggyakoribb opció a `nameserver`. A DNS-szerver IPv4 vagy IPv6 címének megadására szolgál. Ezen könyv írásának időpontjában legfeljebb három névszervert adhatunk meg. Ha az `/etc/resolv.conf` fájlban nincs `nameserver` opció, akkor a rendszer alapértelmezés szerint a helyi gépen lévő névszervert fogja használni.

Alább látható egy reprezentatív példafájl egyszerű, általános konfigurációk esetén:

```
search lpi.org
```

```
nameserver 10.0.0.53
nameserver fd00:ffff::2:53
```

A `search` opció a rövid keresések engedélyezésére szolgál. A példában az `lpi.org` van beállítva egyetlen keresési domainként. Ez azt jelenti, hogy a domainrész nélküli hostnév feloldására tett kísérletekhez a keresés előtt a `.lpi.org` lesz csatolva. Ha például egy `learning` nevű hostot keresnénk, a resolver a `learning.lpi.org` címet keresné. Legfeljebb hat keresési domain konfigurálható.

Egy másik gyakori opció a `domain`. Ezt a helyi domainnév beállítására használjuk. Ha ez az opció hiányzik, akkor alapértelmezés szerint a gép hostnevében az első `.` után minden szerepel. Ha a hostnév nem tartalmaz egy `.`-t, akkor a rendszer feltételezi, hogy a gép a gyökérdomain része. A `search`-hez hasonlóan a `domain` is használható rövid névkereséshez.

Ne feledjük, hogy a `domain` és a `search` kölcsönösen kizárják egymást! Ha mégis van mindkettő, akkor a fájlban lévő utolsó példány lesz használatban.

Számos beállítással befolyásolható a resolver viselkedése. Ezek beállításához az `options` kulcsszót használjuk, amelyet a beállítandó opció neve követ, és adott esetben egy `:`, amelyet pedig az érték. Az alábbiakban látható egy példa a timeout opció beállítására, amely azt az időt adja meg másodpercben kifejezve, ameddig a resolver várni fog egy névszerverre, mielőtt feladja:

```
option timeout:3
```

A `resolv.conf`-nak vannak más beállításai is, de ezek a leggyakoribbak.

Az `/etc/hosts` fájl

Az `/etc/hosts` fájlt használjuk a nevek IP-címekre való feloldására és fordítva. Az IPv4 és az IPv6 egyaránt támogatott. A bal oldali oszlop az IP-cím, a többi a címhez tartozó nevek. Az `/etc/hosts` leggyakoribb felhasználási területe az olyan hostok és címek, ahol nem lehet DNS, ilyenek például a loopback címek. Az alábbi példában a kritikus infrastruktúra-elemek IP-címei vannak definiálva.

Íme egy reális példa egy `/etc/hosts` fájlra:

```
127.0.0.1       localhost
127.0.1.1       proxy
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters
```

```
10.0.0.1        gateway.lpi.org gateway gw
fd00:ffff::1    gateway.lpi.org gateway gw

10.0.1.53       dns1.lpi.org
fd00:ffff::1:53 dns1.lpi.org
10.0.2.53       dns2.lpi.org
fd00:ffff::2:53 dns2.lpi.org
```

systemd-resolved

A systemd biztosít egy `systemd-resolved` nevű szolgáltatást, ami pedig az mDNS-t, a DNS-t és az LLMNR-t. Amikor fut, akkor a DNS-kérelmekre figyel a `127.0.0.53` címen. Teljes értékű DNS-szervert_nem_ biztosít. A hozzá érkező DNS-kéréseket az `/etc/systemd/resolv.conf` vagy `/etc/resolv.conf` állományban konfigurált szerverek lekérdezésével keresi. Ha ezt használnánk, használjuk az `/etc/nsswitch.conf` állományban a `resolve`-t a `hosts` helyett! Ne feledjük, hogy a `systemd-resolved` mappát tartalmazó csomag alapértelmezés szerint nem feltétlenül van telepítve.

Eszközök a névfeloldáshoz

A Linux-felhasználók számára számos eszköz áll rendelkezésre a névfeloldáshoz. Ebben a leckében hárommal foglalkozunk. Az egyik, a `getent` arra jó, hogy megnézzük, hogyan oldódnak fel a valós kérések. Egy másik parancs a `host`, amely egyszerű DNS-lekérdezésekhez hasznos. A `dig` nevű program az összetett DNS műveletek elvégzésére alkalmas, amely segíthet a DNS-szerver problémáinak az elhárításában.

A `getent` parancs

A `getent` segédprogram a névszolgáltatási adatbázisok bejegyzéseinek megjelenítésére szolgál. Bármilyen, az `/etc/nsswitch.conf` állományban konfigurálható forrásból kérdezhetünk le bejegyzéseket.

A `getent` használatához a parancs után meg kell adnunk a feloldani kívánt névtípust, valamint opcionálisan megadhatunk egy konkrét keresendő bejegyzést. Ha csak a név típusát adjuk meg, a `getent` megpróbálja megjeleníteni az összes ilyen típusú bejegyzést:

```
$ getent hosts
127.0.0.1       localhost
127.0.1.1       proxy
10.0.1.53       dns1.lpi.org
```



```
10.0.2.53       dns2.lpi.org
127.0.0.1       localhost ip6-localhost ip6-loopback
$ getent hosts dns1.lpi.org
fd00:ffff::1:53 dns1.lpi.org
```

A glibc 2.2.5 verziójától kezdve a `-s` kapcsolóval kényszeríthetjük a `getent`-et egy adott adatforrás használatára. Az alábbi példa ezt mutatja be:

```
$ getent -s files hosts learning.lpi.org
::1             learning.lpi.org
$ getent -s dns hosts learning.lpi.org
208.94.166.198  learning.lpi.org
```

A `host` parancs

A `host` egy egyszerű program a DNS-bejegyzések keresésére. Ha a `host`-nak megadunk egy nevet kapcsolók nélkül, visszaadja az A, AAAA és MX rekordkészleteket. Ha IPv4 vagy IPv6 címet adunk meg, akkor a PTR rekordot adja ki, ha van ilyen:

```
$ host wikipedia.org
wikipedia.org has address 208.80.154.224
wikipedia.org has IPv6 address 2620:0:861:ed1a::1
wikipedia.org mail is handled by 10 mx1001.wikimedia.org.
wikipedia.org mail is handled by 50 mx2001.wikimedia.org.
$ host 208.80.154.224
224.154.80.208.in-addr.arpa domain name pointer text-lb.eqiad.wikimedia.org.
```

Ha egy specifikus rekordítpust keresünk, használjuk a `host -t` parancsot:

```
$ host -t NS lpi.org
lpi.org name server dns1.easydns.com.
lpi.org name server dns3.easydns.ca.
lpi.org name server dns2.easydns.net.
$ host -t SOA lpi.org
lpi.org has SOA record dns1.easydns.com. zone.easydns.com. 1593109612 3600 600 1209600 300
```

A `host` egy adott névszerver lekérdezésére is használható, ha nem akarjuk használni az `/etc/resolv.conf` állományban találhatóakat. Egyszerűen csak adjuk meg a használni kívánt szerver IP-címét vagy hostnevét az utolsó argumentumként:

```
$ host -t MX lpi.org dns1.easydns.com
Using domain server:
Name: dns1.easydns.com
Address: 64.68.192.10#53
Aliases:

lpi.org mail is handled by 10 aspmx4.googlemail.com.
lpi.org mail is handled by 10 aspmx2.googlemail.com.
lpi.org mail is handled by 5 alt1.aspmx.l.google.com.
lpi.org mail is handled by 0 aspmx.l.google.com.
lpi.org mail is handled by 10 aspmx5.googlemail.com.
lpi.org mail is handled by 10 aspmx3.googlemail.com.
lpi.org mail is handled by 5 alt2.aspmx.l.google.com.
```

A `dig` parancs

Egy másik eszköz a DNS-szerverek lekérdezésére a `dig`. Ez a parancs sokkal részletesebb, mint a `host`. Alapértelmezés szerint a `dig` A rekordokat kérdez le, valószínűleg túl részletes egy egyszerű IP-cím vagy hostnév kereséséhez. A `dig` egyszerű keresések esetén működik, de inkább a DNS-szerver konfigurációjának hibakeresésére alkalmas:

```
$ dig learning.lpi.org

; <<>> DiG 9.11.5-P4-5.1+deb10u1-Debian <<>> learning.lpi.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63004
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: ca7a415be1cec45592b082665ef87f3483b81ddd61063c30 (good)
;; QUESTION SECTION:
;learning.lpi.org.      IN  A

;; ANSWER SECTION:
learning.lpi.org.   600 IN  A   208.94.166.198

;; AUTHORITY SECTION:
lpi.org.        86400   IN  NS  dns2.easydns.net.
lpi.org.        86400   IN  NS  dns1.easydns.com.
lpi.org.        86400   IN  NS  dns3.easydns.ca.
```

```
;; ADDITIONAL SECTION:
dns1.easydns.com.   172682  IN  A   64.68.192.10
dns2.easydns.net.   170226  IN  A   198.41.222.254
dns1.easydns.com.   172682  IN  AAAA    2400:cb00:2049:1::a29f:1835
dns2.easydns.net.   170226  IN  AAAA    2400:cb00:2049:1::c629:defe

;; Query time: 135 msec
;; SERVER: 192.168.1.20#53(192.168.1.20)
;; WHEN: Sun Jun 28 07:29:56 EDT 2020
;; MSG SIZE  rcvd: 266
```

Láthatjuk, hogy a `dig` rengeteg információt szolgáltat. A kimenet szakaszokra van osztva. Az első szakasz a `dig` telepített verziójáról és a küldött lekérdezésről, valamint a parancshoz használt kapcsolókról ad információt. Ezután a lekérdezéssel és a válasszal kapcsolatos információk következnek.

A következő szakasz a használt EDNS-kiterjesztésekkel és a lekérdezéssel kapcsolatos információkat mutatja be. A példában a cookie kiterjesztést használjuk. A `dig` a `learning.lpi.org` címhez keres egy A rekordot.

A következő szakasz a lekérdezés eredményét mutatja be. A második oszlopban szereplő szám az erőforrás TTL-jét jelenti másodpercben.

A kimenet további része a domain névszervereiről nyújt információt, beleértve annak NS rekordjait, valamint a domain NS rekordjában szereplő szerverek A és AAAA rekordjait.

Ahogy a `host` esetében, úgy most is megadhatjuk a rekord típusát a `-t` kapcsolóval:

```
$ dig -t SOA lpi.org

; <<>> DiG 9.11.5-P4-5.1+deb10u1-Debian <<>> -t SOA lpi.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16695
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 6

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 185c67140a63baf46c4493215ef8906f7bfbe15bdca3b01a (good)
;; QUESTION SECTION:
;lpi.org.           IN  SOA
```

```
;; ANSWER SECTION:
lpi.org.        600 IN  SOA dns1.easydns.com. zone.easydns.com. 1593109612 3600 600 1209600
300

;; AUTHORITY SECTION:
lpi.org.        81989   IN  NS  dns1.easydns.com.
lpi.org.        81989   IN  NS  dns2.easydns.net.
lpi.org.        81989   IN  NS  dns3.easydns.ca.

;; ADDITIONAL SECTION:
dns1.easydns.com.   168271  IN  A   64.68.192.10
dns2.easydns.net.   165815  IN  A   198.41.222.254
dns3.easydns.ca.    107 IN  A   64.68.196.10
dns1.easydns.com.   168271  IN  AAAA    2400:cb00:2049:1::a29f:1835
dns2.easydns.net.   165815  IN  AAAA    2400:cb00:2049:1::c629:defe

;; Query time: 94 msec
;; SERVER: 192.168.1.20#53(192.168.1.20)
;; WHEN: Sun Jun 28 08:43:27 EDT 2020
;; MSG SIZE  rcvd: 298
```

A dig számos lehetőséget kínál mind a kimenet, mind a szervernek küldött lekérdezés finomhangolására. Ezek az kapcsolók `+`-al kezdődnek. Az egyik ilyen a `short`, amely az eredményen kívül minden kimenetet elnyom:

```
$ dig +short lpi.org
65.39.134.165
$ dig +short -t SOA lpi.org
dns1.easydns.com. zone.easydns.com. 1593109612 3600 600 1209600 300
```

Íme egy példa a cookie EDNS kiterjesztés kikapcsolására:

```
$ dig +nocookie -t MX lpi.org

; <<>> DiG 9.11.5-P4-5.1+deb10u1-Debian <<>> +nocookie -t MX lpi.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47774
;; flags: qr rd ra; QUERY: 1, ANSWER: 7, AUTHORITY: 3, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
```

```
;; QUESTION SECTION:
;lpi.org.           IN  MX

;; ANSWER SECTION:
lpi.org.        468 IN  MX  0 aspmx.l.google.com.
lpi.org.        468 IN  MX  10 aspmx4.googlemail.com.
lpi.org.        468 IN  MX  10 aspmx5.googlemail.com.
lpi.org.        468 IN  MX  10 aspmx2.googlemail.com.
lpi.org.        468 IN  MX  10 aspmx3.googlemail.com.
lpi.org.        468 IN  MX  5 alt2.aspmx.l.google.com.
lpi.org.        468 IN  MX  5 alt1.aspmx.l.google.com.

;; AUTHORITY SECTION:
lpi.org.        77130   IN  NS  dns2.easydns.net.
lpi.org.        77130   IN  NS  dns3.easydns.ca.
lpi.org.        77130   IN  NS  dns1.easydns.com.

;; ADDITIONAL SECTION:
dns1.easydns.com.   76140   IN  A   64.68.192.10
dns2.easydns.net.   73684   IN  A   198.41.222.254
dns1.easydns.com.   76140   IN  AAAA    2400:cb00:2049:1::a29f:1835
dns2.easydns.net.   73684   IN  AAAA    2400:cb00:2049:1::c629:defe

;; Query time: 2 msec
;; SERVER: 192.168.1.20#53(192.168.1.20)
;; WHEN: Mon Jun 29 10:18:58 EDT 2020
;; MSG SIZE  rcvd: 389
```

Gyakorló feladatok

1. Mit tesz a lent látható parancs?

```
getent group openldap
```

2. Mi a legnagyobb különbség a `getent`, és a többi tárgyalt eszköz, a `host` és a `dig` között?

3. A `dig` és a `host` melyik kapcsolója határozza meg a lekérdezni kívánt rekord típusát?

4. Az alábbiak közül melyik `/etc/hosts` bejegyzés megfelelő?

`::1 localhost`	
`localhost 127.0.0.1`	

5. A `getent` melyik kapcsolója határozza meg, hogy melyik adatforrást kell használni a kereséshez?

Gondolkodtató feladatok

1. Ha szövegszerkesztővel szerkesztenénk az alábbi `/etc/resolv.conf` fájlt, mi történne?

```
# Generated by NetworkManager
nameserver 192.168.1.20
```

A módosításokat felülírja a NetworkManager.	
A NetworkManager frissíti a konfigurációját a módosításokkal.	
A módosítások nem befolyásolják a rendszert.	
A NetworkManager letiltásra kerül.	

2. Mit csinál az alábbi sor a `/etc/nsswitch.conf` fájlban:

```
hosts: files [SUCCESS=continue] dns
```

3. Az alábbi `/etc/resolv.conf` állományt figyelembe véve miért nem oldja fel a rendszer a neveket a DNS-en keresztül?

```
search lpi.org
#nameserver fd00:ffff::1:53
#nameserver 10.0.1.53
```

4. Mit csinál a `dig +noall +answer +question lpi.org` parancs?

5. Hogyan lehet felülbírálni a `dig` alapértelmezett beállításait anélkül, hogy megadnánk őket a parancssorban?

Összefoglalás

A `getent` parancs egy nagyszerű eszköz a feloldással kapcsolatos hívások eredményeinek megtekintésére. Egyszerű DNS-lekérdezések esetén könnyen használható és egyértelmű választ a `host` parancs ad. Ha részletes információra vagy egy DNS-lekérdezés finomhangolására van szükség, akkor valószínűleg a `dig` a legjobb megoldás.

A Linux a megosztott könyvtári bővítmények hozzáadásának és a resolver viselkedésének konfigurálhatóságának köszönhetően kiválóan támogatja a különböző típusú név- és számfeloldásokat. A `getent` program használható a nevek feloldására a resolver könyvtárak segítségével. A `host` és a `dig` programmal lekérdezhetők a DNS-szerverek.

Az `/etc/nsswitch.conf` fájl a resolver viselkedésének konfigurálására szolgál. Lehetőség van az adatforrások megváltoztatására és néhány egyszerű feltételes logika hozzáadására a több forrással rendelkező névtípusok esetében.

A DNS beállítása az `/etc/resolv.conf` szerkesztésével történik. Sok disztribúció rendelkezik olyan eszközökkel, amelyek kezelik ezt a fájlt helyettünk, ezért mindenképpen ellenőrizzük az általunk használt rendszer dokumentációját, hogy a kézi módosítások megmaradnak-e.

Az `/etc/hosts` fájlt arra használjuk, hogy a hostneveket IP-címekre oldjuk fel, és fordítva. Általában olyan nevek definiálására használjuk, mint például a `localhost`, amelyek nem érhetőek el a DNS-en keresztül.

A leckében tárgyalt konfigurációs fájlokba lehet kommenteket írni. A `#` jobb oldalán lévő szöveget a rendszer figyelmen kívül fogja hagyni.

Válaszok a gyakorló feladatokra

1. Mit tesz a lent látható parancs?

   ```
   getent group openldap
   ```

   A program beolvassa az `/etc/nsswitch.conf` állományt, megkeresi az `openldap` csoportot a felsorolt források közül, és ha megtalálja, megjeleníti a csoportra vonatkozó információkat.

2. Mi a legnagyobb különbség a `getent`, és a többi tárgyalt eszköz, a `host` és a `dig` között?

   A `getent` a resolver könyvtárak segítségével keres neveket, a többi csak a DNS-t kérdezi le. A `getent` használható a `/etc/nsswitch.conf` állomány és a rendszerben használt névfeloldó könyvtárak konfigurációjának hibakeresésére. A `host` és a `dig` a DNS rekordok lekérdezésére szolgál.

3. A `dig` és a `host` melyik kapcsolója határozza meg a lekérdezni kívánt rekord típusát?

   Mindkét program használja a `-t` kapcsolót az általunk keresett rekord típusának specifikálásához.

4. Az alábbiak közül melyik `/etc/hosts` bejegyzés megfelelő?

`::1 localhost`	X
`localhost 127.0.0.1`	

   Az `::1 localhost` sor a helyes. A bal oszlop mindig egy IPv4 vagy IPv6 cím.

5. A `getent` melyik kapcsolója határozza meg, hogy melyik adatforrást kell használni a kereséshez?

   A `-s` kapcsolóval lehet meghatározni az adatforrást. Például:

   ```
   $ getent -s files hosts learning.lpi.org
   192.168.10.25   learning.lpi.org
   $ getent -s dns hosts learning.lpi.org
   208.94.166.198  learning.lpi.org
   ```

Válaszok a gondolkodtató feladatokra

1. Ha szövegszerkesztővel szerkesztenénk az alábbi `/etc/resolv.conf` fájlt, mi történne?

```
# Generated by NetworkManager
nameserver 192.168.1.20
```

A módosításokat felülírja a NetworkManager.	X
A NetworkManager frissíti a konfigurációját a módosításokkal.	
A módosítások nem befolyásolják a rendszert.	
A NetworkManager letiltásra kerül.	

2. Mit csinál az alábbi sor a `/etc/nsswitch.conf` fájlban:

```
hosts: files [SUCCESS=continue] dns
```

   A hostnevek keresése először az `/etc/hosts` fájlokat, majd a DNS-t ellenőrzi. Ha a fájlokban és a DNS-ben is találunk bejegyzést, akkor a DNS-ben található bejegyzést fogja használni.

3. Az alábbi `/etc/resolv.conf` állományt figyelembe véve miért nem oldja fel a rendszer a neveket a DNS-en keresztül?

```
search lpi.org
#nameserver fd00:ffff::1:53
#nameserver 10.0.1.53
```

   Mindkét DNS-szerver ki van kommentezve, valamint a helyi hoston nem fut DNS-szerver.

4. Mit csinál a `dig +noall +answer +question lpi.org` parancs?

   Megnézi az `lpi.org` A rekordját, és csak a lekérdezést és a választ jeleníti meg.

5. Hogyan lehet felülbírálni a `dig` alapértelmezett beállításait anélkül, hogy megadnánk őket a parancssorban?

   Létrehozunk egy `.digrc` fájlt a home mappánkban.

Linux Professional Institute

Témakör 110: Biztonság

110.1 Biztonsági adminisztrációs feladatok elvégzése

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 102, Objective 110.1

Súlyozás

3

Kulcsfontosságú ismeretek

- A rendszer ellenőrzése a suid/sgid bitet tartalmazó fájlok keresésére
- Felhasználói jelszavak és jelszó-öregedési információk beállítása vagy módosítása
- Képesség az nmap és a netstat használatára egy rendszer nyitott portjainak felderítésére
- Korlátok beállítása a felhasználói bejelentkezésekre, folyamatokra és memóriahasználatra vonatkozóan
- Annak megállapítása, hogy mely felhasználók jelentkeztek be a rendszerbe, vagy éppen vannak most bejelentkezve
- Alapvető sudo konfiguráció és használat

A használt fájlok, kifejezések és segédprogramok listája

- `find`
- `passwd`
- `fuser`
- `lsof`
- `nmap`
- `chage`
- `netstat`

- `sudo`
- `/etc/sudoers`
- `su`
- `usermod`
- `ulimit`
- `who`, `w`, `last`

110.1 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	110 Biztonság
Fejezet:	110.1 Biztonsági adminisztrációs feladatok elvégzése
Lecke:	1/1

Bevezetés

A biztonság elengedhetetlen a rendszeradminisztrációban. Jó Linux rendszergazdaként számos dolgot kell szemmel tartanunk, például a fájlok speciális jogosultságait, a felhasználói jelszavak öregedését, a nyitott portokat és socketeket, a rendszer erőforrásainak korlátozását, a bejelentkezett felhasználók kezelését és a jogosultságok kiterjesztését a `su` és `sudo` segítségével. Ebben a leckében ezek közül a témák közül mindegyiket áttekintjük.

SUID és SGID beállítású fájlok ellenőrzése

A hagyományos *read* (olvasás), *write* (írás) és *execute* (futtatás, végrehajtás) jogosultságok mellett a Linux rendszerben a fájlok speciális jogosultságokkal is rendelkezhetnek, mint például a *SUID* vagy a *SGID* bitek.

A SUID bit lehetővé teszi, hogy a fájlt a fájl tulajdonosának jogosultságaival lehessen végrehajtani. Számszerűen `4000`, szimbolikusan pedig `s` vagy `S` jelöli a tulajdonos *execute* jogosultságának bitjén. Klasszikus példa a SUID jogosultsággal rendelkező futtatható fájlra a `passwd`:

```
carol@debian:~$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 63736 jul 27  2018 /usr/bin/passwd
```

A kisbetűs `s` az `rws`-ben a SUID jelenlétét jelzi a fájlban — a *execute* jogosultsággal együtt. A nagybetűs `S` ehelyett (`rwS`) azt jelenti, hogy az alapjául szolgáló *execute* joogosultság nincs beállítva.

NOTE

A `passwd`-ről a következő leckében lesz szó. A segédprogramot leginkább a `root` használja a felhasználók jelszavainak beállítására/megváltoztatására (pl.: `passwd carol`). Azonban a normál felhasználók is használhatják a saját jelszavaik módosítására. Ezért a SUID beállítással együtt kezeljük.

Másik részről az SGID bitet mind fájlokra, mind mappákra be lehet állítani. A fájlok esetében a viselkedése megegyezik a SUID-éval, de a jogosultságok a csoport tulajdonosához tartoznak. Ha azonban egy mappán van beállítva, akkor a mappában létrehozott összes fájl a mappa csoportjának tulajdonjogát örökli. A SUID-hoz hasonlóan az SGID-t is a csoport *execute* jogosultsági bitjén lévő `s` vagy `S` jelképezi. Számszerű jelölése `2000`. Az SGID-t egy mappán a `chmod` használatával állíthatjuk be. A hagyományos jogosultságokhoz (esetünkben `755`) hozzá kell adni a `2` (SGID) értéket:

```
carol@debian:~$ ls -ld shared_directory
drwxr-xr-x 2 carol carol 4096 may 30 23:55 shared_directory
carol@debian:~$ sudo chmod 2755 shared_directory/
carol@debian:~$ ls -ld shared_directory
drwxr-sr-x 2 carol carol 4096 may 30 23:55 shared_directory
```

A SUID és SGID beállítással rendelkező fájlok kereséséhez használhatjuk a `find` parancsot és a `-perm` opciót. Numerikus és szimbolikus értékeket egyaránt megadhatunk. Az értékek — viszont — önmagukban is átadhatók, vagy egy kötőjel (`-`) vagy egy perjel (`/`) előzi meg őket. A jelentés a következő:

`-perm numeric-value` or **`-perm symbolic-value`**
: speciális jogosultsággal rendelkező fájlok *kizárólagos* keresése

`-perm -numeric-value` or **`-perm -symbolic-value`**
: speciális és egyéb jogosultságokkal rendelkező fájlok keresése

`-perm /numeric-value` or **`-perm /symbolic-value`**
: speciális vagy egyéb jogosultságokkal rendelkező fájlok keresése

Ha például a jelenlegi mappában *csak* SUID beállítású fájlokat keresünk, akkor a következő parancsot használjuk:

```
carol@debian:~$ find . -perm 4000
carol@debian:~$ touch file
carol@debian:~$ chmod 4000 file
carol@debian:~$ find . -perm 4000
./file
```

Figyeljük meg, hogy - mivel nem volt egyetlen olyan fájl sem, amely kizárólag a SUID-et tartalmazta volna — létrehoztunk egyet, hogy látható legyen a kimenet. Ugyanezt a parancsot szimbolikus jelöléssel is futtathatjuk:

```
carol@debian:~$ find . -perm u+s
./file
```

A SUID-nak megfelelő fájlok kereséséhez (függetlenül minden más jogosultságtól) az `/usr/bin/` mappában a következő parancsok valamelyikét használhatjuk:

```
carol@debian:~$ sudo find /usr/bin -perm -4000
/usr/bin/umount
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/mount
/usr/bin/passwd
/usr/bin/chsh
/usr/bin/sudo
/usr/bin/su
carol@debian:~$ sudo find /usr/bin -perm -u+s
/usr/bin/umount
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/mount
/usr/bin/passwd
/usr/bin/chsh
/usr/bin/sudo
/usr/bin/su
```

Ha ugyanabban a mappában lévő fájlokat keresnénk az SGID bit beállításával, akkor futtassuk a

`find /usr/bin/ -perm -2000` vagy a `find /usr/bin/ -perm -g+s` parancsot.

Végül, a két speciális jogosultsággal rendelkező fájlok megtalálásához adjuk hozzá a `4`-et és a `2`-t, és használjuk a `/`-t:

```
carol@debian:~$ sudo find /usr/bin -perm /6000
/usr/bin/dotlock.mailutils
/usr/bin/umount
/usr/bin/newgrp
/usr/bin/wall
/usr/bin/ssh-agent
/usr/bin/chage
/usr/bin/dotlockfile
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/mount
/usr/bin/passwd
/usr/bin/chsh
/usr/bin/expiry
/usr/bin/sudo
/usr/bin/bsd-write
/usr/bin/crontab
/usr/bin/su
```

Jelszómenedzselés és öregedés (aging)

Mint fentebb említettük, normál felhasználóként a `passwd` segédprogrammal megváltoztathatjuk a saját jelszavunkat. Ezen kívül a `-S` vagy `--status` kapcsolót is használhatjuk, így státuszinformációkat kapva a fiókunkról:

```
carol@debian:~$ passwd -S
carol P 12/07/2019 0 99999 7 -1
```

Az alábbiakban a kimenet hét mezőjét mutatjuk be:

`carol`
: A felhasználó bejelentkezési neve.

`P`
: Azt jelzi, hogy a felhasználó érvényes jelszóval rendelkezik (`P`). Más lehetséges értékek: `L` a zárolt (locked) jelszó, `NP` pedig a jelszó hiánya (no password).

`12/07/2019`

Az utolsó jelszóváltoztatás dátuma.

`0`

Minimális életkor napokban (a jelszóváltoztatások közötti minimális napok száma). A `0` érték azt jelenti, hogy a jelszó bármikor megváltoztatható.

`99999`

Maximális életkor napokban (a jelszó maximális érvényességi ideje). A `99999` érték kikapcsolja a jelszó lejáratának lehetőségét.

`7`

Figyelmeztetési időszak napokban (a jelszó lejárta előtti napok száma, amikor a felhasználó figyelmeztetést kap).

`-1`

A jelszó inaktív időszaka napokban (a jelszó lejárta utáni inaktív napok száma a fiók zárolása előtt). A `-1` érték eltörli a fiók inaktivitását.

A fiókok állapotáról szóló jelentésen kívül a `passwd` parancsot root felhasználóként fogjuk használni fiókok alapvető karbantartására. A `-l`, `-u`, `-e` és `-d` kapcsolókkal zárolhatjuk és feloldhatjuk a fiókokat, kényszeríthetjük a felhasználót, hogy a következő bejelentkezéskor megváltoztassa jelszavát, illetve törölhetjük a felhasználó jelszavát.

Ezen opciók teszteléséhez célszerű ezen a ponton bevezetni a `su` parancsot. A `su` parancs segítségével a bejelentkezés során felhasználót válthatunk. Így például használjuk a `passwd`-t root-ként, hogy zároljuk `carol`jelszavát. Ezután váltsunk át `carol`-ra, és ellenőrizzük a fiókunk állapotát, hogy meggyőződjünk arról, hogy a jelszó — valóban — zárolva van (`L`), és nem lehet megváltoztatni. Végül, visszatérve a root felhasználóhoz, feloldjuk `carol` jelszavát:

```
root@debian:~# passwd -l carol
passwd: password expiry information changed.
root@debian:~# su - carol
carol@debian:~$ passwd -S
carol L 05/31/2020 0 99999 7 -1
carol@debian:~$ passwd
Changing password for carol.
Current password:
passwd: Authentication token manipulation error
passwd: password unchanged
carol@debian:~$ exit
```

```
logout
root@debian:~# passwd -u carol
passwd: password expiry information changed.
```

Alternatív megoldásként a `usermod` paranccsal is zárolhatjuk és feloldhatjuk a felhasználó jelszavát:

`carol` jelszavának zárolása

`usermod -L carol` vagy `usermod --lock carol`.

`carol` jelszavának feloldása

`usermod -U carol` vagy `usermod --unlock carol`.

NOTE Az `-f` vagy az `--inactive` kapcsolókkal a `usermod` arra is használható, hogy beállítsuk, hány nap múlva kerüljön letiltásra egy lejárt jelszóval rendelkező fiók (pl.: `usermod -f 3 carol`).

A `passwd` és a `usermod` mellett a legközvetlenebb parancs a jelszavak és fiókok öregedésével kapcsolatban a `chage` (“change age”). Root felhasználóként a `chage` parancsnak átadhatjuk az `-l` (vagy `--list`) kapcsolót, amelyet egy felhasználónév követ. Ezzel az adott felhasználó aktuális jelszavát és fiókjának lejárati adatait kiírja a képernyőre; normál felhasználóként pedig a saját adatainkat tekinthetjük meg:

```
carol@debian:~$ chage -l carol
Last password change                    : Aug 06, 2019
Password expires                        : never
Password inactive                       : never
Account expires                         : never
Minimum number of days between password change      : 0
Maximum number of days between password change      : 99999
Number of days of warning before password expires   : 7
```

Kapcsolók nélkül, egyszerűen csak egy felhasználónév megadásával a `chage` interaktívan fog viselkedni:

```
root@debian:~# chage carol
Changing the aging information for carol
Enter the new value, or press ENTER for the default

    Minimum Password Age [0]:
    Maximum Password Age [99999]:
```

```
    Last Password Change (YYYY-MM-DD) [2020-06-01]:
    Password Expiration Warning [7]:
    Password Inactive [-1]:
    Account Expiration Date (YYYY-MM-DD) [-1]:
```

A különböző `chage` beállítások módosításának lehetőségei a következők:

`-m days username` vagy `--mindays days username`

Adjuk meg a jelszóváltoztatások közötti minimális napok számát (pl.: `chage -m 5 carol`). A `0` érték lehetővé teszi a felhasználó számára, hogy bármikor megváltoztassa jelszavát.

`-M days username` vagy `--maxdays days username`

Adjuk meg a jelszó maximális érvényességi idejét (pl.: `chage -M 30 carol`). A jelszó lejárati idejének tiltása esetén ennek az opciónak `99999` értéket kell, hogy adjunk.

`-d days username` vagy `--lastday days username`

Adjuk meg a jelszó utolsó módosítása óta eltelt napok számát (pl.: `chage -d 10 carol`). A `0` érték arra kényszeríti a felhasználót, hogy a következő bejelentkezéskor megváltoztassa a jelszavát.

`-W days username` vagy `--warndays days username`

Adjuk meg, hogy a felhasználó hány napig kapjon emlékeztetőt a jelszava lejártáról.

`-I days username` vagy `--inactive days username`

Adjuk meg a jelszó lejárta utáni inaktív napok számát (pl.: `chage -I 10 carol`) — ez ugyanaz, mint a `usermod -f` vagy `usermod --inactive`. Ha a napok száma letelt, a fiók zárolásra kerül. A `0` értékkel azonban a fiók nem lesz zárolva.

`-E date username` vagy `--expiredate date username`

Adjuk meg azt a dátumot (vagy a napok számát az _epoch _ óta — 1970. január 1.), amikor a fiók zárolásra kerül. Ezt általában `YYYY-MM-DD` formátumban adjuk meg (pl.: `chage -E 2050-12-13 carol`).

NOTE A `passwd`, a `usermod` és a `chage` beállításairól — és kapcsolóikról — többet tudhatunk meg a megfelelő man oldalakon.

Nyitott portok felfedezése

Ha a nyitott portok ellenőrzéséről van szó, négy hatékony segédprogram érhető el a legtöbb Linux rendszeren: az `lsof`, `fuser`, `netstat` és `nmap`. Ebben a szakaszban ezekkel foglalkozunk.

Az `lsof` a “list open files” rövidítése, ami nem kis dolog, tekintve, hogy — Linux esetén — minden egy fájl. Valójában, ha beírjuk a terminálba az `lsof` parancsot, egy nagy listát kapunk a hagyományos fájlokról, eszközfájlokról, socketekről stb. A lecke kedvéért azonban elsősorban a portokra fogunk koncentrálni. Az összes “Internet” hálózati fájl listájának kiírásához futtassuk az `lsof` parancsot az `-i` kapcsolóval:

```
root@debian:~# lsof -i
COMMAND  PID     USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
dhclient 357     root    7u  IPv4  13493      0t0  UDP *:bootpc
sshd     389     root    3u  IPv4  13689      0t0  TCP *:ssh (LISTEN)
sshd     389     root    4u  IPv6  13700      0t0  TCP *:ssh (LISTEN)
apache2  399     root    3u  IPv6  13826      0t0  TCP *:http (LISTEN)
apache2  401 www-data    3u  IPv6  13826      0t0  TCP *:http (LISTEN)
apache2  402 www-data    3u  IPv6  13826      0t0  TCP *:http (LISTEN)
sshd     557     root    3u  IPv4  14701      0t0  TCP 192.168.1.7:ssh->192.168.1.4:60510
(ESTABLISHED)
sshd     569    carol    3u  IPv4  14701      0t0  TCP 192.168.1.7:ssh->192.168.1.4:60510
(ESTABLISHED)
```

A `bootpc` szolgáltatáson kívül — amit a DHCP használ — az eredmény két kapcsolatot váró szolgáltatást mutat — az `ssh` és az Apache webszerver (`http`) — valamint két létrehozott SSH kapcsolatot. Megadhatunk egy adott hostot a `@ip-address` jelöléssel, hogy ellenőrizzük a kapcsolatait:

```
root@debian:~# lsof -i@192.168.1.7
COMMAND PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sshd    557  root    3u  IPv4  14701      0t0  TCP 192.168.1.7:ssh->192.168.1.4:60510
(ESTABLISHED)
sshd    569 carol    3u  IPv4  14701      0t0  TCP 192.168.1.7:ssh->192.168.1.4:60510
(ESTABLISHED)
```

NOTE Ha csak az IPv4 és IPv6 hálózati fájlokat szeretnénk látni, használjuk az `-i4` és az `-i6` kapcsolót.

Hasonlóképpen szűrhetünk portok szerint is, ha az `-i` (vagy `-i@ip-cím`) kapcsolót adjuk meg a `:port` argumentumnak:

```
root@debian:~# lsof -i :22
COMMAND PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sshd    389  root    3u  IPv4  13689      0t0  TCP *:ssh (LISTEN)
sshd    389  root    4u  IPv6  13700      0t0  TCP *:ssh (LISTEN)
```

```
sshd    557  root    3u  IPv4  14701      0t0  TCP 192.168.1.7:ssh->192.168.1.4:60510
(ESTABLISHED)
sshd    569 carol    3u  IPv4  14701      0t0  TCP 192.168.1.7:ssh->192.168.1.4:60510
(ESTABLISHED)
```

Több portot vesszővel kell elválasztani (a tartományokat pedig kötőjellel):

```
root@debian:~# lsof -i@192.168.1.7:22,80
COMMAND PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sshd    705  root    3u  IPv4  13960      0t0  TCP 192.168.1.7:ssh->192.168.1.4:44766
(ESTABLISHED)
sshd    718 carol    3u  IPv4  13960      0t0  TCP 192.168.1.7:ssh->192.168.1.4:44766
(ESTABLISHED)
```

NOTE Az `lsof` rendelkezésére álló lehetőségek száma igen lenyűgöző. Ha többet szeretnénk megtudni róluk, nézzük meg a man oldalt!

A hálózati parancsok listáján a következő a `fuser`. A fő célja a “file’s user” (fájl felhasználójának) a megtalálása — ami azt jelenti, hogy tudni kell, hogy mely folyamatok milyen fájlokhoz férnek hozzá; ezen kívül néhány egyéb információt is megad, többek között a hozzáférés típusát. Például az aktuális mappa ellenőrzéséhez elegendő az `fuser .` futtatása, azonban ha egy kicsit több információt szeretnénk kapni, akkor célszerű a verbose (`-v` vagy `--verbose`) kapcsolót használni:

```
root@debian:~# fuser .
/root:                 580c
root@debian:~# fuser -v .
                     USER        PID ACCESS COMMAND
/root:               root        580 ..c.. bash
```

Bontsuk fel a kimenetet:

File

A fájl, amiről információt kapunk (`/root`).

`USER` oszlop

A fájl tulajdonosa (`root`).

`PID` oszlop

A folyamat azonosítója (`580`).

`ACCESS` oszlop

A hozzáférés típusa (`...`). Az alábbiak:

`c`

Az aktuális mappa.

`e`

Futtatás alatt álló program.

`f`

Megnyitott fájl (az alapértelmezett megjelenítési módban kimarad).

`F`

Írásra megnyitott fájl (az alapértelmezett megjelenítési módban kimarad).

`r`

root mappa.

`m`

mmap'ed fájl vagy megosztott könyvtár.

`.`

Placeholder (az alapértelmezett megjelenítési módban kimarad).

`COMMAND` oszlop

A fájlhoz kapcsolódó parancs (`bash`).

Az `-n` (vagy `--namespace`) kapcsolóval a hálózati portokról/socketekről kaphatunk információt. Meg kell adni a hálózati protokollt és a port számát is. Az Apache webszerverre vonatkozó információk lekérdezéséhez például a következő parancsot kell futtatni:

```
root@debian:~# fuser -vn tcp 80
                     USER        PID ACCESS COMMAND
80/tcp:              root        402 F.... apache2
                     www-data    404 F.... apache2
                     www-data    405 F.... apache2
```

NOTE

A `fuser` arra is használható, hogy a fájlhoz hozzáférő folyamatokat a `-k` vagy `--kill` kapcsolókkal terminálja (pl.: `fuser -k 80/tcp`). Részletesebb információkért keressük fel a man oldalát!

Térjünk át a `netstat`-ra. A `netstat` egy nagyon sokoldalú hálózati eszköz, amelyet leginkább "network statistics" (hálózati statisztikák) megjelenítésére használunk.

Kapcsolók nélkül futtatva a `netstat` mind az aktív internetkapcsolatokat, mind a Unix socketeket megjeleníti. A listázás mérete miatt érdemes a kimenetet a `less`-en keresztül megjeleníteni egy pipe használatával:

```
carol@debian:~$ netstat |less
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 192.168.1.7:ssh         192.168.1.4:55444       ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ]         DGRAM                    10509    /run/systemd/journal/syslog
unix  3      [ ]         DGRAM                    10123    /run/systemd/notify
(...)
```

Ha csak a "hallgató" portokat és socketeket akarjuk listázni, akkor használjuk az `-l` vagy a `--listening` kapcsolókat. A `-t`/`--tcp` és az `-u`/`--udp` kapcsolókkal a TCP és UDP protokollok szerint szűrhetünk (ezek kombinálhatók is ugyanabban a parancsban). Hasonlóképpen, az `-e`/`--extend` további információkat jelenít meg:

```
carol@debian:~$ netstat -lu
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 0.0.0.0:bootpc          0.0.0.0:*
carol@debian:~$ netstat -lt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
tcp        0      0 localhost:smtp          0.0.0.0:*               LISTEN
tcp6       0      0 [::]:http               [::]:*                  LISTEN
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
tcp6       0      0 localhost:smtp          [::]:*                  LISTEN
carol@debian:~$ netstat -lute
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       User
Inode
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN      root
13729
tcp        0      0 localhost:smtp          0.0.0.0:*               LISTEN      root
14372
```

```
tcp6       0      0 [::]:http               [::]:*                  LISTEN      root
14159
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN      root
13740
tcp6       0      0 localhost:smtp          [::]:*                  LISTEN      root
14374
udp        0      0 0.0.0.0:bootpc          0.0.0.0:*                           root
13604
```

Ha elhagyjuk az `l` kapcsolót, akkor *csak* a már létrehozott kapcsolatok jelennek meg:

```
carol@debian:~$ netstat -ute
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       User
Inode
tcp        0      0 192.168.1.7:ssh         192.168.1.4:39144       ESTABLISHED root
15103
```

Ha csak a portokra és hostokra vonatkozó numerikus információkat akarjuk látni, akkor a parancs a `-n` vagy `--numeric` kapcsolóval csak a portok számát és az IP-címeket írja ki. Figyeljük meg, hogy az `ssh` a fenti parancshoz hozzáadva a `-n`-t `22`-re változik:

```
carol@debian:~$ netstat -uten
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       User
Inode
tcp        0      0 192.168.1.7:22          192.168.1.4:39144       ESTABLISHED 0
15103
```

Láthatjuk, hogy nagyon hasznos és produktív `netstat` parancsokat készíthetünk néhány opció kombinálásával. Nézzünk szét a man oldalakon, hogy többet megtudjunk, és megtaláljuk az igényeinknek leginkább megfelelő kombinációkat.

Végül bemutatjuk az `nmap`-ot, vagyis a “network mapper”-t. Ez egy másik nagyon hatékony segédprogram, ez a port scanner egy IP-cím vagy hostnév megadásával futtatható:

```
root@debian:~# nmap localhost
Starting Nmap 7.70 ( https://nmap.org ) at 2020-06-04 19:29 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000040s latency).
Other addresses for localhost (not scanned): ::1
```

```
Not shown: 998 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 1.58 seconds
```

Egyetlen hosttól eltekintve az `nmap` lehetővé teszi a szkennelést:

több host esetén

szóközökkel elválasztva (pl.: `nmap localhost 192.168.1.7`).

host tartományok esetén

kötőjellel elválasztva (pl.: `nmap 192.168.1.3-20`).

alhálózatok esetén

helyettesítő karakter vagy a CIDR-jelölés használatával (pl.: `nmap 192.168.1.*` vagy `nmap 192.168.1.0/24`). Ki is zárhatunk bizonyos hostokat (pl.: `nmap 192.168.1.0/24 --exclude 192.168.1.7`).

Egy adott port vizsgálatához használjuk a `-p` kapcsolót, majd a port számát vagy a szolgáltatás nevét (az `nmap -p 22` és az `nmap -p ssh` ugyanazt a kimenetet adja):

```
root@debian:~# nmap -p 22 localhost
Starting Nmap 7.70 ( https://nmap.org ) at 2020-06-04 19:54 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000024s latency).
Other addresses for localhost (not scanned): ::1

PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds
```

Több portot vagy porttartományt is vizsgálhatunk vesszők és kötőjelek használatával:

```
root@debian:~# nmap -p ssh,80 localhost
Starting Nmap 7.70 ( https://nmap.org ) at 2020-06-04 19:58 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000051s latency).
Other addresses for localhost (not scanned): ::1
```

```
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds
```

```
root@debian:~# nmap -p 22-80 localhost
Starting Nmap 7.70 ( https://nmap.org ) at 2020-06-04 19:58 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000011s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 57 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 1.47 seconds
```

Két másik fontos és hasznos `nmap` kapcsoló:

`-F`
: Gyors szkennelés végrehajtása a 100 leggyakoribb porton.

`-v`
: Részletes kimenet (a `-vv` még részletesebb kimenetet ad).

NOTE Az `nmap` a szkennelés típusainak felhasználásával meglehetősen összetett parancsokat is képes futtatni. Ez a téma azonban nem tartozik ennek a leckének a tárgykörébe.

A felhasználók bejelentkezésének, folyamatainak és memóriahasználatának korlátai

A Linux rendszer erőforrásai nem korlátlanok, így — rendszergazdaként — biztosítani kell a megfelelő egyensúlyt az erőforrások *felhasználói korlátozása* (user limits) és az operációs rendszer megfelelő működése között. Az `ulimit` segíthet ebben a feladatban.

Az `ulimit` a *lágy* (soft) és *kemény* (hard) limitekkel foglalkozik, amelyeket a `-S` és a `-H` kapcsolók határoznak meg. Kapcsolók és argumentumok nélkül futtatva az `ulimit` az aktuális felhasználó soft fájlblokkjait jeleníti meg:

```
carol@debian:~$ ulimit
unlimited
```

A `-a` kapcsolóval az `ulimit` az összes aktuális soft limitet megjeleníti (ugyanúgy, mint a `-Sa`); az összes aktuális hard limit megjelenítéséhez pedig használjuk a `-Ha` kapcsolót:

```
carol@debian:~$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
(...)
carol@debian:~$ ulimit -Ha
core file size          (blocks, -c) unlimited
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
(...)
```

A rendelkezésre álló shell-erőforrásokat olyan kapcsolókkal adhatjuk meg, mint például:

`-b`
: a socket maximális puffermérete

`-f`
: a shell és gyermekei által írt fájlok maximális mérete

`-l`
: a memóriába zárolható maximális méret

`-m`
: maximális rezidens beállított méret (RSS) — a memória aktuális része, amelyet egy folyamat a főmemóriában (RAM) tart

`-v`
: a virtuális memória maximális mérete

`-u`
: az egy felhasználó számára elérhető folyamatok maximális száma

Így a határértékek megjelenítéséhez az `ulimit` parancsot kell használni, amelyet vagy az `-S` (soft) vagy a `-H` (hard) és az erőforrás kapcsoló követ; ha sem az `-S`, sem a `-H` nem szerepel, akkor a soft

limitek jelennek meg:

```
carol@debian:~$ ulimit -u
10000
carol@debian:~$ ulimit -Su
10000
carol@debian:~$ ulimit -Hu
15672
```

Hasonlóképpen, ha egy adott erőforrásra új limiteket szeretnénk beállítani, akkor vagy az `-S` vagy a `-H` értéket adjuk meg, majd a megfelelő erőforrás kapcsolóját és az új értéket. Ez az érték lehet egy szám vagy a speciális szavak: `soft` (jelenlegi soft limit), `hard` (jelenlegi hard limit) vagy `unlimited` (nincs limit (no limit)). Ha sem az `-S`, sem a `-H` érték nincs megadva, akkor mindkét határérték be lesz állítva. Először is olvassuk ki például a shell és gyermekei által írt fájlok aktuális maximális méretének értékét:

```
root@debian:~# ulimit -Sf
unlimited
root@debian:~# ulimit -Hf
unlimited
```

Most pedig változtassuk meg az értéket `unlimited`-ről `500` blokkra anélkül, hogy megadnánk az `S` vagy a `H` értéket. Figyeljük meg, hogy mind a soft, mind a hard limitek megváltoznak:

```
root@debian:~# ulimit -f 500
root@debian:~# ulimit -Sf
500
root@debian:~# ulimit -Hf
500
```

Végül csak a soft limitet csökkentjük `200` blokkra:

```
root@debian:~# ulimit -Sf 200
root@debian:~# ulimit -Sf
200
root@debian:~# ulimit -Hf
500
```

A hard limiteket csak a root felhasználó növelheti. Másrészt a normál felhasználók csökkenthetik

a hard limiteket, és növelhetik a soft limiteket a hard limitek értékéig. Ahhoz, hogy az új határértékek az újraindítások során is megmaradjanak, be kell írni őket az `/etc/security/limits.conf` fájlba. Ez az a fájl, amelyet a rendszergazda is használ az egyes felhasználókra vonatkozó korlátozások alkalmazására.

NOTE Figyelmeztetünk, hogy nincs `ulimit` man oldal! Ez egy bash beépülő, így a bash man oldaláról kell tájékozódni róla!

A bejelentkezett felhasználók kezelése

Rendszergazdaként egy másik feladatunk a bejelentkezett felhasználók nyomon követése. Három segédprogram van, amely segíthet ezekben a feladatokban: `last`, `who` és `w`.

A `last` kiírja az utoljára bejelentkezett felhasználók listáját, tetején a legfrissebb információkkal:

```
root@debian:~# last
carol    pts/0        192.168.1.4      Sat Jun  6 14:25   still logged in
reboot   system boot  4.19.0-9-amd64   Sat Jun  6 14:24   still running
mimi     pts/0        192.168.1.4      Sat Jun  6 12:07 - 14:24  (02:16)
reboot   system boot  4.19.0-9-amd64   Sat Jun  6 12:07 - 14:24  (02:17)
(...)
wtmp begins Sun May 31 14:14:58 2020
```

A csonka listát figyelembe véve információt kapunk a rendszer két utolsó felhasználójáról. Az első két sor a `carol` felhasználóról, a következő két sor pedig a `mimi` felhasználóról szól. Az információ a következő:

1. A `carol` felhasználó a `pts/0` terminálban a `192.168.1.4` hostról elindította a munkamenetét `Sat Jun 6` (június 6-án szombaton) `14:25`-kor és még mindig `logged in` (be van jelentkezve). A rendszer — a kernel `4.19.0-9-amd64` használatával — `Sat Jun 6` (június 6-án szombaton) `14:24`-kor lett elindítva (`reboot system boot`) és `still running` (még mindig fut).
2. A `mimi` felhasználó a `pts/0` terminálban a `192.168.1.4` hostról elindította a munkamenetét `Sat Jun 6` (június 6-án szombaton) `12:07`-kor és kijelentezett `14:24`-kor (a munkamenet `(02:16)` órán át tartott). A rendszer — a kernel `4.19.0-9-amd64` használatával — `Sat Jun 6` (június 6-án szombaton) `12:07`-kor lett elindítva (`reboot system boot`) és `14:24`-kor kikapcsolva (`(02:17)`-en keresztül futott).

NOTE A `wtmp starts Sun May 31 14:14:58 2020` sor a `/var/log/wtmp`-re utal, amely az a speciális logfájl, amelyből a `last` az információkat nyeri.

Átadhatjuk a `last`-nak a felhasználónevet, így csak az adott felhasználóhoz tartozó bejegyzések

jelennek meg:

```
root@debian:~# last carol
carol    pts/0        192.168.1.4      Sat Jun  6 14:25   still logged in
carol    pts/0        192.168.1.4      Sat Jun  6 12:07 - 14:24  (02:16)
carol    pts/0        192.168.1.4      Fri Jun  5 00:48 - 01:28  (00:39)
(...)
```

Ami a második oszlopot (terminal) illeti, a `pts` a *Pseudo Terminal Slave*-t jelenti - szemben a valódi *TeleTYpewriter* vagy `tty` terminállal; a `0` pedig az elsőre utal (a számolás nulláról indul).

NOTE A rossz bejelentkezési kísérletek ellenőrzéséhez futtassuk a `lastb`-t a `last` helyett!

A `who` és `w` segédprogramok az aktuálisan bejelentkezett felhasználókra koncentrálnak és eléggé hasonlóak. Az előbbi azt mutatja meg, hogy ki van bejelentkezve, míg az utóbbi azt is, hogy éppen mit csinálnak.

Opciók nélküli futtatás esetén a `who` négy oszlopot jelenít meg a bejelentkezett felhasználó, a terminál, a dátum és az idő, valamint a hostnév szerint:

```
root@debian:~# who
carol    pts/0        2020-06-06 17:16 (192.168.1.4)
mimi     pts/1        2020-06-06 17:28 (192.168.1.4)
```

A `who` egy sor kapcsolót fogad el, amelyek közül kiemelhetjük a következőket:

`-b`,`--boot`
Az utolsó rendszerindítás idejének megjelenítése.

`-r`,`--runlevel`
Az aktuális runlevel megjelenítése.

`-H`,`--heading`
Az oszlopok fejlécének a megjelenítése.

A `who`-hoz hasonlítva a `w` egy kicsit részletesebb kimenetet ad:

```
root@debian:~# w
 17:56:12 up 40 min,  2 users,  load average: 0.04, 0.12, 0.09
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
carol    pts/0    192.168.1.4    17:16    1.00s  0.15s  0.05s sshd: carol [priv]
```

```
mimi     pts/1    192.168.1.4    17:28   15:08   0.05s  0.05s -bash
```

A felső sorban az aktuális időpontról (`17:56:12`), arról, hogy mióta működik a rendszer (`up 40 min`), a jelenleg bejelentkezett felhasználók számáról (`2 users`) és a terhelés átlagáról (`load average: 0.04, 0.12, 0.09`) kapunk információt. Ezek az értékek a futtatási sorban lévő jobok számát jelentik az elmúlt 1, 5, illetve 15 perc átlagában.

Ezután nyolc oszlopot találunk; nézzük meg őket lebontva:

`USER`
: A felhasználó bejelentkezési neve.

`TTY`
: A felhasználó által használt terminál neve.

`FROM`
: A remote host, ahonnan a felhasználó bejelentkezett.

`LOGIN@`
: A bejelentkezés ideje.

`IDLE`
: Holtidő (idle time).

`JCPU`
: A tty-hez csatlakozó összes folyamat által felhasznált idő (beleértve a jelenleg háttérben futó jobokat is).

`PCPU`
: Az aktuális folyamat által felhasznált idő (ez a folyamat az, ami a `WHAT` alatt látható).

`WHAT`
: Az aktuális folyamat parancssora.

Akárcsak a `who` esetén, a `w`-nek is adhatunk át felhasználóneveket:

```
root@debian:~# w mimi
 18:23:15 up  1:07,  2 users,  load average: 0.00, 0.02, 0.05
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
mimi     pts/1    192.168.1.4      17:28    9:23   0.06s  0.06s -bash
```

Alapvető `sudo` konfiguráció és használat

Amint azt már említettük ebben a leckében, a `su` segítségével a rendszer bármely más felhasználójára válthatunk, ha megadjuk a célfelhasználó jelszavát. Ha a root felhasználó jelszavát terjesztik vagy sok felhasználó ismeri, az veszélyezteti a rendszert, ezáltal pedig egy nagyon rossz biztonsági gyakorlat. A `su` alapvető használata a `su - célfelhasználónév`. A root-ra való váltásnál a célfelhasználónév megadása opcionális:

```
carol@debian:~$ su - root
Password:
root@debian:~# exit
logout
carol@debian:~$ su -
Password:
root@debian:~#
```

A kötőjel (`-`) használata biztosítja, hogy a célfelhasználó környezete betöltődjön. Enélkül a régi felhasználó környezete marad meg:

```
carol@debian:~$ su
Password:
root@debian:/home/carol#
```

A másik lehetőség a `sudo` parancs használata. Ezzel a parancsot root felhasználóként — vagy bármilyen más felhasználóként — is végre tudjuk hajtani. Biztonsági szempontból a `sudo` sokkal jobb lehetőség, mint a `su`, mivel két fő előnye van:

1. egy parancs root felhasználóként történő futtatásához nincs szükség a root felhasználó jelszavára, hanem csak a parancsot futtató felhasználó jelszavára, a biztonsági szabályzatnak megfelelően. Az alapértelmezett biztonsági szabályzat az `/etc/sudoers` és az `/etc/sudoers.d/*` állományokban megadott `sudoers`.
2. a `sudo` lehetővé teszi egyes parancsok futtatását emelt jogosultságokkal, ahelyett, hogy egy teljesen új subshellt kellene indítanunk a root számára, mint a `su` esetén.

A `sudo` alapvető használata a `sudo -u célfelhasználónév parancs`. Ha azonban egy parancsot root felhasználóként szeretnénk futtatni, az `-u célfelhasználónév` kapcsolóra nincs szükség:

```
carol@debian:~$ sudo -u mimi whoami
mimi
```

```
carol@debian:~$ sudo whoami
root
```

NOTE

A `sudoers` egy felhasználónkénti (és terminálonkénti) időbélyeget használ a hitelesítő adatok gyorsítótárazására, így a `sudo`-t jelszó nélkül használhatjuk egy alapértelmezett tizenöt perces időtartamig. Ez az alapértelmezett érték módosítható a `timestamp_timeout` opció `Defaults` beállításként történő hozzáadásával az `/etc/sudoers` állományban (pl.: a `Defaults timestamp_timeout=1` egy percre állítja be a hitelesítő adatok gyorsítótárazásának időkorlátját).

Az `/etc/sudoers` fájl

A `sudo` fő konfigurációs fájlja az `/etc/sudoers` (van még egy `/etc/sudoers.d` mappa is). Ez az a hely, ahol a felhasználók `sudo` jogosultságait meghatározzák. Más szóval itt adható meg, hogy ki milyen parancsokat futtathat milyen felhasználóként, milyen gépeken — és más egyéb beállítások is. A használt szintaxis a következő:

```
carol@debian:~$ sudo less /etc/sudoers
(...)
# User privilege specification
root    ALL=(ALL:ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
(...)
```

A root felhasználó jogosultsági specifikációja az `ALL=(ALL:ALL) ALL`. Ez azt jelenti, hogy a root felhasználó (`root`) minden hostról bejelentkezhet (`ALL`), minden felhasználóként és minden csoportként (`(ALL:ALL)`), és minden parancsot futtathat (`ALL`). Ugyanez igaz a `sudo` csoport tagjaira is — figyeljük meg, hogy a csoportneveket egy százalékjel (%) jelöli.

Így ahhoz, hogy a `carol` felhasználó képes legyen ellenőrizni az `apache2` állapotát bármely hostról, bármelyik felhasználó vagy csoport nevében, a következő sort kell hozzáadni a `sudoers` fájlhoz:

```
carol   ALL=(ALL:ALL) /usr/bin/systemctl status apache2
```

Megkímélhetjük `carol`-t attól a kellemetlenségtől, hogy meg kelljen adnia a jelszavát a `systemctl status apache2` parancs futtatásához. Ehhez így kell módosítanunk a sort:

```
carol   ALL=(ALL:ALL) NOPASSWD: /usr/bin/systemctl status apache2
```

Tegyük fel, hogy most a 192.168.1.7-re akarjuk korlátozni a hostokat és engedélyezzük `carolnak`, hogy `mimi` felhasználóként futtassa a `systemctl status apache2`-t. A sort a következőképpen módosíthatjuk:

```
carol   192.168.1.7=(mimi) /usr/bin/systemctl status apache2
```

Most már ellenőrizhetjük az Apache webszerver állapotát `mimi` felhasználóként:

```
carol@debian:~$ sudo -u mimi systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2020-06-09 13:12:19 CEST; 29min ago
(...)
```

Ha `carol`-t előléptetnénk sysadminnak, és minden jogosultságot meg akarnánk adni neki, a legegyszerűbb megoldás az lenne, ha a `usermod` és a `-G` kapcsolóval felvennénk őt a speciális `sudo` csoportba (érdemes lehet használni az `-a` kapcsolót is, ami biztosítja, hogy a felhasználót nem távolítjuk el a többi csoportból, amelyhez esetleg tartozik):

```
root@debian:~# sudo useradd -aG sudo carol
```

NOTE A Red Hat disztribúciók családjában a `wheel` csoport a Debian rendszerek speciális adminisztrációs `sudo` csoportjának megfelelője.

Ahelyett, hogy közvetlenül szerkesztenénk az `/etc/sudoers` állományt, egyszerűen használjuk a `visudo` parancsot root felhasználóként (pl.: `visudo`), ami megnyitja az `/etc/sudoers` állományt az előre definiált szövegszerkesztővel. Az alapértelmezett szövegszerkesztő megváltoztatásához az `/etc/sudoers` állományban az `editor` opciót a `Defaults` résznél adhatjuk meg. Például, ha a szerkesztőt `nano`-ra akarjuk változtatni, akkor a következő sort kell hozzáadni:

```
Defaults    editor=/usr/bin/nano
```

NOTE Alternatívaként a `visudo` használatakor a `EDITOR` környezeti változóval is megadhatunk egy szövegszerkesztőt (pl.: `EDITOR=/usr/bin/nano visudo`).

A felhasználók és csoportok mellett használhatunk aliasokat is az `/etc/sudoers` állományban. Az

aliasoknak három fő definiálható kategóriája van: *host aliasok* (`Host_Alias`), *felhasználó aliasok* (`User_Alias`) és *parancs aliasok* (`Cmnd_Alias`). Íme egy példa:

```
# Host alias specification

Host_Alias SERVERS = 192.168.1.7, server1, server2

# User alias specification

User_Alias REGULAR_USERS = john, mary, alex

User_Alias PRIVILEGED_USERS = mimi

User_Alias ADMINS = carol, %sudo, PRIVILEGED_USERS, !REGULAR_USERS

# Cmnd alias specification

Cmnd_Alias SERVICES = /usr/bin/systemctl *

# User privilege specification
root    ALL=(ALL:ALL) ALL
ADMINS  SERVERS=SERVICES

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

Ezt a `sudoers` mintafájlt vizsgálva, elmagyarázzuk egy kicsit részletesebben az aliasok három típusát:

Host aliasok

Ezek közé tartozik a hostnevek, IP-címek, valamint hálózatok és hálózati csoportok vesszővel elválasztott listája (a `+` előtaggal). A hálózati maszkokat is meg lehet adni. A `SERVERS` host alias egy IP-címet és két hostnevet tartalmaz::

```
Host_Alias SERVERS = 192.168.1.7, server1, server2
```

Felhsználó aliasok

Ezek a felhasználók vesszővel elválasztott listáját tartalmazzák, amelyeket felhasználónévként, csoportokként (a `%` előtaggal) és hálózati csoportokként (a `+` előtaggal) adunk meg. Bizonyos felhasználókat a `!`-el lehet kizárni. Az `ADMINS` felhasználói alias — például — tartalmazza a `carol` felhasználót, a `sudo` csoport tagjait és a `PRIVILEGE_USERS` felhasználói alias azon

tagjait, amelyek nem tartoznak a `REGULAR_USERS` felhasználói aliasba:

```
User_Alias ADMINS = carol, %sudo, PRIVILEGED_USERS, !REGULAR_USERS
```

Parancs aliasok

A parancsok és mappák vesszővel elválasztott listáját tartalmazzák. Ha egy mappát adunk meg, akkor az abban a mappában lévő bármely fájl bekerül a listába — az almappákat azonban nem veszi figyelembe. A `SERVICES` parancs alias egyetlen parancsot tartalmaz az összes alparancsával együtt — ahogyan azt a csillag (`*`) jelöli:

```
Cmnd_Alias SERVICES = /usr/bin/systemctl *
```

Az alias specifikációk eredményeképpen a `User privilege specification` szakasz `ADMINS SERVERS=SERVICES` sora a következőképpen fordítható le: minden felhasználó, aki az `ADMINS`-ba tartozik, használhatja a `sudo`-t a `SERVICES`-ben szereplő bármely parancs futtatására a `SERVERS` -ben szereplő bármely szerveren.

NOTE

Van egy negyedik típusú alias is, amelyet az `/etc/sudoers` állományban adhatunk meg: *runas aliasok* (`Runas_Alias`). Ezek nagyon hasonlítanak a felhasználói aliasokhoz, de lehetővé teszik, hogy a felhasználókat a *felhasználói azonosítójuk* (UID) alapján adjuk meg. Ez a funkció bizonyos esetekben kényelmes lehet.

Gyakorló feladatok

1. Töltsük ki a következő táblázatot a speciális jogosultságokkal kapcsolatban:

Speciális jogosultság	**Numerikus ábrázolás**	**Szimbolikus ábrázolás**	**Olyan fájlok keresése, amelyeknél *csak* ez a jogosultság van megadva**
`SUID`			
`SGID`			

2. Az olyan fájlok megjelenítése, amelyekben *csak* az `SUID` vagy `SGID` bit van beállítva, általában nem túl praktikus. Végezzük el a következő feladatokat, hogy megbizonyosodjunk arról, hogy a keresések eredményesebbek lehetnek:
   - Az `/usr/bin` állományban beállított `SUID` (és egyéb jogosultságokkal) rendelkező összes fájl megkeresése:
   - Az `/usr/bin` állományban beállított `SGID` (és egyéb jogosultságokkal) rendelkező összes fájl megkeresése:
   - Az `/usr/bin` állományban beállított `SUID` vagy `SGID` jogosultságokkal rendelkező összes fájl megkeresése:

3. A `chage` segítségével megváltoztathatjuk a felhasználó jelszavának lejárati idejét. Root felhasználóként töltsük ki az alábbi táblázatot a `mary` felhasználó megfelelő parancsainak megadásával:

Jelentés	`chage` **parancsok**
Legyen a jelszó 365 napig érvényes!	
A felhasználó a következő bejelentkezéskor változtassa meg jelszavát!	
Legyen a jelszóváltoztatások közötti minimális napok száma 1!	
Jelszó lejárati idő letiltása.	

Jelentés	`chage` **parancsok**
Engedélyezzük, hogy a felhasználó bármikor megváltoztathassa jelszavát!	
A figyelmeztető időszakot 7 napra, a fiók lejárati dátumát pedig 2050. augusztus 20-ra állítsuk be!	
Írassuk ki a felhasználó aktuális jelszavának lejárati adatait!	

4. Töltse ki a következő táblázatot a megfelelő hálózati segédprogrammal:

Művelet	**Parancs(ok)**
A `192.168.1.55` host hálózati fájljainak megjelenítése a `22` porton az `lsof` használatával.	
Az Apache webszerver alapértelmezett portját elérő folyamatok megjelenítése a `fuser` segítségével.	
A gépen lévő összes hallgatózó *udp* socket listázása a `netstat` segítségével.	
A `192.168.1.55` hoston a `80` és `443` közötti portok szkennelése az `nmap` segítségével.	

5. Végezzük el a következő *resident set size* (*RSS*) és `ulimit` feladatokat egyszerű felhasználóként:
   - A *maximum RSS soft* limitjének megjelenítése:
   - A *maximum RSS hard* limitjének megjelenítése:
   - Állítsuk be a *maximum RSS soft* limitjét 5,000 kilobájtra:
   - Állítsuk be a *maximum RSS hard* limitjét 10,000 kilobájtra:
   - Végül, próbáljuk meg a *maximum RSS hard* limitjét 15,000 kilobájtra emelni. Meg tudjuk

tenni? Miért?

6. Tekintsük meg az alábbi `last` parancssort, és válaszoljunk a kérdésekre:

```
carol    pts/0        192.168.1.4      Sun May 31 14:16 - 14:22  (00:06)
```

   - `carol` egy távoli hostról csatlakozott? Miért?
   - Mennyi ideig tartott `carol` munkamenete?
   - `carol` egy igazi klasszikus szöveges terminálon keresztül csatlakozott? Miért?

7. Tekintsük át a következő részletet az `/etc/sudoers` állományból, és válaszoljunk az alábbi kérdésre!

```
# Host alias specification

Host_Alias SERVERS = 192.168.1.7, server1, server2

# User alias specification

User_Alias REGULAR_USERS = john, mary, alex

User_Alias PRIVILEGED_USERS = mimi

User_Alias ADMINS = carol, %sudo, PRIVILEGED_USERS, !REGULAR_USERS

# Cmnd alias specification

Cmnd_Alias WEB_SERVER_STATUS = /usr/bin/systemctl status apache2

# User privilege specification
root    ALL=(ALL:ALL) ALL
ADMINS  SERVERS=WEB_SERVER_STATUS

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

Tudja-e `alex` ellenőrizni az Apache webszerver állapotát bármelyik gépen? Miért?

Gondolkodtató feladatok

1. Az `SUID` és `SGID` mellett van egy harmadik speciális jogosultság is: a *sticky bit*. Jelenleg leginkább az olyan mappáknál használják, mint a `/tmp`, hogy megakadályozzák, hogy a rendszeres felhasználók a sajátjukon kívül más fájlokat is töröljenek vagy áthelyezzenek. Végezzük el a következő feladatokat:
   - Állítsuk be a *sticky bitet* a `~/temporal` mappára:
   - Keressünk olyan mappákat, amelyeken a *sticky bit* (és bármilyen más jogosultság van beállítva) a saját home mappánkban:
   - Vonjuk vissza a *sticky bitet* a `~/temporal` mappáról:
2. Ha egy felhasználó jelszava a `passwd -l username` vagy a `usermod -L username` alapján zárolt, hogyan lehet ezt megállapítani az `/etc/shadow` fájlból?
3. Mi a `usermod` parancs megfelelője a `chage -E date username` vagy a `chage --expiredate date username` parancsnak?
4. Adjunk meg két különböző `nmap` parancsot a localhost mind a 65535 portjának vizsgálatához:

Összefoglalás

Ebben a leckében megtanultuk, hogyan végezhetünk el számos biztonsági felügyeleti feladatot. A következő témakörökkel foglalkoztunk:

- A speciális `SUID` és `SGID` engedélyekkel rendelkező fájlok keresése.
- Felhasználók jelszavainak beállítása és módosítása, valamint a jelszó-öregedési információk kezelése.
- Számos hálózati segédprogram használata a hostok/hálózatok nyitott portjainak felderítésére.
- A rendszer erőforrásainak korlátozása.
- A rendszerbe korábban bejelentkezett vagy éppen bejelentkezett felhasználók ellenőrzése.
- A `sudo` alapvető használata és konfigurálása (az `/etc/sudoers` fájlon keresztül).

A leckében használt fájlok és parancsok:

`find`
: Fájlok keresése egy mappahierarchiában.

`passwd`
: Felhasználói jelszó módosítása.

`chmod`
: Fájl üzemmód bitek módosítása.

`chage`
: A felhasználói jelszó lejárati információinak módosítása.

`lsof`
: Megnyitott fájlok listázása.

`fuser`
: Fájlokat vagy socketeket használó folyamatok azonosítása.

`netstat`
: Hálózati kapcsolatok kiíratása.

`nmap`
: Hálózatfeltáró eszköz és portolvasó.

`ulimit`

Felhasználói limitek lekérdezése és beállítása.

`/etc/security/limits.conf`

Konfigurációs fájl a felhasználókra vonatkozó korlátozások alkalmazásához.

`last`

Az utoljára bejelentkezett felhasználók listájának kiíratása.

`lastb`

A rossz bejelentkezési kísérletek listájának kiíratása.

`/var/log/wtmp`

A felhasználói bejelentkezések adatbázisa.

`who`

Megmutatja, hogy ki van bejelentkezve.

`w`

Megmutatja, hogy ki van bejelentkezve és mit csinál.

`su`

Felhasználóváltás vagy szuperfelhasználóvá válás.

`sudo`

Egy parancs végrehajtása más felhasználóként (beleértve a szuperfelhasználót is).

`/etc/sudoers`

A `sudo` biztonsági szabályzat alapértelmezett konfigurációs fájlja.

Válaszok a gyakorló feladatokra

1. Töltsük ki a következő táblázatot a speciális jogosultságokkal kapcsolatban:

Speciális jogosultság	**Numerikus ábrázolás**	**Szimbolikus ábrázolás**	**Olyan fájlok keresése, amelyeknél *csak* ez a jogosultság van megadva**
`SUID`	`4000`	`s,S`	`find -perm 4000`, `find -perm u+s`
`SGID`	`2000`	`s,S`	`find -perm 2000`, `find -perm g+s`

2. Az olyan fájlok megjelenítése, amelyekben *csak* az `SUID` vagy `SGID` bit van beállítva, általában nem túl praktikus. Végezzük el a következő feladatokat, hogy megbizonyosodjunk arról, hogy a keresések eredményesebbek lehetnek:
   - Az `/usr/bin` állományban beállított `SUID` (és egyéb jogosultságokkal) rendelkező összes fájl megkeresése:

     `find /usr/bin -perm -4000` vagy `find /usr/bin -perm -u+s`
   - Az `/usr/bin` állományban beállított `SGID` (és egyéb jogosultságokkal) rendelkező összes fájl megkeresése:

     `find /usr/bin -perm -2000` vagy `find /usr/bin -perm -g+s`
   - Az `/usr/bin` állományban beállított `SUID` vagy `SGID` jogosultságokkal rendelkező összes fájl megkeresése:

     `find /usr/bin -perm /6000`

A `chage` segítségével megváltoztathatjuk a felhasználó jelszavának lejárati idejét. Root felhasználóként töltsük ki az alábbi táblázatot a `mary` felhasználó megfelelő parancsainak megadásával:

+

Jelentés	`chage` **parancsok**
Legyen a jelszó 365 napig érvényes!	`chage -M 365 mary`, `chage --maxdays 365 mary`

Jelentés	`chage` **parancsok**
A felhasználó a következő bejelentkezéskor változtassa meg jelszavát!	`chage -d 0 mary`, `chage --lastday 0 mary`
Legyen a jelszóváltoztatások közötti minimális napok száma 1!	`chage -m 1 mary`, `chage --mindays 1 mary`
Jelszó lejárati idő letiltása.	`chage -M 99999 mary`, `chage --maxdays 99999 mary`
Engedélyezzük, hogy a felhasználó bármikor megváltoztathassa jelszavát!	`chage -m 0 mary`, `chage --mindays 0 mary`
A figyelmeztető időszakot 7 napra, a fiók lejárati dátumát pedig 2050. augusztus 20-ra állítsuk be!	`chage -W 7 -E 2050-08-20 mary`, `chage --warndays 7 --expiredate 2050-08-20 mary`
Írassuk ki a felhasználó aktuális jelszavának lejárati adatait!	`chage -l mary`, `chage --list mary`

1. Töltse ki a következő táblázatot a megfelelő hálózati segédprogrammal:

Művelet	**Parancs(ok)**
A `192.168.1.55` host hálózati fájljainak megjelenítése a `22` porton az `lsof` használatával.	`lsof -i@192.168.1.55:22`
Az Apache webszerver alapértelmezett portját elérő folyamatok megjelenítése a `fuser` segítségével.	`fuser -vn tcp 80`, `fuser --verbose --namespace tcp 80`
A gépen lévő összes hallgatózó *udp* socket listázása a `netstat` segítségével.	`netstat -lu`,`netstat --listening --udp`
A `192.168.1.55` hoston a `80` és `443` közötti portok szkennelése az `nmap` segítségével.	`nmap -p 80-443 192.168.1.55`

2. Végezzük el a következő *resident set size* (*RSS*) és `ulimit` feladatokat egyszerű felhasználóként:
   - A *maximum RSS soft* limitjének megjelenítése:

     `ulimit -m`, `ulimit -Sm`
   - A *maximum RSS hard* limitjének megjelenítése:

     `ulimit -Hm`

- Állítsuk be a *maximum RSS soft* limitjét 5,000 kilobájtra:

  `ulimit -Sm 5000`

- Állítsuk be a *maximum RSS hard* limitjét 10,000 kilobájtra:

  `ulimit -Hm 10000`

- Végül, próbáljuk meg a *maximum RSS hard* limitjét 15,000 kilobájtra emelni. Meg tudjuk tenni? Miért?

  Nem. Ha egyszer beállításra került, az átlagos felhasználók nem növelhetik a hard limiteket.

3. Tekintsük meg az alábbi `last` parancssort, és válaszoljunk a kérdésekre:

```
carol    pts/0        192.168.1.4      Sun May 31 14:16 - 14:22  (00:06)
```

   - `carol` egy távoli hostról csatlakozott? Miért?

     Igen, a távoli host IP-címe a harmadik oszlopban található.

   - Mennyi ideig tartott `carol` munkamenete?

     Hat percig (az utolsó oszlopban látható).

   - `carol` egy igazi klasszikus szöveges terminálon keresztül csatlakozott? Miért?

     Nem, a `pts/0` a második oszlopban azt jelzi, hogy a kapcsolat egy grafikus terminál emulátoron keresztül jött létre (más néven *Pseudo Terminal Slave*).

4. Tekintsük át a következő részletet az `/etc/sudoers` állományból, és válaszoljunk az alábbi kérdésre!

```
# Host alias specification

Host_Alias SERVERS = 192.168.1.7, server1, server2

# User alias specification

User_Alias REGULAR_USERS = john, mary, alex

User_Alias PRIVILEGED_USERS = mimi
```

```
User_Alias ADMINS = carol, %sudo, PRIVILEGED_USERS, !REGULAR_USERS

# Cmnd alias specification

Cmnd_Alias WEB_SERVER_STATUS = /usr/bin/systemctl status apache2

# User privilege specification
root    ALL=(ALL:ALL) ALL
ADMINS  SERVERS=WEB_SERVER_STATUS

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

Tudja-e `alex` ellenőrizni az Apache webszerver állapotát bármelyik gépen? Miért?

Nem, mivel a `REGULAR_USERS` tagja és annak a csoportnak a felhasználói ki vannak zárva az `ADMINS`-ból, akik az egyetlen felhasználók (`carol`, a `sudo` csoport tagjai és a `root` kivételével), akik futtathatják a `systemctl status apache2`t a `SERVERS`-en.

Válaszok a gondolkodtató feladatokra

1. Az `SUID` és `SGID` mellett van egy harmadik speciális jogosultság is: a *sticky bit*. Jelenleg leginkább az olyan mappáknál használják, mint a `/tmp`, hogy megakadályozzák, hogy a rendszeres felhasználók a sajátjukon kívül más fájlokat is töröljenek vagy áthelyezzenek. Végezzük el a következő feladatokat:
   - Állítsuk be a *sticky bitet* a `~/temporal` mappára:

     `chmod +t temporal`, `chmod 1755 temporal`
   - Keressünk olyan mappákat, amelyeken a *sticky bit* (és bármilyen más jogosultság van beállítva) a saját home mappánkban:

     `find ~ -perm -1000`, `find ~ -perm /1000`
   - Vonjuk vissza a *sticky bitet* a `~/temporal` mappáról:

     `chmod -t temporal`, `chmod 0755 temporal`
2. Ha egy felhasználó jelszava a `passwd -l username` vagy a `usermod -L username` alapján zárolt, hogyan lehet ezt megállapítani az `/etc/shadow` fájlból?

   A második mezőben egy felkiáltójel jelenik meg, közvetlenül az érintett felhasználó bejelentkezési neve után (pl.: `mary:!$6$gOg9xJgv...`).
3. Mi a `usermod` parancs megfelelője a `chage -E date username` vagy a `chage --expiredate date username` parancsnak?

   `usermod -e date username`, `usermod --expiredate date username`
4. Adjunk meg két különböző `nmap` parancsot a localhost mind a 65535 portjának vizsgálatához:

   `nmap -p 1-65535 localhost` és `nmap -p- localhost`

110.2 Host-biztonság beállítása

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 102, Objective 110.2

Súlyozás

3

Kulcsfontosságú ismeretek

- A shadow jelszavak és működésük ismerete
- A használaton kívüli hálózati szolgáltatások kikapcsolása
- A TCP wrapperek szerepének megértése

A használt fájlok, kifejezések és segédprogramok listája

- `/etc/nologin`
- `/etc/passwd`
- `/etc/shadow`
- `/etc/xinetd.d/`
- `/etc/xinetd.conf`
- `systemd.socket`
- `/etc/inittab`
- `/etc/init.d/`
- `/etc/hosts.allow`
- `/etc/hosts.deny`

Linux Professional Institute

110.2 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	110 Biztonság
Fejezet:	110.2 Host-biztonság beállítása
Lecke:	1/1

Bevezetés

Ez a fejezet a host-biztonság javításának négy alapvető módját fejti ki:

1. Néhány alapvető parancs és konfigurációs beállítás a shadow jelszavakkal történő hitelesítés biztonságának javítására.
2. Hogyan használjuk a superdaemonokat a bejövő hálózati kapcsolatok figyelésére.
3. A hálózati szolgáltatások ellenőrzése a felesleges daemonok szempontjából.
4. TCP wrapperek, mint egyfajta egyszerű tűzfal.

A hitelesítési biztonság javítása shadow jelszavakkal

A felhasználói fiókadatok alapvető összetevői az `/etc/passwd` fájlban vannak tárolva. Ez a fájl hét mezőt tartalmaz: bejelentkezési név, userid, groupid, jelszó, megjegyzés (más néven GECOS), home mappa helye és végül az alapértelmezett shell. E mezők sorrendjét egyszerűen úgy lehet megjegyezni, ha a felhasználó bejelentkezésének folyamatára gondolunk: először megadjuk a bejelentkezési nevet, másodszor a rendszer ezt egy userid-re (uid), harmadszor pedig egy groupid-re (gid) képezi le. A negyedik lépés jelszót kér, az ötödik megnézi a megjegyzést, a hatodik megadja

a felhasználó home könyvtárát, a hetedik lépés pedig beállítja az alapértelmezett shellt.

A modern rendszerekben a jelszót már nem az `/etc/passwd` fájlban tárolják. Ehelyett a jelszó mezőben csak egy kisbetűs `x` szerepel. Az `/etc/passwd` fájlnak minden felhasználó számára olvashatónak kell lennie, ezért nem jó ötlet a jelszavakat ott tárolni. Az `x` azt jelzi, hogy a titkosított (hashed) jelszó valójában az `/etc/shadow` fájlban van tárolva. Ez a fájl nem lehet minden felhasználó számára olvasható.

A jelszóbeállítások a `passwd` és a `chage` parancsokkal konfigurálhatók. Mindkét parancs megváltoztatja az `emma` felhasználóhoz tartozó bejegyzést az `/etc/shadow` fájlban. Szuperfelhasználóként a következő paranccsal állíthatjuk be az `emma` felhasználó jelszavát:

```
$ sudo passwd emma
New password:
Retype new password:
passwd: password updated successfully
```

Ezután a rendszer kétszer kéri az új jelszó megerősítését.

Az `emma` felhasználó jelszó lejárati idejének és egyéb jelszó lejárati beállításainak listázásához használjuk az alábbit:

```
$ sudo chage -l emma
Last password change                    : Apr 27, 2020
Password expires                        : never
Password inactive                       : never
Account expires                         : never
Minimum number of days between password change      : 0
Maximum number of days between password change      : 99999
Number of days of warning before password expires   : 7
```

Annak érdekében, hogy az `emma` felhasználó ne tudjon bejelentkezni a rendszerbe, a szuperfelhasználó beállíthat egy olyan jelszó lejárati dátumot, amely megelőzi az aktuális dátumot. Ha például a mai dátum 2020-03-27 lenne, akkor egy régebbi dátummal lejárhatna a felhasználó jelszava:

```
$ sudo chage -E 2020-03-26 emma
```

A szuperfelhasználó használhat más, alternatív megoldást is:

```
$ sudo passwd -l emma
```

a fiók ideiglenes zárolása a `passwd l` kapcsolójával. A változtatások hatásának teszteléséhez próbáljunk meg bejelentkezni az `emma` fiókkal:

```
$ sudo login emma
Password:
Your account has expired; please contact your system administrator

Authentication failure
```

Ahhoz, hogy a root felhasználó kivételével egy felhasználó se tudjon ideiglenesen bejelentkezni a rendszerbe, a szuperfelhasználó létrehozhat egy `/etc/nologin` nevű fájlt. Ez a fájl tartalmazhat egy üzenetet a felhasználóknak, amelyben értesítik őket arról, hogy miért nem tudnak bejelentkezni (például értesítés rendszerkarbantartásról). A részletekért lsd. a `man 5 nologin` című részt! Megjegyzendő, hogy létezik egy `nologin` parancs is, amely a bejelentkezés megakadályozására használható, ha egy felhasználó alapértelmezett shelljeként van beállítva. Például:

```
$ sudo usermod -s /sbin/nologin emma
```

A `man 8 nologin` segítségével még több részletet megtudhatunk.

Hogyan használjunk superdaemont a bejövő hálózati kapcsolatok figyelésére

Az olyan hálózati szolgáltatások, mint a webszerverek, e-mail szerverek és nyomtatószerverek általában önálló szolgáltatásként futnak, és egy dedikált hálózati porton hallgatnak. Ezek az önálló szolgáltatások mindegyike egymás mellett fut. A klasszikus Sys-V-init alapú rendszerben ezek a szolgáltatások mindegyike a `service` paranccsal vezérelhető. A jelenlegi systemd alapú rendszereken a `systemctl` parancsot használjuk a szolgáltatás kezelésére.

Korábban a számítógépes erőforrások elérhetősége sokkal kisebb volt. Számos szolgáltatás önálló üzemmódban, tandemben történő futtatása nem volt jó opció. Ehelyett egy úgynevezett superdaemont használtak, amely figyelte a bejövő hálózati kapcsolatokat, és igény szerint elindította a megfelelő szolgáltatást. A hálózati kapcsolat kiépítésének ez a módszere egy kicsit több időt vett igénybe. Jól ismert superdaemonok az `inetd` és az `xinetd`. A systemd-re épülő jelenlegi rendszereken a `systemd.socket` egységet lehet hasonló módon használni. Ebben a

részben a `xinetd`-t fogjuk használni a `sshd` daemonhoz való kapcsolatok elfogására, és a kérésre elindítjuk ezt a daemont, hogy bemutassuk, hogyan működött a superdaemon.

A `xinetd` szolgáltatás konfigurálása előtt némi előkészítésre van szükség. Nem számít, hogy Debian vagy Red Hat alapú rendszert használunk. Bár ezeket a magyarázatokat a Debian/GNU Linux 9.9-es verziójával teszteltük, minden jelenlegi, systemd-t tartalmazó Linux rendszeren működniük kell, jelentős változtatások nélkül. Először is győződjünk meg róla, hogy az `openssh-server` és a `xinetd` csomagok telepítve vannak. Most ellenőrizzük, hogy az SSH szolgáltatás működik-e a következőkkel:

```
$ systemctl status sshd
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2020-04-27 09:33:48 EDT; 3h 11min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 430 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 460 (sshd)
    Tasks: 1 (limit: 1119)
   Memory: 5.3M
   CGroup: /system.slice/ssh.service
           └─460 /usr/sbin/sshd -D
```

Ellenőrizzük azt is, hogy az SSH szolgáltatás a 22-es szabványos hálózati porton figyel-e:

```
# lsof -i :22
COMMAND  PID USER   FD   TYPE   DEVICE SIZE/OFF NODE NAME
sshd    1194 root    3u  IPv4 16053268      0t0  TCP *:ssh (LISTEN)
sshd    1194 root    4u  IPv6 16053270      0t0  TCP *:ssh (LISTEN)
```

Végül állítsuk le az SSH-t az alábbi módon:

```
$ sudo systemctl stop sshd.service
```

Abban az esetben, ha véglegesíteni szeretnénk ezt a változást, úgy, hogy az újraindítás után is érvényben maradjon, használjuk a `systemctl disable sshd.service` parancsot!

Most létrehozhatjuk a xinetd konfigurációs fájlt `/etc/xinetd.d/ssh` néhány alapvető beállítással:

```
service ssh
{
    disable     = no
    socket_type = stream
    protocol    = tcp
    wait        = no
    user        = root
    server      = /usr/sbin/sshd
    server_args     = -i
    flags       = IPv4
    interface   = 192.168.178.1
}
```

Indítsuk újra a xinetd szolgáltatást az alábbi módon:

```
$ sudo systemctl restart xinetd.service
```

Ellenőrizzük, hogy melyik szolgáltatás figyel most a bejövő SSH-kapcsolatokra!

```
$ sudo lsof -i :22
COMMAND   PID USER   FD   TYPE  DEVICE SIZE/OFF NODE NAME
xinetd  24098 root    5u  IPv4 7345141      0t0  TCP 192.168.178.1:ssh (LISTEN)
```

Láthatjuk, hogy a xinetd szolgáltatás átvette a 22-es port elérésének irányítását.

Íme néhány további részlet az `xinetd` konfigurációról. A fő konfigurációs fájl az `/etc/xinetd.conf`:

```
# Simple configuration file for xinetd
#
# Some defaults, and include /etc/xinetd.d/

defaults
{

# Please note that you need a log_type line to be able to use log_on_success
# and log_on_failure. The default is the following :
# log_type = SYSLOG daemon info

}
```

```
includedir /etc/xinetd.d
```

Az alapértelmezett beállításokon kívül csak egy utasítás van az include mappa beállítására. Ebben a mappában minden olyan szolgáltatáshoz, amelyet az `xinetd`-vel szeretnénk kezelni, egyetlen konfigurációs fájlt állíthatunk be. Mi ezt a fentiekben az SSH szolgáltatás esetében tettük meg, és az állományt `/etc/xinetd.d/ssh`-nek neveztük el. A konfigurációs fájlok nevei tetszőlegesen megválaszthatók, kivéve a pontot (`.`) tartalmazó vagy tilde-vel (`~`) végződő fájlneveket. De elterjedt gyakorlat, hogy a fájlt a konfigurálni kívánt szolgáltatás után nevezzük el.

Néhány konfigurációs fájlt az `/etc/xinet.d/` mappában a disztribúció már biztosít:

```
$ ls -l /etc/xinetd.d
total 52
-rw-r--r-- 1 root root 640 Feb  5  2018 chargen
-rw-r--r-- 1 root root 313 Feb  5  2018 chargen-udp
-rw-r--r-- 1 root root 502 Apr 11 10:18 daytime
-rw-r--r-- 1 root root 313 Feb  5  2018 daytime-udp
-rw-r--r-- 1 root root 391 Feb  5  2018 discard
-rw-r--r-- 1 root root 312 Feb  5  2018 discard-udp
-rw-r--r-- 1 root root 422 Feb  5  2018 echo
-rw-r--r-- 1 root root 304 Feb  5  2018 echo-udp
-rw-r--r-- 1 root root 312 Feb  5  2018 servers
-rw-r--r-- 1 root root 314 Feb  5  2018 services
-rw-r--r-- 1 root root 569 Feb  5  2018 time
-rw-r--r-- 1 root root 313 Feb  5  2018 time-udp
```

Ezek a fájlok sablonként használhatók abban a ritka esetben, amikor olyan régi szolgáltatásokat kell használnunk, mint például a `daytime`, az időszerver egy nagyon korai implementációja. Mindegyik sablonfájl tartalmazza a `disable = yes` direktívát.

Íme néhány további részlet a fenti `/etc/xinetd.d/ssh` ssh példafájlban használt direktívákról.

```
service ssh
{
    disable     = no
    socket_type = stream
    protocol    = tcp
    wait        = no
    user        = root
    server      = /usr/sbin/sshd
    server_args     = -i
    flags       = IPv4
```

```
    interface   = 192.168.178.1
}
```

`service`

A xinetd által irányítandó szolgáltatások listája. Használhatunk akár egy portszámot, mint például 22, vagy a `/etc/services`-ben a portszámhoz rendelt nevet, például `ssh`.

`{`

A részletes beállítások egy nyitó kapcsos zárójellel kezdődnek.

`disable`

Ha szeretnénk aktiválni ezeket a beállításokat, állítsuk be a `no` értéket! Ha ideiglenesen kikapcsolnánk, akkor pedig a `yes` értéket.

`socket_type`

Kiválaszthatjuk a `stream` opciót TCP socketekhez vagy a `dgram` opciót UDP socketekhez és így tovább.

`protocol`

Kiválaszthatjuk vagy a TCP-t, vagy az UDP-t.

`wait`

TCP kapcsolatok esetén ez általában `no`.

`user`

Az ebben a sorban indított szolgáltatás ennek a felhasználónak a tulajdonában lesz.

`server`

Az `xinetd` által indítandó szolgáltatás teljes elérési útja.

`server_args`

Opciókat adhatunk hozzá a szolgáltatáshoz. Ha egy szuperszerver indítja el, sok szolgáltatás speciális opciót igényel. Az SSH esetében ez az `-i` kapcsoló.

`flags`

Választhatunk IPv4-et, IPv6-ot és másokat.

`interface`

A hálózati interfész, amelyet a `xinetd`-nek kell irányítania. Megjegyzés: választhatjuk a `bind` direktívát is, ami csak az `interface` szinonimája.

`}`

Záró kapcsos zárójellel zárjuk le.

A szuperszerver `xinetd` által indított szolgáltatások utódai a systemd socket egységek. Egy systemd socket egység beállítása nagyon egyszerű és könnyű, mivel az SSH-hoz már rendelkezésre áll egy előre definiált systemd socket egység. Győződjünk meg róla, hogy az `xinetd` és az SSH szolgáltatások nem futnak!

Most már csak el kell indítani az SSH socket egységet:

```
$ sudo systemctl start ssh.socket
```

Annak ellenőrzésére, hogy melyik szolgáltatás hallgat-e a 22-es porton, ismét az `lsof`-ot használjuk. Figyeljük meg, hogy itt a `-P` kapcsolót használtuk, hogy a kimeneten a szolgáltatás neve helyett a portszámot jelenítsük meg:

```
$ sudo lsof -i :22 -P
COMMAND PID USER   FD   TYPE   DEVICE SIZE/OFF NODE NAME
systemd   1 root   57u  IPv6 14730112      0t0  TCP *:22 (LISTEN)
```

Ahhoz, hogy ez a munkamenet teljes legyen, meg kell próbálnunk bejelentkezni a szerverre egy általunk választott SSH klienssel.

TIP Ha a `systemctl start ssh.socket` nem működik az általunk használt disztribúcióban, próbáljuk meg a `systemctl start sshd.socket`-et!

Szükségtelen daemonok ellenőrzése a szolgáltatásokban

Biztonsági okokból, valamint a rendszer erőforrásainak ellenőrzése érdekében fontos, hogy áttekintést kapjunk arról, hogy milyen szolgáltatások futnak. A nem szükséges és nem használt szolgáltatásokat le kell tiltani. Például ha nem osztunk meg weboldalakat, akkor nincs szükség olyan webszerverek futtatására sem, mint az Apache vagy az nginx.

SyS-V-init alapú rendszereken a következőkkel ellenőrizhetjük az összes szolgáltatás állapotát:

```
$ sudo service --status-all
```

Ellenőrizzük, hogy a parancs által listázott szolgáltatások szükségesek-e, és tiltsuk le az összes feleslegeset (Debian-alapú rendszerek esetén):

```
$ sudo update-rc.d SERVICE-NAME remove
```

Red Hat-alapú rendszerek esetén:

```
$ sudo chkconfig SERVICE-NAME off
```

A modern systemd alapú rendszereken a következő módon listázhatjuk az összes futó szolgáltatást:

```
$ systemctl list-units --state active --type service
```

Ezután minden egyes felesleges szolgáltatásegységet letilthatunk az alábbi módon:

```
$ sudo systemctl disable UNIT --now
```

Ez a parancs leállítja a szolgáltatást, és eltávolítja a szolgáltatások listájáról, hogy a következő rendszerindításkor ne indulhasson el.

Emellett a hallgatózó hálózati szolgáltatások áttekintéséhez régebbi rendszereken használhatjuk a `netstat` programot (feltéve, hogy telepítve van a `net-tools` csomag):

```
$ netstat -ltu
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
tcp        0      0 localhost:mysql         0.0.0.0:*               LISTEN
tcp6       0      0 [::]:http               [::]:*                  LISTEN
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
udp        0      0 0.0.0.0:bootpc          0.0.0.0:*
```

Modernebb rendszereken pedig az ezzel egyenértékű `ss` (“socket services”) parancsot használhatjuk:

```
$ ss -ltu
Netid      State         Recv-Q        Send-Q      Local Address:Port          Peer
Address:Port
udp        UNCONN        0             0                 0.0.0.0:bootpc
0.0.0.0:*
```

```
tcp        LISTEN       0            128              0.0.0.0:ssh
0.0.0.0:*
tcp        LISTEN       0            80             127.0.0.1:mysql
0.0.0.0:*
tcp        LISTEN       0            128                 *:http
*:*
tcp        LISTEN       0            128                [::]:ssh
[::]:*
```

TCP wrapperek, mint egyfajta egyszerű tűzfal

Azokban az időkben, amikor még nem álltak rendelkezésre tűzfalak a Linuxhoz, TCP wrappereket használtak a hálózati kapcsolatok biztosítására. Manapság sok program már nem engedelmeskedik a TCP wrappereknek. A legújabb Red Hat (pl. Fedora 29) alapú disztribúciókban a TCP wrapperek támogatása teljesen megszűnt. Hogy támogatni tudjuk a TCP wrappereket még mindig használó régebbi Linux rendszereket, hasznos lehet, ha rendelkezünk némi alapismerettel erről a technológiáról.

Egyszerű példaként ismét az SSH szolgáltatást fogjuk használni. A szolgáltatásnak a példánk hostján csak a helyi hálózatról kell elérhetőnek lennie. Először is ellenőrizzük, hogy az SSH daemon használja-e a `libwrap` könyvtárat, amely TCP wrapper támogatást nyújt:

```
$ ldd /usr/sbin/sshd | grep "libwrap"
        libwrap.so.0 => /lib/x86_64-linux-gnu/libwrap.so.0 (0x00007f91dbec0000)
```

Most hozzáadjuk a következő sort az `/etc/hosts.deny` fájlhoz:

```
sshd: ALL
```

Végül konfigurálunk egy kivételt az `/etc/hosts.allow` fájlban a helyi hálózatról érkező SSH-kapcsolatokra:

```
sshd: LOCAL
```

A változások azonnal hatályba lépnek, nincs szükség semmilyen szolgáltatás újraindítására. Ezt az `ssh` klienssel ellenőrizhetjük.

Gyakorló feladatok

1. Hogyan lehet a korábban zárolt `emma` fiók zárolását feloldani?

2. Korábban az `emma` fióknak volt egy lejárati dátuma. Hogyan lehet a lejárati dátumot `never`-re állítani?

3. Képzeljük el, hogy a nyomtatási feladatokat kezelő CUPS nyomtatási szolgáltatásra nincs szükség a szerveren. Hogyan lehet a szolgáltatást véglegesen letiltani? Hogyan ellenőrizhetjük, hogy a megfelelő port már nem aktív?

4. Telepítettük az nginx webszervert. Hogyan tudjuk ellenőrizni, hogy az nginx támogatja-e a TCP wrappereket?

Gondolkodtató feladatok

1. Ellenőrizzük, hogy a `/etc/nologin` fájl létezése megakadályozza-e a `root` felhasználó bejelentkezését?

2. A `/etc/nologin` fájl létezése megakadályozza a jelszó nélküli bejelentkezést SSH-kulcsokkal?

3. Mi történik bejelentkezéskor, ha a `/etc/nologin` fájlban csak ez a szöveg szerepel: `login currently is not possible`?

4. Megkaphatja-e egy közönséges felhasználó (`emma`) az `/etc/passwd` fájlban lévő `root` felhasználóról szóló információkat, például a `grep root /etc/passwd` paranccsal?

5. Egy közönséges, `emma` nevű felhasználó lekérdezheti-e az `/etc/shadow` fájlban található saját kódolt jelszaváról szóló információt, például a `grep emma /etc/shadow` paranccsal?

6. Milyen lépéseket kell tenni a xinetd által kezelendő ősrégi nappali szolgáltatás engedélyezéséhez és ellenőrzéséhez? Megjegyzés: ez csak egy kísérlet, éles környezetben ne próbáljuk ki!

Összefoglalás

Ebben a leckében megtanultuk:

1. Melyik fájlban vannak tárolva a jelszavak, valamint néhány jelszóval kapcsolatos biztonsági beállítás, pl. lejárati idő.
2. A `xinetd` superdaemon célját, valamint hogyan lehet futtatni és igény szerint elindítani az `sshd` szolgáltatást.
3. Annak ellenőrzését, hogy mely hálózati szolgáltatások futnak, és hogyan lehet letiltani a felesleges szolgáltatásokat.
4. A TCP wrapperek használatát egyszerű tűzfalként.

A leckében használt parancsok:

`chage`

Egy felhasználói jelszó életkorának módosítása.

`chkconfig`

Egy klasszikus parancs, amelyet eredetileg a Red Hat alapú rendszereken használtak annak beállítására, hogy egy szolgáltatás elinduljon-e a rendszerindításkor vagy sem.

`netstat`

Egy klasszikus segédprogram (most már a `net-tools` csomagban), amely megjeleníti a rendszer hálózati portjait elérő daemonokat és azok használatát.

`nologin`

Egy parancs, amely a felhasználó shellje helyett használható, hogy megakadályozza a bejelentkezést.

`passwd`

Egy felhasználó jelszavának létrehozására vagy módosítására szolgál.

`service`

Régebbi módszer egy daemon állapotának ellenőrzésére, például egy szolgáltatás leállítására vagy indítására.

`ss`

A `netstat` modern megfelelője, de további információkat is megjelenít a rendszerben használt különböző socketekről.

`systemctl`

A rendszervezérlő parancs, amely a systemd használatával a számítógépen lévő szolgáltatások és socketek különböző aspektusainak vezérlésére szolgál.

`update-rc.d`

A `chkconfig`-hoz hasonló klasszikus parancs, amely engedélyezi vagy letiltja a rendszer indítását a Debian alapú disztribúciók indításakor.

`xinetd`

Egy superdaemon, amely képes igény szerint szabályozni egy hálózati szolgáltatáshoz való hozzáférést, így a szolgáltatás inaktív marad, amíg ténylegesen nem hívják valamilyen feladat elvégzésére.

Válaszok a gyakorló feladatokra

1. Hogyan lehet a korábban zárolt `emma` fiók zárolását feloldani?

   A szuperfelhasználó a `passwd -u emma` paranccsal feloldhatja a fiók zárolását.

2. Korábban az `emma` fióknak volt egy lejárati dátuma. Hogyan lehet a lejárati dátumot `never`-re állítani?

   A szuperfelhasználó a `chage -E -1 emma` parancsot használhatja erre a célra. Ez a beállítás a `chage -l emma` paranccsal ellenőrizhető.

3. Képzeljük el, hogy a nyomtatási feladatokat kezelő CUPS nyomtatási szolgáltatásra nincs szükség a szerveren. Hogyan lehet a szolgáltatást véglegesen letiltani? Hogyan ellenőrizhetjük, hogy a megfelelő port már nem aktív?

   Szuperfelhasználóként:

   ```
   systemctl disable cups.service --now
   ```

   Most már ellenőrizhetjük:

   ```
   netstat -l | grep ":ipp "` or `ss -l | grep ":ipp "
   ```

4. Telepítettük az nginx webszervert. Hogyan tudjuk ellenőrizni, hogy az nginx támogatja-e a TCP wrappereket?

   A

   ```
   ldd /usr/sbin/nginx | grep "libwrap"
   ```

   parancs egy bejegyzést jelenít meg, ha az nginx támogatja a TCP wrappereket.

Válaszok a gondolkodtató feladatokra

1. Ellenőrizzük, hogy a `/etc/nologin` fájl létezése megakadályozza-e a `root` felhasználó bejelentkezését?

   A `root` továbbra is képes bejelentkezni.

2. A `/etc/nologin` fájl létezése megakadályozza a jelszó nélküli bejelentkezést SSH-kulcsokkal?

   Igen, a jelszó nélküli bejelentkezések is meg lesznek akadályozva.

3. Mi történik bejelentkezéskor, ha a `/etc/nologin` fájlban csak ez a szöveg szerepel: `login currently is not possible`?

   A `login currently is not possible` üzenet jelenik meg és a bejelentkezés megakadályozásra kerül.

4. Megkaphatja-e egy egyszerű felhasználó (`emma`) az `/etc/passwd` fájlban lévő `root` felhasználóról szóló információkat, például a `grep root /etc/passwd` paranccsal?

   Igen, mert minden felhasználónak olvasási joga van ehhez a fájlhoz.

5. Egy egyszerű, `emma` nevű felhasználó lekérdezheti-e az `/etc/shadow` fájlban található saját kódolt jelszaváról szóló információt, például a `grep emma /etc/shadow` paranccsal?

   Nem, mivel az egyszerű felhasználóknak nincs olvasási jogosultságuk ezen a fájlon.

6. Milyen lépéseket kell tenni a xinetd által kezelendő ősrégi nappali szolgáltatás engedélyezéséhez és ellenőrzéséhez? Megjegyzés: ez csak egy kísérlet, éles környezetben ne próbáljuk ki!

   Először módosítsuk a `/etc/xinetd.d/daytime` fájlt, és állítsuk be a `disable = no` direktívát! Másodszor indítsuk újra az xinetd szolgáltatást: `systemctl restart xinetd.service` (vagy `service xinetd restart` SyS-V-Init rendszereken). Most már ellenőrizhetjük, hogy működik-e az `nc localhost daytime`! Az `nc` helyett használhatjuk a `netcat`-et is.

110.3 Az adatok védelme titkosítással

Hivatkozás az LPI célkitűzésre

LPIC-1 version 5.0, Exam 102, Objective 110.3

Súlyozás

4

Kulcsfontosságú ismeretek

- Az OpenSSH 2 kliens alapvető konfigurációja és használata
- Az OpenSSH 2 szerver host kulcsok szerepének megértése
- Alapvető GnuPG konfiguráció, használat és visszavonás
- A GPG használata fájlok titkosítására, visszafejtésére, aláírására és ellenőrzésére
- Az SSH port tunnelek (beleértve az X11 tunneleket is) megértése

A használt fájlok, kifejezések és segédprogramok listája

- `ssh`
- `ssh-keygen`
- `ssh-agent`
- `ssh-add`
- `~/.ssh/id_rsa` és `id_rsa.pub`
- `~/.ssh/id_dsa` és `id_dsa.pub`
- `~/.ssh/id_ecdsa` és `id_ecdsa.pub`
- `~/.ssh/id_ed25519` és `id_ed25519.pub`
- `/etc/ssh/ssh_host_rsa_key` és `ssh_host_rsa_key.pub`

- `/etc/ssh/ssh_host_dsa_key` és `ssh_host_dsa_key.pub`
- `/etc/ssh/ssh_host_ecdsa_key` és `ssh_host_ecdsa_key.pub`
- `/etc/ssh/ssh_host_ed25519_key` és `ssh_host_ed25519_key.pub`
- `~/.ssh/authorized_keys`
- `ssh_known_hosts`
- `gpg`
- `gpg-agent`
- `~/.gnupg/`

110.3 Lecke 1

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	110 Biztonság
Fejezet:	110.3 Az adatok védelme titkosítással
Lecke:	1/2

Bevezetés

Az adatok titkosítással történő védelme a mai rendszeradminisztráció számos területén kiemelkedő fontosságú — még inkább, amikor a rendszerekhez való távoli hozzáférésről van szó. Az olyan nem biztonságos megoldásokkal szemben, mint a *telnet*, *rlogin* vagy *FTP*, az *SSH* (*Secure Shell*) protokollt a biztonság szem előtt tartásával tervezték. A nyilvános kulcsú kriptográfiát használva hitelesíti a hostokat és a felhasználókat, és titkosítja az összes későbbi információcserét. Az SSH továbbá *port alagutak* (port tunnel) létrehozására is használható, ami — többek között — lehetővé teszi, hogy egy nem titkosított protokoll titkosított SSH-kapcsolaton keresztül továbbítson adatokat. Az SSH protokoll jelenlegi, ajánlott verziója a 2.0. Az *OpenSSH* az SSH protokoll szabad és nyílt forráskódú implementációja.

Ez a lecke az *OpenSSH* kliens alapvető konfigurációját, valamint az *OpenSSH* szerver host kulcsainak szerepét tárgyalja. Az SSH port tunnelek koncepciója is szóba kerül. Két gépet fogunk használni a következő beállításokkal:

Gép szerepe	OS	IP-cím	Hostnév	Felhasználó
Kliens	Debian GNU/Linux 10 (buster)	`192.168.1.55`	`debian`	`carol`
Szerver	openSUSE Leap 15.1	`192.168.1.77`	`halof`	`ina`

Az OpenSSH kliens alapvető konfigurációja és használata

Bár az OpenSSH szerver és kliens külön csomagokban érkezik, általában telepíthetünk egy metacsomagot, amely egyszerre biztosítja mindkettőt. Az SSH-szerverrel való távoli munkamenet létrehozásához az `ssh` parancsot használjuk, megadva a felhasználót, akinek a nevével a távoli gépen csatlakozni szeretnénk, valamint a távoli gép IP-címét vagy hostnevét. Amikor először csatlakozunk egy távoli hosthoz, egy ehhez hasonló üzenetet kapunk:

```
carol@debian:~$ ssh ina@192.168.1.77
The authenticity of host '192.168.1.77 (192.168.1.77)' can't be established.
ECDSA key fingerprint is SHA256:5JF7anupYipByCQm2BPvDHRVFJJixeslmppi2NwATYI.
Are you sure you want to continue connecting (yes/no)?
```

Miután beírtuk, hogy `yes` és megnyomtuk az Entert, a rendszer megkérdezi a távoli felhasználó jelszavát. Ha sikeresen megadtuk, megjelenik egy figyelmeztető üzenet, majd bejelentkezünk a távoli gépen:

```
Warning: Permanently added '192.168.1.77' (ECDSA) to the list of known hosts.
Password:
Last login: Sat Jun 20 10:52:45 2020 from 192.168.1.4
Have a lot of fun...
ina@halof:~>
```

Az üzenetek eléggé maguktól értetődőek: mivel ez volt az első alkalom, hogy kapcsolatot létesítettünk a `192.168.1.77` távoli szerverrel, annak hitelességét nem lehetett ellenőrizni egyetlen adatbázis alapján sem. Ezért a távoli szerver a nyilvános kulcsának `ECDSA kulcsujjlenyomatát` (key fingerprint) adta meg (az `SHA256` hash függvényt használva). A kapcsolat elfogadása után a távoli szerver nyilvános kulcsa hozzá lett adva a *known hosts* adatbázishoz, így lehetővé vált a szerver hitelesítése a jövőbeli kapcsolatokhoz. Az *known hosts* nyilvános kulcsainak listáját a `known_hosts` fájlban tartjuk, amely az `~/.ssh`-ban található:

```
ina@halof:~> exit
logout
Connection to 192.168.1.77 closed.
carol@debian:~$ ls .ssh/
known_hosts
```

Mind az `.ssh`, mind a `known_hosts` az első távoli kapcsolat létrehozása után jött létre. Az `~/.ssh` az alapértelmezett mappa a felhasználó-specifikus konfigurációs és hitelesítési információk számára.

NOTE Használhatjuk az `ssh`-t arra is, hogy csak egyetlen parancsot hajtsunk végre a távoli gépen, majd visszatérjünk a helyi terminálra (pl.: `ssh ina@halof ls` futtatása).

Ha ugyanazt a felhasználót használjuk a helyi és a távoli gépen, akkor az SSH-kapcsolat létrehozásakor nem szükséges a felhasználónév megadása. Ha például a `debian` rendszeren `carol` felhasználóként vagyunk bejelentkezve, és a `halof` rendszerhez szintén `carol` felhasználóként szeretnénk csatlakozni, akkor egyszerűen írjuk be az `ssh 192.168.1.77` vagy az `ssh halof` parancsot (ha a név feloldható):

```
carol@debian:~$ ssh halof
Password:
Last login: Wed Jul  1 23:45:02 2020 from 192.168.1.55
Have a lot of fun...
carol@halof:~>
```

Most tegyük fel, hogy új távoli kapcsolatot hozunk létre egy olyan hosttal, amely történetesen ugyanazzal az IP-címmel rendelkezik, mint a `halof` (ez gyakori dolog, ha DHCP-t használunk a helyi hálózaton). Figyelmeztetést kapunk a *man-in-the-middle* támadás lehetőségére:

```
carol@debian:~$ ssh john@192.168.1.77
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:KH4q3vP6C7e0SEjyG8Wlz9fVlf+jmWJ5139RBxBh3TY.
Please contact your system administrator.
Add correct host key in /home/carol/.ssh/known_hosts to get rid of this message.
```

```
Offending ECDSA key in /home/carol/.ssh/known_hosts:1
  remove with:
  ssh-keygen -f "/home/carol/.ssh/known_hosts" -R "192.168.1.77"
ECDSA host key for 192.168.1.77 has changed and you have requested strict checking.
Host key verification failed.
```

Mivel nem egy *man-in-the-middle* támadással van dolgunk, nyugodtan hozzáadhatjuk az új host nyilvános kulcsának ujjlenyomatát az `.ssh/known_hosts` állományhoz. Ahogy az üzenet is jelzi, először az `ssh-keygen -f "/home/carol/.ssh/known_hosts" -R "192.168.1.77"` parancsot használhatjuk az *offending* kulcs eltávolításához (alternatívaként választhatjuk az `ssh-keygen -R 192.168.1.77` parancsot is, hogy törölje az összes `192.168.1.77` kulcsot az `~/.ssh/known_hosts`-ból). Ezután már tudunk kapcsolatot létesíteni az új hosttal.

Kulcsalapú bejelentkezések

Beállíthatjuk az SSH-klienst úgy, hogy bejelentkezéskor ne jelszavakat adjunk meg, hanem nyilvános kulcsokat. Ez az SSH-n keresztül történő távoli szerverhez való csatlakozás előnyben részesített módja, mivel sokkal biztonságosabb. Először is létre kell hoznunk egy kulcspárt a kliensgépen. Ehhez az `ssh-keygen` parancsot kell használnunk a `-t` kapcsolóval, amely megadja a kívánt titkosítás típusát (esetünkben *Elliptic Curve Digital Signature Algorithm*). Ezután megkérdezi a kulcspár mentési útvonalát (az `~/.ssh/` ideális, és egyben ez az alapértelmezett hely is), valamint egy passphrase-t. Bár a passphrase megadása opcionális, erősen ajánlott mindig használni egyet.

```
carol@debian:~/.ssh$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/carol/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/carol/.ssh/id_ecdsa.
Your public key has been saved in /home/carol/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:tlamD0SaTquPZYdNepwj8XN4xvqmHCbe8g5FKKUfMo8 carol@debian
The key's randomart image is:
+---[ECDSA 256]---+
|      .          |
|     o .         |
|    = o o        |
|     B *         |
|    E B S o      |
|     o & O       |
|      @ ^ =      |
```

```
|     *.@ @.      |
|    o.o+B+o      |
+----[SHA256]-----+
```

NOTE A kulcspár létrehozásakor átadhatjuk az `ssh-keygen`-nek a `-b` kapcsolót, hogy megadjuk a kulcs méretét bitben (pl.: `ssh-keygen -t ecdsa -b 521`).

Az előző parancs két további fájlt hozott létre a `~/.ssh` mappánkban:

```
carol@debian:~/.ssh$ ls
id_ecdsa  id_ecdsa.pub  known_hosts
```

`id_ecdsa`
: This is your private key.

`id_ecdsa.pub`
: This is your public key.

NOTE Az aszimmetrikus kriptográfiában (más néven nyilvános kulcsú kriptográfia) a nyilvános és a titkos kulcsok matematikailag úgy kapcsolódnak egymáshoz, hogy amit az egyik titkosít, azt csak a másik tudja visszafejteni.

A következő dolog, amit meg kell tennünk, az az, hogy hozzáadjuk a nyilvános kulcsát annak a felhasználónak az `~/.ssh/authorized_keys` fájljához, akinek a nevén be szeretnénk jelentkezni a távoli hoston (ha az `~/.ssh` mappa még nem létezik, akkor először létre kell hoznunk). A nyilvános kulcsot többféleképpen is átmásolhatjuk a távoli szerverre: USB pendrive segítségével, az `scp` paranccsal — ami az SSH segítségével továbbítja a fájlt — vagy úgy, hogy *catoljuk* a nyilvános kulcs tartalmát, és átadjuk az `ssh`-nak egy pipe segítségével, így:

```
carol@debian:~/.ssh$ cat id_ecdsa.pub |ssh ina@192.168.1.77 'cat >> .ssh/authorized_keys'
Password:
```

Miután a nyilvános kulcsot hozzáadtuk a távoli gépen lévő `authorized_keys` fájlhoz, két eshetőséggel szembesülhetünk, amikor új kapcsolatot próbálunk létrehozni:

- Ha a kulcspár létrehozásakor nem adtunk meg passphrase-t, akkor automatikusan bejelentkezünk. Bár kényelmes, ez a módszer a helyzettől függően nem feltétlenül biztonságos:

```
carol@debian:~$ ssh ina@192.168.1.77
Last login: Thu Jun 25 20:31:03 2020 from 192.168.1.55
```

```
Have a lot of fun...
ina@halof:~>
```

- Ha a kulcspár létrehozásakor megadtuk a passphrase-t, akkor azt minden kapcsolódásnál ugyanúgy meg kell adni, mintha jelszó lenne. A nyilvános kulcson kívül ez a módszer egy további biztonsági réteget ad hozzá a passphrase formájában, és ezért biztonságosabbnak tekinthető. Ami azonban a kényelmet illeti, ez pontosan ugyanolyan, mintha minden kapcsolat létrehozásakor meg kellene adni a jelszót. Ha nem használunk passphrase-t, és valakinek sikerül megszereznie a privát SSH-kulcsfájlunkat, akkor hozzáférhet minden olyan szerverhez, amelyen a nyilvános kulcs telepítve van.

```
carol@debian:~/.ssh$ ssh ina@192.168.1.77
Enter passphrase for key '/home/carol/.ssh/id_ecdsa':
Last login: Thu Jun 25 20:39:30 2020 from 192.168.1.55
Have a lot of fun...
ina@halof:~>
```

Van azonban egy módszer, amely ötvözi a biztonságot és a kényelmet: az *SSH authentication agent* (`ssh-agent`) használata. Ennek saját shellt kell létrehoznia, és a munkamenet hátralévő részében a privát kulcsokat — a nyilvános kulcsú hitelesítéshez — a memóriában tartja. Lássuk egy kicsit részletesebben, hogy ez hogyan működik:

1. Az `ssh-agent` használatával indítsunk egy új Bash shellt:

```
carol@debian:~/.ssh$ ssh-agent /bin/bash
carol@debian:~/.ssh$
```

2. Az `ssh-add` paranccsal hozzáadhatjuk a privát kulcsot a memória egy biztonságos területéhez. Ha a kulcspár létrehozásakor megadtuk a passphrase-t — ami ajánlott az extra biztonság érdekében --, akkor a parancs el fogja kérni azt:

```
carol@debian:~/.ssh$ ssh-add
Enter passphrase for /home/carol/.ssh/id_ecdsa:
Identity added: /home/carol/.ssh/id_ecdsa (carol@debian)
```

   Miután hozzáadtuk az identitást, anélkül jelentkezhetünk be bármelyik távoli szerverre, amelyen a nyilvános kulcsunk jelen van, hogy újra be kellene írnunk a passphrase-ünket. A modern asztali számítógépeken bevett gyakorlat, hogy ezt a parancsot a számítógép indításakor kell lefuttatni, mivel a számítógép kikapcsolásáig (vagy a kulcs kézzel történő

kitöltéséig) a memóriában marad.

Zárjuk le ezt a szakaszt azzal, hogy felsoroljuk a négyféle nyilvános kulcsú algoritmust, amelyek az `ssh-keygen` segítségével megadhatók:

`RSA`

Az 1977-ben megjelent, Ron Rivest, Adi Shamir és Leonard Adleman nevével fémjelzett kiadványt az alkotói után nevezték el. Biztonságosnak számít, és ma is széles körben használják. Minimális kulcsmérete 1024 bit (alapértelmezett 2048).

`DSA`

A *Digital Signature Algorithm* nem bizonyult biztonságosnak, és az OpenSSH 7.0-tól kezdve elavult. A DSA kulcsoknak pontosan 1024 bit hosszúságúnak kell lenniük.

`ecdsa`

Az *Elliptic Curve Digital Signature Algorithm* a DSA továbbfejlesztése, ezért biztonságosabbnak tekinthető. Elliptikus görbületű kriptográfiát használ. Az ECDSA kulcshosszát a három lehetséges elliptikus görbeméret egyike határozza meg bitben kifejezve: 256, 384 vagy 521 bit.

`ed25519`

Ez az `EdDSA` — *Edwards-curve Digital Signature Algorithm* — egy olyan implementációja, amely az erősebb 25519-es görbét használja. Ezt tartják a legbiztonságosabbnak az összes közül. Minden Ed25519 kulcs hossza 256 bit.

NOTE Ha `-t` specifikáció nélkül hívjuk meg, az `ssh-keygen` alapértelmezés szerint egy `RSA` kulcspárt fog generálni.

Az OpenSSH szerver hostkulcsainak szerepe

Az OpenSSH globális konfigurációs könyvtára az `/etc` mappában található:

```
halof:~ # tree /etc/ssh
/etc/ssh
├── moduli
├── ssh_config
├── ssh_host_dsa_key
├── ssh_host_dsa_key.pub
├── ssh_host_ecdsa_key
├── ssh_host_ecdsa_key.pub
├── ssh_host_ed25519_key
├── ssh_host_ed25519_key.pub
```

```
├── ssh_host_rsa_key
├── ssh_host_rsa_key.pub
└── sshd_config

0 directories, 11 files
```

A `moduli`, a kliens (`ssh_config`), valamint a szerver (`sshd_config`) konfigurációs fájljain kívül négy kulcspárt találunk — minden támogatott algoritmushoz egy kulcspárt --, amelyek az *OpenSSH* szerver telepítésekor jönnek létre. Mint már említettük, a szerver ezeket a *hostkulcsokat* használja arra, hogy szükség szerint azonosítsa magát a kliensek számára. A nevek mintája a következő:

Privát kulcsok

`ssh_host_` prefix + *algoritmus* + `key` suffix (pl.: `ssh_host_rsa_key`)

Publikus kulcsok (vagy publikus kulcs ujjlenyomatok)

`ssh_host_` prefix + *algoritmus* + `key.pub` suffix (pl.: `ssh_host_rsa_key.pub`)

NOTE

Az ujjlenyomat egy kriptográfiai hash-függvény nyilvános kulcsra történő alkalmazásával jön létre. Mivel az ujjlenyomatok rövidebbek, mint a kulcsok, amelyekre vonatkoznak, jól jönnek bizonyos kulcskezelési feladatok egyszerűsítéséhez.

A privát kulcsokat tartalmazó fájlok jogosultságai `0600` vagy `-rw-------`: csak a tulajdonos (`root`) olvashatja és írhatja. Az összes nyilvános kulcsfájlt pedig a tulajdonos csoport tagjai és mindenki más is olvashatja (`0644` vagy `-rw-r—r--`):

```
halof:~ # ls -l /etc/ssh/ssh_host_*
-rw------- 1 root root 1381 Dec 21 20:35 /etc/ssh/ssh_host_dsa_key
-rw-r--r-- 1 root root  605 Dec 21 20:35 /etc/ssh/ssh_host_dsa_key.pub
-rw------- 1 root root  505 Dec 21 20:35 /etc/ssh/ssh_host_ecdsa_key
-rw-r--r-- 1 root root  177 Dec 21 20:35 /etc/ssh/ssh_host_ecdsa_key.pub
-rw------- 1 root root  411 Dec 21 20:35 /etc/ssh/ssh_host_ed25519_key
-rw-r--r-- 1 root root   97 Dec 21 20:35 /etc/ssh/ssh_host_ed25519_key.pub
-rw------- 1 root root 1823 Dec 21 20:35 /etc/ssh/ssh_host_rsa_key
-rw-r--r-- 1 root root  397 Dec 21 20:35 /etc/ssh/ssh_host_rsa_key.pub
```

A kulcsok ujjlenyomatait az `ssh-keygen -l` kapcsolójának használatával nézhetjük meg. A kulcsfájl elérési útvonalának megadásához az `-f` kapcsolót is meg kell adnunk:

```
halof:~ # ssh-keygen -l -f /etc/ssh/ssh_host_ed25519_key
256 SHA256:8cnPrinC49ZHc+/9Ai5pV+1JfZ4WBRZhd3rDOsc2zlA root@halof (ED25519)
```

```
halof:~ # ssh-keygen -l -f /etc/ssh/ssh_host_ed25519_key.pub
256 SHA256:8cnPrinC49ZHc+/9Ai5pV+1JfZ4WBRZhd3rDOsc2zlA root@halof (ED25519)
```

A kulcs ujjlenyomatának és a véletlenszerűségének megtekintéséhez csak adjuk hozzá a `-v` kapcsolót a következő módon:

```
halof:~ # ssh-keygen -lv -f /etc/ssh/ssh_host_ed25519_key.pub
256 SHA256:8cnPrinC49ZHc+/9Ai5pV+1JfZ4WBRZhd3rDOsc2zlA root@halof (ED25519)
+--[ED25519 256]--+
|              +oo|
|             .+o.|
|        .    ..E.|
|         + .  +.o|
|        S +  + *o|
|          ooo Oo=|
|     . . . =o+.==|
|      = o =oo o=o|
|     o.o +o+..o.+|
+----[SHA256]-----+
```

SSH port tunnelek

Az OpenSSH hatékony továbbítási funkcióval rendelkezik, amelynek segítségével a forrásporton érkező forgalom tunnelbe kerül egy SSH folyamaton keresztül — titkosítva --, amely aztán átirányítja azt egy célállomás portjára. Ez a mechanizmus *port tunnelling* vagy *port forwarding* néven ismert, és olyan fontos előnyei vannak, mint az alábbiak:

- Lehetővé teszi a tűzfalak megkerülését a távoli host portjainak eléréséhez.
- Lehetővé teszi a hozzáférést kívülről a privát hálózaton lévő hosthoz.
- Titkosítást biztosít minden adatcseréhez.

Többé-kevésbé mondhatjuk azt, hogy megkülönböztethetünk helyi és távoli port tunnellinget.

Helyi port tunnel

Helyileg meghatároz egy portot, amely a célállomás felé továbbítja a forgalmat a közbeiktatott SSH-folyamaton keresztül. Az SSH-folyamat futhat a helyi hoston vagy egy távoli szerveren. Ha például valamilyen okból a helyi gépen lévő `8585` portot használó SSH-n keresztül a `www.gnu.org` -hoz szeretnénk a kapcsolatot tunnelezni, akkor valami ilyesmit tehetünk:

```
carol@debian:~$ ssh -L 8585:www.gnu.org:80 debian
carol@debian's password:
Linux debian 4.19.0-9-amd64 #1 SMP Debian 4.19.118-2 (2020-04-29) x86_64

The programs included with the Debian GNU/Linux system are free software;
(...)
Last login: Sun Jun 28 13:47:27 2020 from 127.0.0.1
```

A magyarázat a következő: az `-L` kapcsolóval a `8585` helyi portot adjuk meg, hogy a `www.gnu.org` *http* port `80` portjához csatlakozzunk a `debian`-- a mi *localhost*-unkon futó SSH folyamat segítségével. Ugyanezt a hatást elérhetnénk, ha azt írnánk, hogy `ssh -L 8585:www.gnu.org:80 localhost`. Ha most egy webböngészővel a `http://localhost:8585` címre megyünk, akkor a `www.gnu.org` címre fog továbbítani minket. A példa demonstrálásához a `lynx`-t (a klasszikus, szöveges módú webböngészőt) fogjuk használni:

```
carol@debian:~$ lynx http://localhost:8585
(...)
  * Back to Savannah Homepage
     * Not Logged in
     * Login
     * New User
     * This Page
     * Language
     * Clean Reload
     * Printer Version
     * Search
     * _
(...)
```

Ha pontosan ugyanezt akarnánk csinálni, de a `halof` rendszeren futó SSH folyamaton keresztül csatlakozva, akkor a következőképpen járhatunk el:

```
carol@debian:~$ ssh -L 8585:www.gnu.org:80 -Nf ina@192.168.1.77
Enter passphrase for key '/home/carol/.ssh/id_ecdsa':
carol@debian:~$
carol@debian:~$ lynx http://localhost:8585
(...)
  * Back to Savannah Homepage
     * Not Logged in
     * Login
     * New User
```

```
     * This Page
     * Language
     * Clean Reload
     * Printer Version
     * Search
     * _
(...)
```

Fontos, hogy észrevegyünk három részletet a parancsban:

- Az `-N` kapcsolónak köszönhetően nem jelentkeztünk be a `halof'-ba, hanem helyette port forwardingot végeztünk.
- Az `-f` kapcsoló utasította az SSH-t, hogy a háttérben fusson.
- Megadtuk az `ina` felhasználót a továbbításhoz: `ina@192.168.1.77`

Távoli port tunnel

A távoli port tunnelling (vagy fordított port forwarding) során a távoli szerver egy portján érkező forgalom a helyi gépen futó SSH-folyamathoz, majd onnan a célszerver (amely lehet a helyi gép is) megadott portjára továbbítódik. Tegyük fel például, hogy meg szeretnénk engedni a hálózaton kívülről valakinek, hogy a helyi gépünkön futó Apache webszervert a `halof` (`192.168.1.77`) SSH-szerveren futó `8585` porton keresztül érhesse el! A következő parancsot használnánk:

```
carol@debian:~$ ssh -R 8585:localhost:80 -Nf ina@192.168.1.77
Enter passphrase for key '/home/carol/.ssh/id_ecdsa':
carol@debian:~$
```

Mostantól bárki, aki kapcsolódik a `halof`-hoz a `8585` porton, a `Debian` Apache2 alapértelmezett honlapját fogja látni:

```
carol@debian:~$ lynx 192.168.1.77:8585
(...)
                                                                Apache2 Debian Default
Page: It works (p1 of 3)
   Debian Logo Apache2 Debian Default Page
   It works!

   This is the default welcome page used to test the correct operation of the Apache2 server
after
   installation on Debian systems. If you can read this page, it means that the Apache HTTP
```

```
server
   installed at this site is working properly. You should replace this file (located at
   /var/www/html/index.html) before continuing to operate your HTTP server.
(...)
```

NOTE

Létezik egy harmadik, összetettebb port forwarding típus, amely nem tartozik ennek a leckének a tárgykörébe: a *dinamikus port forwarding*. Ahelyett, hogy egyetlen porttal lépne kapcsolatba, ez a fajta továbbítás különböző TCP-kommunikációkat használ egy sor porton keresztül.

X11 tunnelek

Most, hogy értjük a port tunneleket, fejezzük be a leckét az X11 tunnel (más néven *X11forwarding*) megvitatásával. Az X11 tunnelen keresztül a távoli hoston lévő *X Window System* a helyi gépre kerül továbbításra. Ehhez csak át kell adni az `ssh`-nak a `-X` opciót:

```
carol@debian:~$ ssh -X ina@halof
...
```

Most már elindíthatunk egy grafikus alkalmazást, például a `firefox` webböngészőt a következő eredménnyel: az alkalmazás a távoli szerveren fog futni, de a megjelenítés a helyi gépre lesz továbbítva.

Ha a `-x` kapcsolóval indítunk új SSH-munkamenetet, az *X11forwarding* le lesz tiltva. Ha most próbáljuk meg elindítani a `firefoxot`, a következő hibát fogjuk kapni:

```
carol@debian:~$ ssh -x ina@halof
carol@192.168.0.106's password:
(...)
ina@halof:~$ firefox

(firefox-esr:1779): Gtk-WARNING **: 18:45:45.603: Locale not supported by C library.
    Using the fallback 'C' locale.
Error: no DISPLAY environment variable specified
```

NOTE

A helyi, a távoli és az X11 port forwardinghoz kapcsolódó három konfigurációs direktíva a következő: `AllowTcpForwarding`, `GatewayPorts` és `X11Forwarding`. További információkért írjuk be a `man ssh_config` és/vagy a `man sshd_config` parancsot!

Gyakorló feladatok

1. A kliensgépen `sonya` felhasználóként bejelentkezve hajtsuk végre a következő SSH feladatokat a távoli `halof` szerveren:
   - Listázzuk az `~/.ssh` tartalmát `serena` felhasználóként a távoli gépen; majd térjünk vissza a helyi terminálba!
   - Jelentkezzünk be `serena` felhasználóként a távoli gépre!
   - Jelentkezzünk be `sonya` felhasználóként a távoli gépre!
   - Töröljük az összes `halof`-hoz tartozó kulcsot a helyi `~/.ssh/known_hosts` fájlunkból!
   - Hozzunk létre egy 256 bites `ecdsa` kulcspárt a kliensgépen!
   - Hozzunk létre egy 256 bites `ed25519` kulcspárt a kliensgépen!
2. Rendezzük a végrehajtás szerinti sorrendbe az alábbi lépéseket, amelyek az az *SSH authentication agent* használatával hozzák létre az SSH-kapcsolatot:
   - A kliensen indítsunk egy új Bash-shellt az *authentication agent* számára az `ssh-agent /bin/bash` paranccsal!
   - A kliensen hozzunk létre egy kulcspárt az `ssh-keygen` segítségével!
   - A kliensen adjuk hozzá a privát kulcsot a memória egy biztonságos területéhez az `ssh-add` segítségével!
   - Adjuk hozzá a kliens nyilvános kulcsát annak a felhasználónak az `~/.ssh/authorized_keys` fájljához, akit a távoli hoston be akarunk jelentkeztetni!
   - Ha még nem létezik, hozzuk létre az `~/.ssh`-t azon felhasználó számára, akit a szerveren be szeretnénk jelentkeztetni!
   - Csatlakozzunk a távoli szerverhez!

   A helyes sorrend:

1. lépés:	
2. lépés:	
3. lépés:	
4. lépés:	
5. lépés:	
6. lépés:	

3. A *port forwarding* tekintetében milyen opciót és direktívát használnak a következő tunnel típusok esetében:

Tunnel típusa	Opció	Direktíva
Lokális		
Remote vagy Reverse		
X		

4. Tegyük fel, hogy a kliens terminálján beírjuk az `ssh -L 8888:localhost:80 -Nf ina@halof` parancsot. Még mindig a kliensgépen a böngészővel nyissuk meg a `http://localhost:8888` címet! Mit fogunk kapni?

Gondolkodtató feladatok

1. A biztonsági irányelvek figyelembevételével:
   - Milyen direktívát használnak az `/etc/ssh/sshd_config'-ban a `root` bejelentkezések engedélyezéséhez:
   - Milyen direktívát használhatnánk az `/etc/ssh/sshd_config'-ban ahhoz, hogy csak egy helyi fiókot adjunk meg az SSH-kapcsolatok elfogadásához?
2. Ha ugyanazt a felhasználót használja a kliens és a szerver, milyen `ssh` paranccsal tudjuk átküldeni a kliens nyilvános kulcsát a szerverre, hogy nyilvános kulcsos hitelesítéssel tudjon bejelentkezni?
3. Hozzunk létre két helyi port tunnelt egyetlen paranccsal, amelyek a 8080-as és a 8585-ös nem privilegizált helyi portokat továbbítják a `halof` távoli szerveren keresztül a `www.gnu.org` és a `www.melpa.org` weboldalakra! Az `ina` felhasználót használjuk a távoli szerveren, és ne felejtsük el használni az `-Nf` kapcsolókat:

Összefoglalás

Ebben a leckében az *OpenSSH 2*-t tárgyaltuk, amely a *Secure Shell* protokollt használja a szerver és a kliens közötti kommunikáció titkosítására. Megtanultuk:

- hogyan jelentkezzünk be egy távoli szerverre.
- hogyan futtassunk parancsokat távolról.
- hogyan hozzunk létre kulcspárokat.
- hogyan lehet kulcs-alapú bejelentkezéseket létrehozni.
- hogyan használjuk az *authentication agentet* az extra biztonság és kényelem érdekében.
- az *OpenSSH* által támogatott publikus-kulcs algoritmusok: `RSA`, `DSA`, `ecdsa`, `ed25519`.
- az *OpenSSH* hostkulcsainak szerepe.
- hogyan hozzunk létre port tunneleket: lokális, távoli és X.

A leckében az alábbi parancsokat tárgyaltuk:

`ssh`
: Bejelentkezés vagy parancsok végrehajtása egy távoli gépen.

`ssh-keygen`
: Autentikációs kulcsok generálása, menedzselése vagy konvertálása.

`ssh-agent`
: OpenSSH autentikációs agent.

`ssh-add`
: Privát kulcs identitások hozzáadása az autentikációs agenthez.

Válaszok a gyakorló feladatokra

1. A kliensgépen `sonya` felhasználóként bejelentkezve hajtsuk végre a következő SSH feladatokat a távoli `halof` szerveren:
   - Listázzuk az `~/.ssh` tartalmát `serena` felhasználóként a távoli gépen; majd térjünk vissza a helyi terminálba!

     ```
     ssh serena@halof ls .ssh
     ```

   - Jelentkezzünk be `serena` felhasználóként a távoli gépre!

     ```
     ssh serena@halof
     ```

   - Jelentkezzünk be `sonya` felhasználóként a távoli gépre!

     ```
     ssh halof
     ```

   - Töröljük az összes `halof`-hoz tartozó kulcsot a helyi `~/.ssh/known_hosts` fájlunkból!

     ```
     ssh-keygen -R halof
     ```

   - Hozzunk létre egy 256 bites `ecdsa` kulcspárt a kliensgépen!

     ```
     ssh-keygen -t ecdsa -b 256
     ```

   - Hozzunk létre egy 256 bites `ed25519` kulcspárt a kliensgépen!

     ```
     ssh-keygen -t ed25519
     ```

2. Rendezzük a végrehajtás szerinti sorrendbe az alábbi lépéseket, amelyek az az *SSH authentication agent* használatával hozzák létre az SSH-kapcsolatot:
   - A kliensen indítsunk egy új Bash-shellt az *authentication agent* számára az `ssh-agent /bin/bash` paranccsal!
   - A kliensen hozzunk létre egy kulcspárt az `ssh-keygen` segítségével!
   - A kliensen adjuk hozzá a privát kulcsot a memória egy biztonságos területéhez az `ssh-add`

segítségével!

- Adjuk hozzá a kliens nyilvános kulcsát annak a felhasználónak az `~/.ssh/authorized_keys` fájljához, akit a távoli hoston be akarunk jelentkeztetni!
- Ha még nem létezik, hozzuk létre az `~/.ssh`-t azon felhasználó számára, akit a szerveren be szeretnénk jelentkeztetni!
- Csatlakozzunk a távoli szerverhez!

A helyes sorrend:

1. lépés:	A kliensen hozzunk létre egy kulcspárt az `ssh-keygen` segítségével!
2. lépés:	Ha még nem létezik, hozzuk létre az `~/.ssh` -t azon felhasználó számára, akit a szerveren be szeretnénk jelentkeztetni!
3. lépés:	Adjuk hozzá a kliens nyilvános kulcsát annak a felhasználónak az `~/.ssh/authorized_keys` fájljához, akit a távoli hoston be akarunk jelentkeztetni!
4. lépés:	A kliensen indítsunk egy új Bash-shellt az *authentication agent* számára az `ssh-agent /bin/bash` paranccsal!
5. lépés:	A kliensen adjuk hozzá a privát kulcsot a memória egy biztonságos területéhez az `ssh-add` segítségével!
6. lépés:	Csatlakozzunk a távoli szerverhez!

3. A *port forwarding* tekintetében milyen opciót és direktívát használnak a következő tunnel típusok esetében:

Tunnel típusa	**Opció**	**Direktíva**
Lokális	`-L`	`AllowTcpForwarding`
Remote vagy Reverse	`-R`	`GatewayPorts`
X	`-X`	`X11Forwarding`

4. Tegyük fel, hogy a kliens termináljába beírjuk az `ssh -L 8888:localhost:80 -Nf ina@halof` parancsot. Még mindig a kliensgépen a böngészővel nyissuk meg a

`http://localhost:8888` címet! Mit fogunk kapni?

A webszerver `halof` kezdőoldalát, mint `localhost`-ot a szerver szemszögéből értendően.

Válaszok a gondolkodtató feladatokra

1. A biztonsági irányelvek figyelembevételével:
   - Milyen direktívát használnak az `/etc/ssh/sshd_config'-ban a `root` bejelentkezések engedélyezéséhez:

     `PermitRootLogin`

   - Milyen direktívát használhatnánk az `/etc/ssh/sshd_config'-ban ahhoz, hogy csak egy helyi fiókot adjunk meg az SSH-kapcsolatok elfogadásához:

     `AllowUsers`

2. Ha ugyanazt a felhasználót használja a kliens és a szerver, milyen `ssh` paranccsal tudjuk átküldeni a kliens nyilvános kulcsát a szerverre, hogy nyilvános kulcsos hitelesítéssel tudjon bejelentkezni?

   `ssh-copy-id`

3. Hozzunk létre két helyi port tunnelt egyetlen paranccsal, amelyek a 8080-as és a 8585-ös nem privilegizált helyi portokat továbbítják a `halof` távoli szerveren keresztül a `www.gnu.org` és a `www.melpa.org` weboldalakra! Az `ina` felhasználót használjuk a távoli szerveren, és ne felejtsük el használni az `-Nf` kapcsolókat:

   ```
   ssh -L 8080:www.gnu.org:80 -L 8585:www.melpa.org:80 -Nf ina@halof
   ```

110.3 Lecke 2

Tanúsítvány:	LPIC-1
Verzió:	5.0
Témakör:	110 Biztonság
Fejezet:	110.3 Az adatok védelme titkosítással
Lecke:	2/2

Bevezetés

Az előző leckében megtanultuk, hogyan használhatjuk az *OpenSSH*-t a távoli bejelentkezési munkamenetek és minden más későbbi információcsere titkosítására. Más esetekben is előfordulhat, hogy titkosítani szeretnénk fájlokat vagy e-maileket, hogy azok biztonságban és a kíváncsi szemektől védve jussanak el a címzetthez. Az is előfordulhat, hogy digitálisan alá kell írniunk ezeket a fájlokat vagy üzeneteket, hogy ezzel megakadályozzuk a manipulációjukat.

Az ilyen típusú feladatokhoz kiváló eszköz a *GNU Privacy Guard* (más néven *GnuPG* vagy egyszerűen *GPG*), amely a szabadalmaztatott *Pretty Good Privacy* (*PGPG*) szabad és nyílt forráskódú implementációja. A *GPG* az *OpenPGP* szabványt használja, amelyet az *Internet Engineering Task Force* (*IETF*) *OpenPGP munkacsoportja* az RFC 4880-ban definiált. Ebben a leckében a *GNU Privacy Guard* alapjait tekintjük át.

Alapvető GnuPG konfiguráció, használat és visszavonás végrehajtása

Az SSH-hoz hasonlóan a GPG mögöttes mechanizmusa az *aszimmetrikus kriptográfia* vagy

nyilvános kulcsú kriptográfia. A felhasználó kulcspárt generál, amely egy *privát kulcsból* és egy *publikus kulcsból* áll. A kulcsok matematikailag úgy kapcsolódnak egymáshoz, hogy amit az egyik kulcs titkosít, azt csak a másik kulcs tudja visszafejteni. A sikeres kommunikációhoz a felhasználónak el kell küldenie a nyilvános kulcsát a címzettnek.

A GnuPG használata és konfigurációja

A GPG-vel való munkához a parancs a `gpg`. Számos kapcsolót adhatunk meg különböző feladatok elvégzéséhez. Kezdjük egy kulcspár létrehozásával `carol` felhasználóként. Ehhez az `gpg --gen-key` parancsot fogjuk használni:

```
carol@debian:~$ gpg --gen-key
gpg (GnuPG) 2.2.12; Copyright (C) 2018 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: directory '/home/carol/.gnupg' created
gpg: keybox '/home/carol/.gnupg/pubring.kbx' created
Note: Use "gpg --full-generate-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name:

(...)
```

Miután tájékoztat minket — többek között — arról, hogy a `~/.gnupg` konfigurációs mappa és a `~/.gnugpg/pubring.kbx` nyilvános kulcstár létrehozásra került, a `gpg` megkér minket, hogy adjuk meg a valódi nevünket és e-mail címünket:

```
(...)
Real name: carol
Email address: carol@debian
You selected this USER-ID:
    "carol <carol@debian>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit?
```

Ha a kapott `USER-ID` megfelel és megnyomjuk a `O` gombot, akkor a rendszer kérni fogja a passphrase-t (ami ajánlott, hogy elég összetett legyen):

```
┌──────────────────────────────────────────────────────┐
│ Please enter the passphrase to                       │
│ protect your new key                                 │
│                                                      │
│ Passphrase:  │

(...)
```

Néhány utolsó üzenet jelenik meg, amely tájékoztat minket a többi fájl és a kulcsok létrehozásáról, majd befejezi a kulcsgenerálási folyamatot:

```
(...)
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: /home/carol/.gnupg/trustdb.gpg: trustdb created
gpg: key 19BBEFD16813034E marked as ultimately trusted
gpg: directory '/home/carol/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/carol/.gnupg/openpgp-
revocs.d/D18FA0021F644CDAF57FD0F919BBEFD16813034E.rev'
public and secret key created and signed.

pub   rsa3072 2020-07-03 [SC] [expires: 2022-07-03]
      D18FA0021F644CDAF57FD0F919BBEFD16813034E
uid                      carol <carol@debian>
sub   rsa3072 2020-07-03 [E] [expires: 2022-07-03]
```

Most már láthatjuk, hogy mi van a `~/.gnupg` mappában (a GPG konfigurációs mappájában):

```
carol@debian:~/.gnupg$ ls -l
total 16
drwx------ 2 carol carol 4096 Jul  3 23:34 openpgp-revocs.d
drwx------ 2 carol carol 4096 Jul  3 23:34 private-keys-v1.d
-rw-r--r-- 1 carol carol 1962 Jul  3 23:34 pubring.kbx
-rw------- 1 carol carol 1240 Jul  3 23:34 trustdb.gpg
```

Most nézzük meg az egyes fájlok célját:

`opengp-revocs.d`

A kulcspárral együtt létrehozott visszavonási tanúsítvány van itt tárolva. Ennek a mappának a

jogosultságai meglehetősen korlátozottak, mivel bárki, aki hozzáfér a tanúsítványhoz, visszavonhatja a kulcsot (a kulcs visszavonásáról bővebben a következő alfejezetben).

`private-keys-v1.d`

Ez az a mappa, ahol a privát kulcsok vannak tárolva, ezért a jogosultságok korlátozottak.

`pubring.kbx`

Ez a saját, nyilvános kulcstárunk. Ez tárolja a saját és importált nyilvános kulcsokat.

`trustdb.gpg`

A bizalmi adatbázis (trust database). Ez a *Web of Trust* fogalmához kapcsolódik (ami nem tartozik ennek a leckének a tárgykörébe).

NOTE

A *GnuPG 2.1* megjelenése magával hozott néhány jelentős változást, például a `secring.gpg` és `pubring.gpg` fájlok eltűnését a `private-keys-v1.d` és `pubring.kbx` fájlok javára.

A kulcspár létrehozása után a nyilvános kulcsokat a `gpg --list-keys` paranccsal tekinthetjük meg, ami megjeleníti a nyilvános kulcstár tartalmát:

```
carol@debian:~/.gnupg$ gpg --list-keys
/home/carol/.gnupg/pubring.kbx
------------------------------
pub   rsa3072 2020-07-03 [SC] [expires: 2022-07-03]
      D18FA0021F644CDAF57FD0F919BBEFD16813034E
uid           [ultimate] carol <carol@debian>
sub   rsa3072 2020-07-03 [E] [expires: 2022-07-03]
```

A `D18FA0021F644CDAF57FD0F919BBEFD16813034E` hexadecimális string a *publikus kulcs ujjlenyomatunk*.

NOTE

A `USER-ID` (a példában `carol`) mellett létezik a `KEY-ID` is. A `KEY-ID` a nyilvános kulcsunk ujjlenyomatának utolsó 8 hexadecimális számjegyéből áll (`6813 034E`). A kulcs ujjlenyomatunkat a `gpg --fingerprint USER-ID` paranccsal ellenőrizhetjük.

Kulcs terjesztése és visszavonása

Most, hogy megvan a nyilvános kulcsunk, el kell mentenünk (azaz *exportálni* kell) egy fájlba, hogy a jövőbeli címzettek számára elérhetővé tegyük. Ők aztán használhatják majd a nekünk szánt fájlok vagy üzenetek titkosítására (mivel a titkos kulcsot csak mi birtokoljuk, így a titkosítást is

csak mi tudjuk majd visszafejteni és elolvasni). Hasonlóképpen, a címzettek is használni fogják a titkosított vagy aláírt üzenetek/fájlok visszafejtésére és ellenőrzésére. A használandó parancs a `gpg --export`, amelyet a `USER-ID` és egy átirányítás követ az általunk választott kimeneti fájl nevével:

```
carol@debian:~/.gnupg$ gpg --export carol > carol.pub.key
carol@debian:~/.gnupg$ ls
carol.pub.key  openpgp-revocs.d  private-keys-v1.d  pubring.kbx  trustdb.gpg
```

NOTE

Az `-a` vagy `--armor` kapcsoló átadása a `gpg --export`-nak (pl.: `gpg --export --armor carol > carol.pub.key`) ASCII kódolású kimenetet hoz létre (az alapértelmezett bináris OpenPGP formátum helyett), ami biztonságosan elküldhető e-mailben.

Mint már említettük, most el kell küldeni a nyilvános kulcsfájlt (`carol.pub.key`) annak a címzettnek, akivel információt szeretnénk cserélni. Például küldjük el a nyilvános kulcsfájlt `ina` -nak` a `halof` távoli szerverre az `scp`(*secure copy*) segítségével:

```
carol@debian:~/.gnupg$ scp carol.pub.key ina@halof:/home/ina/
Enter passphrase for key '/home/carol/.ssh/id_ecdsa':
carol.pub.key
100% 1740   775.8KB/s   00:00
carol@debian:~/.gnupg$
```

`ina` most a `carol.pub.key` birtokában van. A következő szakaszban ezzel fog egy fájlt titkosítani és elküldeni `carol` számára.

NOTE

A nyilvános kulcsok terjesztésének másik módja a *kulcsszerverek* használata: a nyilvános kulcsot a `gpg --keyserver keyserver-name --send-keys KEY-ID` paranccsal töltjük fel a szerverre, a többi felhasználó pedig a `gpg --keyserver keyserver-name --recv-keys KEY-ID` paranccsal megkapja (azaz *importálja*).

Zárjuk ezt a szakaszt a kulcsok visszavonásával. A kulcsvisszavonást akkor kell használni, ha a privát kulcsok veszélybe kerültek vagy már megszűntek. Az első lépés egy visszavonási tanúsítvány létrehozása az gpg-nek a `--gen-revoke` kapcsoló és a `USER-ID` megadásával. A `--gen-revoke` kapcsolót megelőzheti az `--output` opció, utána pedig egy célfájlnév, azért, hogy a kapott tanúsítványt egy fájlba mentse (a terminálra való kiíratás helyett). A visszavonási folyamat során a kimeneti üzenetek eléggé maguktól értetődőek:

```
sonya@debian:~/.gnupg$ gpg --output revocation_file.asc --gen-revoke sonya
```

```
sec  rsa3072/0989EB7E7F9F2066 2020-07-03 sonya <sonya@debian>

Create a revocation certificate for this key? (y/N) y
Please select the reason for the revocation:
  0 = No reason specified
  1 = Key has been compromised
  2 = Key is superseded
  3 = Key is no longer used
  Q = Cancel
(Probably you want to select 1 here)
Your decision? 1
Enter an optional description; end it with an empty line:
> My laptop was stolen.
>
Reason for revocation: Key has been compromised
My laptop was stolen.
Is this okay? (y/N) y
ASCII armored output forced.
Revocation certificate created.

Please move it to a medium which you can hide away; if Mallory gets
access to this certificate he can use it to make your key unusable.
It is smart to print this certificate and store it away, just in case
your media become unreadable.  But have some caution:  The print system of
your machine might store the data and make it available to others!
```

A visszavonási tanúsítvány a `revocation_file.asc` (ASCII formátum esetén `asc`) fájlba került mentésre:

```
sonya@debian:~/.gnupg$ ls
openpgp-revocs.d  private-keys-v1.d  pubring.kbx  revocation_file.asc  trustdb.gpg
sonya@debian:~/.gnupg$ cat revocation_file.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----
Comment: This is a revocation certificate

iQHDBCABCgAtFiEEiIVjfDnnpieFi0wvnlcN6yLCeHEFAl8ASx4PHQJzdG9sZW4g
bGFwdG9wAAoJEJ5XDesiwnhxT9YMAKkjQiMpo9Uyiy9hyvukPPSrLcmtAGLk4pKS
pLZfzA5kxa+HPQwBglAEvfNRR6VMxqXUgUGYC/IAyQQM62oNAcY2PCPrxyJNgVF7
8l4mMZKvW++5ikjZwyg6WWV0+w6oroeo9qruJFjcu752p4T+9gsHVa2r+KRqcPQe
aZ65sAvsBJlcsUDZqfWUXg2kQp9mNPCdQuqvDaKRgNCHA1zbzNFzXWVd2X5RgFo5
nY+tUP8ZQA9DTQPBLPcggICmfLopMPZYB2bft5geb2mMi2oNpf9CNPdQkdccimNV
aRjqdUP9C89PwTafBQkQiONlsR/dWTFcqprG5KOWQPA7xjeMV8wretdEgsyTxqHp
```

```
v1iRzwjshiJCKBXXvz7wSmQrJ4OfiMDHeS4ipR0AYdO8QCzmOzmcFQKikGSHGMy1
z/YRlttd6NZIKjf1TD0nTrFnRvPdsZOlKYSArbfqNrHRBQkgirOD4JPI1tYKTffq
iOeZFx25K+fj2+0AJjvrbe4HDo5m+Q==
=umI8
-----END PGP PUBLIC KEY BLOCK-----
```

A privát kulcs hatékony visszavonásához most egyesíteni kell a tanúsítványt a kulccsal, ami úgy történik, hogy a visszavonási tanúsítványfájlt importáljuk a kulcstárba:

```
sonya@debian:~/.gnupg$ gpg --import revocation_file.asc
gpg: key 9E570DEB22C27871: "sonya <sonya@debian>" revocation certificate imported
gpg: Total number processed: 1
gpg:    new key revocations: 1
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2022-07-04
```

Most listázzuk a kulcsokat és informálódjunk a visszavont kulcsról:

```
sonya@debian:~/.gnupg$ gpg --list-keys
/home/sonya/.gnupg/pubring.kbx
pub   rsa3072 2020-07-04 [SC] [revoked: 2020-07-04]
      8885637C39E7A627858B4C2F9E570DEB22C27871
uid           [ revoked] sonya <sonya@debian>
```

Végül — de nem utolsósorban — győződjünk meg arról, hogy a visszavont kulcsot elérhetővé tesszük minden olyan fél számára, aki rendelkezik a nyilvános kulcsokkal (beleértve a kulcsszervereket is).

A GPG használata fájlok titkosítására, visszafejtésére, aláírására és hitelesítésére

Az előző szakaszban `carol` elküldte a publikus kulcsát `ina`-nak. Ezt most arra fogjuk használni, hogy megvitassuk, hogyan képes a GPG fájlok titkosítására, visszafejtésére, aláírására és hitelesítésére.

Fájlok titkosítása és visszafejtése

Először is, `ina`-nak importálnia kell `carol` nyilvános kulcsát (`carol.pub.key`) a kulcstárába, hogy elkezdhessen vele dolgozni:

```
ina@halof:~> gpg --import carol.pub.key
gpg: /home/ina/.gnupg/trustdb.gpg: trustdb created
gpg: key 19BBEFD16813034E: public key "carol <carol@debian>" imported
gpg: Total number processed: 1
gpg:               imported: 1
ina@halof:~> gpg --list-keys
/home/ina/.gnupg/pubring.kbx
----------------------------
pub   rsa3072 2020-07-03 [SC] [expires: 2022-07-03]
      D18FA0021F644CDAF57FD0F919BBEFD16813034E
uid           [ unknown] carol <carol@debian>
sub   rsa3072 2020-07-03 [E] [expires: 2022-07-03]
```

Ezután létrehozunk egy fájlt úgy, hogy beleírunk egy szöveget, majd a `gpg` segítségével titkosítjuk (mivel nem írtuk alá `carol` kulcsát, kifejezetten megkérdezésre kerül, hogy akarjuk-e használni a kulcsot):

```
ina@halof:~> echo "This is the message ..." > unencrypted-message
ina@halof:~> gpg --output encrypted-message --recipient carol --armor --encrypt unencrypted-
message
gpg: 0227347CC92A5CB1: There is no assurance this key belongs to the named user
sub  rsa3072/0227347CC92A5CB1 2020-07-03 carol <carol@debian>
 Primary key fingerprint: D18F A002 1F64 4CDA F57F  D0F9 19BB EFD1 6813 034E
      Subkey fingerprint: 9D89 1BF9 39A4 C130 E44B  1135 0227 347C C92A 5CB1

It is NOT certain that the key belongs to the person named
in the user ID.  If you really know what you are doing,
you may answer the next question with yes.

Use this key anyway? (y/N) y
```

Bontsuk fel a `gpg` parancsot:

`--output encrypted-message`

Az eredeti fájl titkosított változatának fájlneve (a példában `encrypted-message`).

`--recipient carol`

A címzett `USER-ID` specifikációja (példánkban `carol`). Ha nincs megadva, a GnuPG kérni fogja (kivéve, ha a `--default-recipient` van megadva).

`--armor`

Ez az opció ASCII kódolású kimenetet eredményez, amely bemásolható egy e-mailbe.

`--encrypt unencrypted-message`

A titkosítandó eredeti fájl fájlnevének megadása.

Most már elküldhetjük az `encrypted-message`-t a `debian`-on lévő `carol`-nak az `scp` segítségével:

```
ina@halof:~> scp encrypted-message carol@debian:/home/carol/
carol@debian's password:
encrypted-message                                                    100%  736
1.8MB/s   00:00
```

Ha most `carol`-ként jelentkezünk be, és megpróbáljuk elolvasni az `encrypted-message` üzenetet, azt fogjuk tapasztalni, hogy az valóban titkosított és — ezért — olvashatatlan:

```
carol@debian:~$ cat encrypted-message
-----BEGIN PGP MESSAGE-----

hQGMAwInNHzJKlyxAQv/brJ8Ubs/xya35sbv6kdRKm1C7ONLxL3OueWA4mCs0Y/P
GBna6ZEUCrMEgl/rCyByj3Yq74kuiTmzxAIRUDdvHfj0TtrOWjVAqIn/fPSfMkjk
dTxKo1i55tLJ+sj17dGMZDcNBinBTP4U1atuN71A5w7vH+XpcesRcFQLKiSOmYTt
F7SN3/5x5J6io4ISn+b0KbJgiJNNx+Ne/ub4Uzk4NlK7tmBklyC1VRualtxcG7R9
1klBPYSld6fTdDwT1Y4MofpyILAiGMZvUR1RXauEKf7OIzwC5gWU+UQPSgeCdKQu
X7QL0ZIBS0Ug2XKrO1k93lmDjf8PWsRIml6n/hNelaOBA3HMP0b6Ozv1gFeEsFvC
IxhUYPb+rfuNFTMEB7xIO94AAmWB9N4qknMxdDqNE8WhA728Plw6y8L2ngsplY15
MR4lIFDpljA/CcVh4BXVe9j0TdFWDUkrFMfaIfcPQwKLXEYJp19XYIaaEazkOs5D
W4pENN0YOcX0KWyAYX6r0l8BF0rq/HMenQwqAVXMG3s8ATuUOeqjBbR1x1qCvRQP
CR/3V73aQwc2j5ioQmhWYpqxiro0yKX2Ar/E6rZyJtJYrq+CUk8O3JoBaudknNFj
pwuRwF1amwnSZ/MZ/9kMKQ==
=g1jw
-----END PGP MESSAGE-----
```

Mivel azonban a privát kulcs a birtokunkban van, könnyen visszafejthetjük az üzenetet, ha megadjuk a gpg-ben a `--decrypt` kapcsolót, majd a titkosított fájl elérési útvonalát (a privát kulcs passphrase-ére szükség lesz):

```
carol@debian:~$ gpg --decrypt encrypted-message
gpg: encrypted with 3072-bit RSA key, ID 0227347CC92A5CB1, created 2020-07-03
      "carol <carol@debian>"
```

```
This is the message ...
```

Megadhatjuk az `--output` opciót is, hogy az üzenetet egy új, titkosítatlan fájlba mentsük:

```
carol@debian:~$ gpg --output unencrypted-message --decrypt encrypted-message
gpg: encrypted with 3072-bit RSA key, ID 0227347CC92A5CB1, created 2020-07-03
      "carol <carol@debian>"
carol@debian:~$ cat unencrypted-message
This is the message ...
```

Fájlok aláírása és hitelesítése

A titkosítás mellett a GPG fájlok aláírására is használható, amihez a `--sign` kapcsoló szükséges. Kezdjük egy új üzenet (`message`) létrehozásával, és írjuk alá a `--sign` opcióval (a privát kulcsunk passphrase-ére lesz szükség):

```
carol@debian:~$ echo "This is the message to sign ..." > message
carol@debian:~$ gpg --output message.sig --sign message
(...)
```

A `gpg` parancs felbontása:

`--output message`
: Az eredeti fájl aláírt változatának fájlneve (példánkban `message.sig`).

`--sign message`
: Az eredeti fájl elérési útvonala.

NOTE A `--sign` használatával a dokumentumot tömörítjük, majd aláírjuk. A kimenet bináris formátumú.

Ezután átküldjük a fájlt az `ina`-ra a `halof`-on az `scp message.sig ina@halof:/home/ina` segítségével. Visszatérve `ina`-ként a `halof`-ra, most már ellenőrizhetjük azt a `--verify` opcióval:

```
ina@halof:~> gpg --verify message.sig
gpg: Signature made Sat 04 jul 2020 14:34:41 CEST
gpg:                using RSA key D18FA0021F644CDAF57FD0F919BBEFD16813034E
gpg: Good signature from "carol <carol@debian>" [unknown]
(...)
```

Ha el is akarjuk olvasni a fájlt, akkor egy új fájlba (esetünkben az `--output` kapcsolóval) kell visszafejteni:

```
ina@halof:~> gpg --output message --decrypt message.sig
gpg: Signature made Sat 04 jul 2020 14:34:41 CEST
gpg:                using RSA key D18FA0021F644CDAF57FD0F919BBEFD16813034E
gpg: Good signature from "carol <carol@debian>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: D18F A002 1F64 4CDA F57F  D0F9 19BB EFD1 6813 034E
ina@halof:~> cat message
This is the message to sign ...
```

GPG-Agent

A leckét az `gpg-agent` rövid bemutatásával zárjuk. Az `gpg-agent` az a daemon, amely a GPG privát kulcsait kezeli (a `gpg` indítja el igény szerint). A leghasznosabb beállítások összefoglalójának megtekintéséhez futtassuk a `gpg-agent --help` vagy a `gpg-agent -h` parancsot:

```
carol@debian:~$ gpg-agent --help
gpg-agent (GnuPG) 2.2.4
libgcrypt 1.8.1
Copyright (C) 2017 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://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.

Syntax: gpg-agent [options] [command [args]]
Secret key management for GnuPG

Options:

     --daemon                      run in daemon mode (background)
     --server                      run in server mode (foreground)
     --supervised                  run in supervised mode
 -v, --verbose                     verbose
 -q, --quiet                       be somewhat more quiet
 -s, --sh                          sh-style command output
 -c, --csh                         csh-style command output
(...)
```

NOTE További információkért tekünstük meg a `gpg-agent` man oldalát!

Gyakorló feladatok

1. Töltsük ki a táblázatot a megfelelő fájlnév megadásával:

Leírás	**Fájlnév**
Bizalmi adatbázis	
A visszavonási tanúsítványok mappája	
A privát kulcsok mappája	
Publikus kulcstár	

2. Válaszoljuk meg az alábbi kérdéseket:
   - Milyen típusú kriptográfiát használ a *GnuPG*?
   - Mi a nyilvános kulcsú kriptográfia két fő összetevője?
   - Mi a `07A6 5898 2D3A F3DD 43E3 DA95 1F3F 3147 FA7F 54C7` publikus kulcs ujjlenyomat `KEY-ID`-je?
   - Milyen módszert használnak a nyilvános kulcsok globális szintű terjesztésére?

3. Helyezzük a következő, privát kulcs visszavonásával kapcsolatos lépéseket a megfelelő sorrendbe:
   - Tegyük elérhetővé a visszavont kulcsot a levelezőpartnerek számára!
   - Hozzunk létre egy visszavonási tanúsítványt!
   - Importáljuk a visszavonási tanúsítványt a kulcstárba!

   A helyes sorrend:

1. lépés:	
2. lépés:	
3. lépés:	

4. A fájlok titkosításával kapcsolatban mit jelent az `--armor` kapcsoló a `gpg --output`

`encrypted-message --recipient carol --armor --encrypt unencrypted-message` parancsban?

Gondolkodtató feladatok

1. A legtöbb `gpg` kapcsolónak van egy hosszú és egy rövid változata is. Töltsük ki a táblázatot a megfelelő rövid változattal:

Hosszú változat	Rövid változat
`--armor`	
`--output`	
`--recipient`	
`--decrypt`	
`--encrypt`	
`--sign`	

2. Válaszoljunk a következő, a kulcsok exportálásával kapcsolatos kérdésekre:
   - Milyen paranccsal exportálhatnánk az összes nyilvános kulcsunkat egy `all.key` nevű fájlba?
   - Milyen paranccsal exportálhatnánk az összes privát kulcsunkat egy `all_private.key` nevű fájlba?

3. Melyik `gpg` kapcsoló teszi lehetővé a legtöbb kulcskezeléssel kapcsolatos feladat elvégzését egy menü megjelenítésével?

4. Milyen `gpg` kapcsolóval lehet cleartext aláírást készíteni?

Összefoglalás

Ebben a leckében a *GNU Privacy Guard* programmal foglalkoztunk, amely kiváló választás a fájlok titkosítására/visszafejtésére és digitális aláírására/ellenőrzésére. Megtanultuk:

- hogyan generáljunk kulcspárokat.
- hogyan listázzuk a kulcsokat a kulcstárunkban.
- a `~/.gnupg` mappa tartalmát.
- mi a `USER-ID` és a `KEY-ID`.
- hogyan terjesszük a nyilvános kulcsokat a levelezőpartnereink között.
- hogyan terjesszük globálisan a nyilvános kulcsokat a kulcsszervereken keresztül.
- hogyan vonjuk vissza a privát kulcsokat.
- hogyan titkosítsunk és fejtsünk vissza fájlokat.
- hogyan írjunk alá és hitelesítsünk fájlatok.
- a *GPG-Agent* alapjait.

A leckében az alábbi parancsokról volt szó:

`gpg`
: *OpenPGP* titkosító és aláíró eszköz.

Válaszok a gyakorló feladatokra

1. Töltsük ki a táblázatot a megfelelő fájlnév megadásával:

Leírás	**Fájlnév**
Bizalmi adatbázis	`trustdb.gpg`
A visszavonási tanúsítványok mappája	`opengp-revocs.d`
A privát kulcsok mappája	`private-keys-v1.d`
Publikus kulcstár	`pubring.kbx`

2. Válaszoljuk meg az alábbi kérdéseket:
   - Milyen típusú kriptográfiát használ a *GnuPG*?

     Nyilvános kulcsú kriptográfia vagy aszimmetrikus kriptográfia.

   - Mi a nyilvános kulcsú kriptográfia két fő összetevője?

     A publikus és a privát kulcsok.

   - Mi a `07A6 5898 2D3A F3DD 43E3 DA95 1F3F 3147 FA7F 54C7` publikus kulcs ujjlenyomat `KEY-ID`-je?

     `FA7F 54C7`

   - Milyen módszert használnak a nyilvános kulcsok globális szintű terjesztésére?

     Kulcsszervereket.

3. Helyezzük a következő, privát kulcs visszavonásával kapcsolatos lépéseket a megfelelő sorrendbe:
   - Tegyük elérhetővé a visszavont kulcsot a levelezőpartnerek számára!
   - Hozzunk létre egy visszavonási tanúsítványt!
   - Importáljuk a visszavonási tanúsítványt a kulcstárba!

     A helyes sorrend:

1. lépés:	Hozzunk létre egy visszavonási tanúsítványt!

2. lépés:	Importáljuk a visszavonási tanúsítványt a kulcstárba!
3. lépés:	Tegyük elérhetővé a visszavont kulcsot a levelezőpartnerek számára!

4. A fájlok titkosításával kapcsolatban mit jelent az `--armor` kapcsoló a `gpg --output encrypted-message --recipient carol --armor --encrypt unencrypted-message` parancsban?

   ASCII kódolású kimenetet készít, amely lehetővé teszi, hogy az így kapott meglévő titkosított fájlt e-mailbe másoljuk.

Válaszok a gondolkodtató feladatokra

A legtöbb `gpg` kapcsolónak van egy hosszú és egy rövid változata is. Töltsük ki a táblázatot a megfelelő rövid változattal:

+

Hosszú változat	Rövid változat
`--armor`	`-a`
`--output`	`-o`
`--recipient`	`-r`
`--decrypt`	`-d`
`--encrypt`	`-e`
`--sign`	`-s`

1. Válaszoljunk a következő, a kulcsok exportálásával kapcsolatos kérdésekre:
   - Milyen paranccsal exportálhatnánk az összes nyilvános kulcsunkat egy `all.key` nevű fájlba?

     `gpg --export --output all.key` vagy `gpg --export -o all.key`

   - Milyen paranccsal exportálhatnánk az összes privát kulcsunkat egy `all_private.key` nevű fájlba?

     `gpg --export-secret-keys --output all_private.key` vagy `gpg --export-secret-keys -o all_private.key` (az `--export-secret-keys` lecserélhető `--export-secret-subkeys`-re egy kicsit eltérő eredménnyel — a `man pgp` segítségével több információt kaphatunk).

2. Melyik `gpg` kapcsoló teszi lehetővé a legtöbb kulcskezeléssel kapcsolatos feladat elvégzését egy menü megjelenítésével?

   `--edit-key`

3. Milyen `gpg` kapcsolóval lehet cleartext aláírást készíteni?

   `--clearsign`

Impresszum

PDF generálva: 2024-03-26

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.