Linux
Professional
Institute

LPIC-1

バージョン 5.0
日本語

101

Table of Contents

Linux
Professional
Institute

課題 101: システムアーキテクチャ

101.1 ハードウェア設定の決定と設定

LPI目標への参照

LPIC-1 v5, Exam 101, Objective 101.1

総重量

2

主な知識分野

- 統合された機器の有効化と無効化。
- 異なる様々な種類のマスストレージの決定。
- デバイスのハードウエアリソースの決定。
- さまざまなハードウェア情報を一覧表示するためのツールとユーティリティ（lsusb、lspciなど）。
- USBデバイスを操作するためのツールとユーティリティ。
- sysfs、udev、dbusの概念的理解。

用語とユーティリティ

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

101.1 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	101 システムアーキテクチャ
Objective:	101.1 ハードウェア設定の決定と構成
Lesson:	1 of 1

はじめに

コンピュータ黎明期から、ビジネスコンピュータやパーソナルコンピュータのメーカーは、さまざまなハードウェアパーツをマシンに組み込んできました。そして、オペレーティングシステムはそれらのパーツをサポートする必要がありました。しかし、オペレーティングシステム開発者にとっては、命令セットやデバイス通信の業界での標準が確立されない限り、これらのサポートはとても困難なものでした。オペレーティングシステムは、アプリケーションに提供する標準化された抽象化レイヤーと同様に、ハードウェアが標準化されることによって、特定のハードウェアモデルに限定されないシステムの作成と保守が容易になりました。ただし、統合された基盤となるハードウェアの複雑さによっては、正常にインストールして正しく機能させるために、オペレーティングシステムの公開について調整する必要があります。

これらの調整の一部は、オペレーティングシステムがインストールされていなくても実行できます。ほとんどのマシンでは、マシンの電源を入れたときに実行できる構成ユーティリティがあります。2000年代半ばまで、構成ユーティリティとしてはBIOS（Basic Input/Output System ：バイオス）が実装されていました。これは、x86マザーボードにある基本的な構成機能を含む標準化されたファームウェアです。2000年代の初めから10年間かけて、x86アーキテクチャに基づくマシンは、BIOSをUEFI（Unified Extensible Firmware Interface）と呼ばれる新しい実装に置き換え始めました。UEFIは、（ハードウェアの）識別、テスト、構成、およびファームウェアのアップグレードのためのより高度な機能を備えています。この大きな変化にもかかわらず、どちらの実装も同じ基本的な機能を

満たしているので、未だにどちらもBIOSと呼ぶことが珍しくありません。

NOTE BIOSとUEFIの類似点と相違点の詳細については、後のレッスンで説明します。

デバイスのアクティベーション

システム構成ユーティリティは、コンピュータの電源を入れたときに、特定のキーを押すと表示されます。どのキーを押すかはメーカーによって異なりますが、通常は`Del`か、`F2`や`F12`などのファンクションキーです。使用するキーの組み合わせは、電源を入れた時に画面に表示されることが多くなっています。

BIOSセットアップユーティリティでは、認識している周辺機器の有効化と無効化、自己診断プログラムの実行、IRQ（割り込み要求）やDMA（ダイレクトメモリアクセス）などのハードウェア設定の変更が可能です。最近のマシンではこれらの設定を変更する必要はほとんどありませんが、特定の問題に対処するために調整が必要になる場合があります。たとえば、デフォルト値よりも速いデータ転送速度と互換性のあるRAMテクノロジを利用するために、（RAMの）製造元が指定した値に変更すると良いでしょう。一部のCPUには、特定のオペレーティングシステムでは必要とされない機能を備えており、それを無効化する機能を（BIOSが）提供します。無効にした機能は、既知のバグを含むCPU機能も無効にできるため、消費電力を削減し、システムの保護を強化できます。

マシンに多くのストレージデバイスが装備されている場合、ブートローダーが格納されているストレージデバイスを、ブート順序リストの一番最初のエントリとして定義することが重要です。間違ったデバイスをブート順序リストに指定すると、オペレーティングシステムがロードされない場合があります。

Linuxでのデバイス検査

正しく識別されたデバイスとそれに必要となるソフトウェアコンポーネントを関連付けるのは、オペレーティングシステムの役割です。ハードウェアが期待どおりに機能しない場合は、問題が発生している箇所を正確に特定することが重要です。一部のハードウェアがオペレーティングシステムによって検出されない場合、そのパーツ（またはそれが接続されているポート）に欠陥がある可能性があります。ハードウェアパーツが正しく検出されても正常に動作しない場合は、オペレーティングシステム側に問題がある可能性があります。したがって、ハードウェア関連の問題に対処する際の最初のステップの1つは、オペレーティングシステムがデバイスを正しく検出しているかどうかを確認することです。 Linuxシステムでハードウェアリソースを識別する基本的な方法が2つあります。特殊なコマンドを使用する方法と、特殊なファイルシステムから特定のファイルを読み出す方法です。

調査のためのコマンド

Linuxシステムにおいて、接続されているデバイスを識別するための2つの重要なコマンドがあります:

`lspci`

PCI（Peripheral Component Interconnect）バスに現在接続されているすべてのデバイスを表示します。 PCIデバイスには、ディスクコントローラのようにマザーボードに接続されたコンポーネントと、外部グラフィックカードのようにPCIスロットに取り付けられた拡張カードがあります。

`lsusb`

現在マシンに接続されているUSB（Universal Serial Bus）デバイスを一覧表示します。USB機器にはあらゆる機能を実現するものが存在しますが、主には入力デバイス（キーボード、ポインティングデバイス）やリムーバブルストレージメディアを接続するために使用されます。

コマンド `lspci` および `lsusb` は、オペレーティングシステムによって識別されたすべてのPCIおよびUSBデバイスのリストを出力します。ただし、すべてのハードウェアパーツには、対応するデバイスを制御するためのソフトウェアコンポーネントが必要であるため、デバイスはまだ完全に動作していない可能性があります。このソフトウェアコンポーネントはkernel module と呼ばれ、公式のLinuxカーネルの一部である場合も、他のソースから追加される場合もあります。

ハードウェアデバイスに関係するLinuxカーネルモジュールは、他のオペレーティングシステムと同様に ドライバ とも呼ばれます。ただし、Linux用のドライバは、デバイスの製造元から提供されるとは限りません。一部のメーカーは個別にインストールする独自のバイナリドライバを提供していますが、多くのドライバは独立した開発者によって作成されています。例えば、以前はWindowsで動作するパーツに、Linux用のドライバがない場合がありましたが、現在では、Linuxベースのオペレーティングシステムは強力なハードウェアサポートを備えており、ほとんどのデバイスが簡単に動作します。

ハードウェアに直接関連するコマンドは、実行するためにroot権限を必要とします。ユーザー権限で実行した時には限られた情報しか表示しないことが多いため、rootとしてログインするか、 `sudo` を使用してコマンドを実行する必要があります。

たとえば、コマンド `lspci` の次の例は、いくつかの識別されたデバイスを示しています。

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

コマンド `lspci` の出力は数十行になる可能性があるため、前の例と次の例には表示の一部を示しています。各行の先頭にある16進数は、対応するPCIデバイスの一意のアドレスで

す。コマンド `lspci` に、`-s` オプションとPCIアドレス、`-v` オプション を指定すると、特定のデバイスに関する詳細な情報を表示します。

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

アドレス `04:02.0` にあるデバイスの詳細が出力されました。デバイス名が `Ralink corp. RT2561/RT61 802.11g PCI` のネットワークコントローラです。 `Subsystem` は、デバイスのブランドとモデル―`Linksys WMP54G v4.1`―を示しており、診断の際に役立ちます。

`kernel driver in use` の行で、対応するカーネルモジュール `rt61pci` が分かります。表示された情報から、以下の事柄が分かります:

1. デバイスが識別されました。
2. 一致するカーネルモジュールがロードされました。
3. デバイスは使用できる状態になっているようです。

指定されたデバイスでどのカーネルモジュールが使用されているかを確認する別の方法は、オプション `-k` です。これは、 `lspci` の最新バージョンで使用できます。

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

選択したデバイスであるNVIDIA GPUボードの場合、`lspci` は、行 `kernel driver in use: nvidia` に、使用中のモジュールの名前が `nvidia` であることを示し、行 `kernel modules: nouveau, nvidia_drm, nvidia` に、関連するすべてのカーネルモジュールをリスト表示します。

コマンド `lsusb` は `lspci` に似ていますが、PCIではなくUSBの情報をリスト表示します。

```
$ lsusb
Bus 001 Device 029: ID 1781:0c9f Multiple Vendors USBtiny
```

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

コマンド `lsusb` は、使用可能なUSBチャネルとそれらに接続されているデバイスを表示します。 `lspci` と同様に、`-v` オプションはより詳細な情報を出力します。`-d` オプションにIDを指定することにより、検査対象のデバイスを指定します。

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

`-t` オプションを使用すると、コマンド `lsusb` は現在のUSBデバイスマッピングを階層ツリーとして表示します。

```
$ lsusb -t
/:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=dwc_otg/1p, 480M
    |__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/4p, 480M
        |__ Port 1: Dev 3, If 0, Class=Hub, Driver=hub/3p, 480M
            |__ Port 2: Dev 11, If 1, Class=Human Interface Device, Driver=usbhid,
1.5M
            |__ Port 2: Dev 11, If 0, Class=Human Interface Device, Driver=usbhid,
1.5M
```

```
            |__ Port 3: Dev 20, If 0, Class=Wireless, Driver=btusb, 12M
            |__ Port 3: Dev 20, If 1, Class=Wireless, Driver=btusb, 12M
            |__ Port 3: Dev 20, If 2, Class=Application Specific Interface,
Driver=, 12M
            |__ Port 1: Dev 7, If 0, Class=Vendor Specific Class, Driver=lan78xx,
480M
        |__ Port 2: Dev 28, If 0, Class=Human Interface Device, Driver=usbhid, 1.5M
        |__ Port 3: Dev 29, If 0, Class=Vendor Specific Class, Driver=, 1.5M
```

すべてのデバイスに、対応するモジュールが関連付けられているとは限りません。ある種のデバイスでは、モジュールの仲介なしで、アプリケーションが直接デバイスと通信することができます。とはいえ、 `lsusb -t` の出力には重要な情報があります。一致するモジュールが存在する場合、その名前が `Driver=btusb` のようにデバイス行の最後に表示されます。デバイス `Class` 行には、`Human Interface Device`、`Wireless`、`Vendor Specific Class`、`Mass Storage` などの一般的なカテゴリが表示されます。前のリストにあるモジュール `btusb` を使用しているデバイスを確認するには、 `lsusb` コマンドの `-s` オプションに `Bus` と `Dev` の両方の番号を指定します。

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

標準的なLinuxシステムには、大量のカーネルモジュールがロードされているのが一般的です。`kmod` パッケージによって提供されるコマンドを使用すると、それらを調べることができます。このパッケージは、Linuxカーネルモジュールに対する、追加、削除、表示、プロパティの確認、依存関係やエイリアスの解決など、一般的なタスクを処理するツールのセットです。たとえば、 `lsmod` コマンドは、現在ロードされているすべてのモジュールを表示します。

```
$ lsmod
Module                  Size  Used by
kvm_intel             138528  0
kvm                   421021  1 kvm_intel
iTCO_wdt               13480  0
iTCO_vendor_support    13419  1 iTCO_wdt
snd_usb_audio         149112  2
snd_hda_codec_realtek  51465  1
snd_ice1712            75006  3
snd_hda_intel          44075  7
arc4                   12608  2
snd_cs8427             13978  1 snd_ice1712
snd_i2c                13828  2 snd_ice1712,snd_cs8427
snd_ice17xx_ak4xxx     13128  1 snd_ice1712
```

```
snd_ak4xxx_adda        18487  2 snd_ice1712,snd_ice17xx_ak4xxx
microcode              23527  0
snd_usbmidi_lib        24845  1 snd_usb_audio
gspca_pac7302          17481  0
gspca_main             36226  1 gspca_pac7302
videodev              132348  2 gspca_main,gspca_pac7302
rt61pci                32326  0
rt2x00pci              13083  1 rt61pci
media                  20840  1 videodev
rt2x00mmio             13322  1 rt61pci
hid_dr                 12776  0
snd_mpu401_uart        13992  1 snd_ice1712
rt2x00lib              67108  3 rt61pci,rt2x00pci,rt2x00mmio
snd_rawmidi            29394  2 snd_usbmidi_lib,snd_mpu401_uart
```

コマンド `lsmod` の出力は、次の3つのカラムに分かれています。

`Module`
: モジュール名

`Size`
: モジュールが占有するRAMの量。（バイト単位）

`Used by`
: 依存するモジュール。

以下に示すオーディオデバイスなど一部のモジュールでは、正しく動作するために別のモジュールを必要とします。

```
$ lsmod | fgrep -i snd_hda_intel
snd_hda_intel          42658  5
snd_hda_codec         155748  3 snd_hda_codec_hdmi,snd_hda_codec_via,snd_hda_intel
snd_pcm                81999  3 snd_hda_codec_hdmi,snd_hda_codec,snd_hda_intel
snd_page_alloc         13852  2 snd_pcm,snd_hda_intel
snd                    59132  19
snd_hwdep,snd_timer,snd_hda_codec_hdmi,snd_hda_codec_via,snd_pcm,snd_seq,snd_hda_co
dec,snd_hda_intel,snd_seq_device
```

3番目のカラム `Used by` は、最初のカラムのモジュールが正しく機能するために必要なモジュールを示しています。接頭辞 `snd` が付いたLinuxサウンドアーキテクチャの多くのモジュールは相互に依存しています。システムに発生している問題を探すときは、現在ロードされているいずれかのモジュールをアンロードすると便利な場合があります。`modprobe` コマンドは、カーネルモジュールのロードとアンロードの両方に使用できます。モジュールと

その関連モジュールをアンロードするには、コマンド `modprobe -r` を使用しますが、実行中のプロセスがモジュールを使用している場合にはエラーになります。たとえば、モジュール `snd-hda-intel` （HDA Intelオーディオデバイス用のモジュール）と、サウンドシステムに関連するその他のモジュールをアンロードするには、次のようにします。

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

`modprobe` コマンドは、システムの実行中にカーネルモジュールをロードおよびアンロードするだけでなく、カーネルのロード時にモジュールのパラメータを変更することができます。これは、コマンドにオプションを渡すこととそれほど変わりません。各モジュールは所定のパラメータを受け入れますが、ほとんどの場合はデフォルト値が推奨され、追加のパラメータは必要ありません。ただし、場合によっては、パラメータを使用してモジュールの動作を変更し、適切に機能させる必要があります。

コマンド `modinfo` にモジュール名のみを引数として与えると、指定されたモジュールの説明、ファイル、作成者、ライセンス、ID、依存関係、および使用可能なパラメータを表示します。モジュールをカスタマイズするパラメータを永続化するには、ファイル `/etc/modprobe.conf` またはディレクトリ `/etc/modprobe.d/` にある接尾辞 `.conf` の個々のファイルに記入します。`-p` オプションを指定すると、`modinfo` コマンドは使用可能なすべてのパラメータを表示して終了します（何もしません）。

```
# modinfo -p nouveau
vram_pushbuf:Create DMA push buffers in VRAM (int)
tv_norm:Default TV norm.
                Supported: PAL, PAL-M, PAL-N, PAL-Nc, NTSC-M, NTSC-J,
                        hd480i, hd480p, hd576i, hd576p, hd720p, hd1080i.
                Default: PAL
                NOTE Ignored for cards with external TV encoders. (charp)
nofbaccel:Disable fbcon acceleration (int)
fbcon_bpp:fbcon bits-per-pixel (default: auto) (int)
mst:Enable DisplayPort multi-stream (default: enabled) (int)
tv_disable:Disable TV-out detection (int)
ignorelid:Ignore ACPI lid status (int)
duallink:Allow dual-link TMDS (default: enabled) (int)
hdmimhz:Force a maximum HDMI pixel clock (in MHz) (int)
config:option string to pass to driver core (charp)
debug:debug string to pass to driver core (charp)
noaccel:disable kernel/abi16 acceleration (int)
modeset:enable driver (default: auto, 0 = disabled, 1 = enabled, 2 = headless)
(int)
atomic:Expose atomic ioctl (default: disabled) (int)
runpm:disable (0), force enable (1), optimus only default (-1) (int)
```

上記のサンプル例は、NVIDIA GPUカードのプロプラエタリなドライバーの代わりになるNouveau Projectが提供するカーネルモジュールである `nouveau` で使用可能なすべてのパラメータを示しています。たとえば、オプション `modeset` を使用すると、ディスプレイの解像度と深度をユーザー空間ではなくカーネル空間に設定することができます。ファイル `/etc/modprobe.d/nouveau.conf` に `options nouveau modeset=0` を追加すると、カーネルによるモード設定が無効になります。

モジュールが問題を引き起こしている場合は、ファイル `/etc/modprobe.d/blacklist.conf` を使用してモジュールのロードを禁止できます。たとえば、モジュール `nouveau` が自動的にロードされることを防ぐには、ファイル `/etc/modprobe.d/blacklist.conf` に行 `blacklist nouveau` を追加します。システムに（プロプラエタリナな）専用モジュール `nvidia` がインストールされており、デフォルトモジュールである `nouveau` を無効にする場合には、この操作を行います。

NOTE

システムにすでに存在するデフォルトの `/etc/modprobe.d/blacklist.conf` ファイルを変更しても構いませんが、特定のカーネルモジュールのみに適用される設定を含む、別の構成ファイル `/etc/modprobe.d/<module_name>.conf` を作成することがお勧めです。

カーネル情報ファイルとデバイスファイル

コマンド `lspci`、`lsusb`、および `lsmod` は、オペレーティングシステムが認識しているハードウェア情報を読み取るためのフロントエンドとして機能します。この種の情報は、ディレクトリ `/proc` および `/sys` 内の特別なファイルに保存されます。これらのディレクトリはディスク上には存在せず、カーネルが使用するメモリ上に作成される、疑似的なファイルシステムへのマウントポイントです。このようなファイルシステムは疑似ファイルシステムと呼ばれ、ストレージ上に保存されることはなく、システムの実行中にのみ存在します。
`/proc` ディレクトリには、実行中のプロセスとハードウェアリソースに関する情報を含むファイルが含まれています。ハードウェアを検査するための `/proc` 内の重要なファイルには次のようなものがあります:

`/proc/cpuinfo`
オペレーティングシステムによって検出されたCPUに関する詳細情報。

`/proc/interrupts`
CPU毎のIOデバイスが使用する割り込み番号のリスト。

`/proc/ioports`
現在登録されている入出力ポート領域のリスト。

`/proc/dma`
使用中の登録済みDMA（ダイレクトメモリアクセス）チャネルのリスト。

`/sys` ディレクトリ内のファイルは、`/proc` 内のファイルと同様の役割を持っています。

ただし、`/sys` ディレクトリはハードウェアに関連するデバイス情報とカーネルデータを格納するという目的に特化しており、`/proc` ディレクトリは実行中のプロセスとさまざまなカーネルの構成データに関する情報も含んでいます。

標準的なLinuxシステムにおいて、デバイスに直接関連するもう1つのディレクトリに `/dev` があります。 `/dev` 内のすべてのファイルは、システムデバイス、特にストレージデバイスに関連付けられています。たとえば、マザーボードの最初のIDEチャネルに接続されているレガシーIDEハードドライブは、ファイル `/dev/hda` で表されます。このディスク内のすべてのパーティションは、`/dev/hda1`、`/dev/hda2` のように（末尾の数字で）表されます。

リムーバブルデバイスは udev サブシステムによって処理されて、それぞれに対応するデバイスが `/dev` に作成されます。Linuxカーネルはハードウェアイベントを検出して、udevプロセスに引き渡します。その後、udevプロセスはデバイスを識別し、事前定義されたルールを使ってデバイスに対応するファイルを `/dev` に動的に作成します。

現在のほとんどのLinuxディストリビューションでは、電源投入時にすでに存在するデバイス（coldplug）と、システムの実行中に追加されるデバイス（hotplug）の識別と構成をudevが担当します。 Udevは、`/sys` にマウントされたハードウェア関連情報の疑似ファイルシステムである SysFS を利用しています。

NOTE

ホットプラグは、USBデバイスが挿入されたときなど、システムの実行中のデバイスの検出と構成を意味する用語です。バージョン2.6以降のLinuxカーネルはホットプラグをサポートしており、ほとんどのシステムバス（PCI、USBなど）がデバイスの接続または切断時にホットプラグイベントを生成できるようになっています。

新しいデバイスが検出されると、udevはディレクトリ `/etc/udev/rules.d/` に保存されている事前定義されたルールから一致するものを検索します。ほとんどの重要なルールはディストリビューションに含まれていますが、新しいルールを追加することもできます。

ストレージデバイス

デバイスから読み取られたブロック単位のデータを、さまざまなサイズや位置のバッファを経由して読み書きすることから、Linuxではストレージデバイスをブロックデバイスと呼びます。すべてのブロックデバイスは、 `/dev` ディレクトリ内のファイルに関連付けられており、そのファイルの名前はデバイス種別（IDE、SATA、SCSIなど）とそのパーティションを示します。たとえば、CD/DVDやフロッピーディスクは `/dev` 内のファイルに関連付けられます。2番目のIDEチャネルに接続されたCD/DVDドライブは `/dev/hdc` に関連付けられます。（`/dev/hda` と `/dev/hdb` は最初のIDEチャネルのマスターデバイスとスレーブデバイス用に予約されています）。古いフロッピーディスクは `/dev/fd0`、`/dev/fd1` などに関連付けられます。

Linuxカーネルバージョン2.4以降、ほとんどのストレージデバイスは、ハードウェア種別に関係なく、SCSIデバイスであるかのように表記されるようになりました。 IDE、SSD、およびUSBブロックデバイスのプレフィックスは `sd` になります。 IDEディスクの場合も `sd` プレフィックスが使用されますが、ドライブがマスターかスレーブかによって3番目の文字

が選択されます（最初のIDEチャネルでは、マスターは `sda`、スレーブは `sdb` になります）。パーティションは数値で表記されます。最初に見つけたブロックデバイスの1番目と2番目のパーティションには `/dev/sda1` と `/dev/sda2` が使用され、2番目に見つけたブロックデバイスの1番目と2番目のパーティションには `/dev/sdb1`、`/dev/sdb2` が使われます。このパターンの例外は、メモリカード（SDカード）とNVMeデバイス（PCI Expressバスに接続されたSSD）です。SDカードの場合、最初に見つけたデバイスの1番目と2番目のパーティションに `/dev/mmcblk0p1` と `/dev/mmcblk0p2` が使用され、2番目に見つけたデバイスの1番目と2番目のパーティションに `/dev/mmcblk1p1` と `/dev/mmcblk1p2` が使われます。NVMeデバイスは、`/dev/nvme0n1p1` や `/dev/nvme0n1p2` のように、プレフィックス `nvme` を使います。

演習

1. システムに2番目のSATAディスクを追加した後、オペレーティングシステムが起動できないとします。すべての部品に欠陥がないことをが判明している場合、このエラーの原因は何でしょうか？

2. 新しく購入したデスクトップコンピュータのPCIバスに接続されている外部ビデオカードが、実際に製造元によって宣伝されているものであることを確認したいが、コンピュータケースを開くと保証が無効になるとします。オペレーティングシステムによって検出されたビデオカードの詳細を一覧表示するには、どのコマンドを使用しますか？

3. 次の行は、コマンド `lspci` によって生成される出力の一部です。

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

   このデバイスで使用されているカーネルモジュールを調べるコマンドは何ですか？

4. システム管理者は、`bluetooth` カーネルモジュールのさまざまなパラメータを、システムを再起動せずに試してみたいと考えています。しかしながら、`modprobe -r bluetooth` を使用してモジュールをアンロードしようとすると、次のエラーが発生します。

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

   このエラーの原因として考えられることは何ですか？

発展演習

1. 時代遅れの接続方法で制御コンピューターと通信する装置など、実稼働環境でレガシーマシンを見つけることは珍しくありません。そういった古いマシンには、特別な注意を払う必要がある特徴があります。たとえば、古いBIOSファームウェアを搭載した一部のx86サーバーは、キーボードが検出できない場合は起動しません。このような問題を回避する方法は何でしょう？

2. ARMアーキテクチャに基づくシングルボードコンピュータなど、x86以外のさまざまなコンピュータアーキテクチャでも、Linuxカーネルを中心に構築されたオペレーティングシステムを利用できます。注意深いユーザーは、Raspberry Piのようなマシンには `lspci` コマンドがないことに気付くでしょう。x86マシンとの違いは何でしょうか？

3. 多くのネットワークルーターには、USBハードドライブなどの外部デバイスの接続を可能にするUSBポートがあります。これらのほとんどはLinuxベースのオペレーティングシステムを使用していますが、ルーターに一般的なブロックデバイスが存在しない場合に、外付けUSBハードドライブは `/dev/` ディレクトリでどのように名付けられますか？

4. 2018年に、Meltdown というハードウェアの脆弱性が発見されました。これは、多くのアーキテクチャのほぼすべてのプロセッサに影響します。 最近のバージョンのLinuxカーネルは、使用中のシステムに脆弱性があるか否かを通知できます。どうするとこの情報を得ることができますか？

まとめ

このレッスンでは、主にx86アーキテクチャにおいてLinuxカーネルがハードウェアリソースを扱う方法に関する一般的な概念を説明しました。このレッスンでは、次のトピックを取り上げました:

- BIOSないしUEFI構成ユーティリティでの設定が、オペレーティングシステムがハードウェアを扱う方法にどのような影響を与えるか。
- 標準的なLinuxシステムに備わったツールを使用して、ハードウェアに関する情報を取得する方法。
- ファイルシステム内で、固定的な記憶装置と取り外し可能な記憶装置を識別する方法。

以下のコマンドと手順を取り上げました:

- 検出されたハードウェアを調査するコマンド: `lspci` と `lsusb` 。
- カーネルモジュールを管理するコマンド: `lsmod` と `modprobe` 。
- ハードウェアに関連する特殊ファイル: `/dev/` ディレクトリと、疑似ファイルシステム `/proc/` と `/sys/` にあるファイル。

演習の解答

1. システムに2番目のSATAディスクを追加した後、オペレーティングシステムが起動できないとします。すべての部品に欠陥がないことをが判明している場合、このエラーの考えられる原因は何でしょうか？

   BIOSセットアップユーティリティで、ブートデバイスの順序を設定します。設定しないとBIOSがブートローダーを実行できないことがあります。

2. 新しく購入したデスクトップコンピュータのPCIバスに接続されている外部ビデオカードが、実際に製造元によって宣伝されているものであることを確認したいが、コンピュータケースを開くと保証が無効になるとします。オペレーティングシステムによって検出されたビデオカードの詳細を一覧表示するには、どのコマンドを使用しますか？

   `lspci` コマンドで、PCIバスに接続されているすべてのデバイスに関する詳細情報を一覧表示します。

3. 次の行は、コマンド `lspci` によって生成される出力の一部です。

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

   このデバイスで使用されているカーネルモジュールを調べるコマンドは何ですか？

   `lspci -s 03:00.0 -v` ないし `lspci -s 03:00.0 -k` コマンド

4. システム管理者は、`bluetooth` カーネルモジュールのさまざまなパラメータを、システムを再起動せずに試してみたいと考えています。しかしながら、`modprobe -r bluetooth` を使用してモジュールをアンロードしようとすると、次のエラーが発生します。

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

   このエラーの原因として考えられることは何ですか？

   モジュール `bluetooth` が、実行中のプロセスによって使用されている。

発展演習の解答

1. 時代遅れの接続方法で制御コンピューターと通信する装置など、実稼働環境でレガシーマシンを見つけることは珍しくありません。そういった古いマシンには、特別な注意を払う必要がある特徴があります。たとえば、古いBIOSファームウェアを搭載した一部のx86サーバーは、キーボードが検出できない場合は起動しません。このような問題を回避する方法は何でしょう？

   BIOS構成ユーティリティで、キーボードが見つからない場合にコンピュータをロックする、というオプションを無効にします。

2. ARMアーキテクチャに基づくシングルボードコンピュータなど、x86以外のさまざまなコンピュータアーキテクチャでも、Linuxカーネルを中心に構築されたオペレーティングシステムを利用できます。注意深いユーザーは、Raspberry Piのようなマシンには `lspci` コマンドがないことに気付くでしょう。x86マシンとの違いは何でしょうか？

   多くのx86マシンとは異なり、Raspberry PiのようなARMベースのコンピューターにはPCIバスがないので、`lspci` コマンドは無意味です。

3. 多くのネットワークルーターには、USBハードドライブなどの外部デバイスの接続を可能にするUSBポートがあります。これらのほとんどはLinuxベースのオペレーティングシステムを使用していますが、ルーターに一般的なブロックデバイスが存在しない場合に、外付けUSBハードドライブは `/dev/` ディレクトリでどのように名付けられますか？

   最新のLinuxカーネルはUSBハードドライブをSATAデバイスとして扱います。したがって、ブロックデバイスがシステムに存在しない場合には、USBドライブに対応するファイルは `/dev/sda` になります。

4. 2018年に、Meltdown というハードウェアの脆弱性が発見されました。これは、多くのアーキテクチャのほぼすべてのプロセッサに影響します。 最近のバージョンのLinuxカーネルは、使用中のシステムに脆弱性があるか否かを通知できます。どうするとこの情報を得ることができますか？

   `/proc/cpuinfo` ファイルには、`bugs:cpu_meltdown` のように、対応するCPUの既知のバグを示す行があります。

101.2 システムの起動

LPI目標への参照

LPIC-1 v5, Exam 101, Objective 101.2

総重量

3

主な知識分野

- 起動時に共通のコマンドをブートローダに提供し、オプションをカーネルに提供する。
- BIOS/UEFIから起動完了までのブートシーケンスの知識を示す。
- SysVinitとsystemdの理解。
- Upstartの知識。
- ログファイルのブートイベントを確認する。

用語とユーティリティ

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

- systemd

101.2 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	101 システムアーキテクチャ
Objective:	101.2 システムの起動
Lesson:	1 of 1

はじめに

マシンを制御するには、オペレーティングシステムの主要コンポーネント～すなわちカーネルを、ブートローダー（bootloader）というプログラムでロードする必要があります。このプログラム自体は、BIOSやUEFIなどのプリインストールされたファームウェアによってロードされます。ブートローダーは、ルートファイルシステムを含むパーティションやオペレーティングシステムを実行するモードなどのパラメータを、カーネルに渡すようにカスタマイズできます。カーネルがロードされると、ブートの過程でハードウェアを識別して構成します。最後にカーネルは、システムサービスの開始と管理を担当するユーティリティを呼び出します。

NOTE 一部のLinuxディストリビューションでは、このレッスンで実行するコマンドにroot権限が必要なことがあります。

BIOSまたはUEFI

x86マシンでブートローダーを実行する手順は、BIOSとUEFIのどちらを使用するかによっ

て異なります。 BIOS（Basic Input/Output System）は、マザーボードに搭載された不揮発性メモリチップに保存され、コンピュータの電源がオンになると実行されるプログラムです。この種のプログラムは ファームウェア と呼ばれ、そのシステムに搭載されたストレージデバイスとは異なる場所に保存されています。BIOSは、（BIOS構成ユーティリティでの定義順序による）最初のストレージデバイスの最初の440バイトに、ブートローダーの最初のステージ（ブートストラップ とも呼ばれる）が置かれていると想定します。ストレージデバイスの最初の512バイトはMBR（Master Boot Record）と呼ばれて、ブートローダーの最初のステージと、DOSパーティションスキーマに基づく標準的なパーティションテーブルを含んでいます。MBRに正しいデータが含まれていない場合は、通常の方法ではシステムを起動できません。

BIOSを搭載したシステムにおいては、一般的に、起動前に以下の処理が行われます:

1. POST（power-on self-test）処理は、マシンの電源がオンになるとすぐに、単純なハードウェア診断を実行します。
2. BIOSは、ビデオ出力、キーボード、ストレージメディアなど、システムをロードするための基本コンポーネントを起動します。
3. BIOSは、MBRからブートローダーの最初のステージをロードします（BIOS設定で指定したデバイスの最初の440バイト）。
4. ブートローダーの第1ステージは、ブートオプションの提示とカーネルのロードを行う、ブートローダーの第2ステージを呼び出します。

UEFI（Unified Extensible Firmware Interface）は、いくつかの重要な点でBIOSとは異なります。 UEFIもBIOSと同様にファームウェアの一種ですが、パーティションを識別して、そこにあるファイルシステムを読み取ることができます。UEFIはMBRではなく、マザーボードに搭載された不揮発性メモリ（NVRAM）に保存されている設定情報を利用します。設定情報に書かれた位置ないしブートメニューで選択された位置から、UEFIと互換性を持った EFIアプリケーション と呼ばれるプログラムが実行されます。EFIアプリケーションには、ブートローダー、オペレーティングシステムセレクター、システム診断および修復用のツールなどがあります。これらは、伝統的なストレージデバイス上のパーティション上の、UEFIと互換性のあるファイルシステムに存在する必要があります。標準的なUEFI互換のファイルシステムは、ブロックデバイスの場合はFAT12、FAT16、FAT32、光メディアの場合はISO-9660です。UEFIのこのアプローチにより、BIOSよりもはるかに高度なツールの実装が可能になっています。

EFIアプリケーションを含むパーティションは、 EFI System Partition または単にESPと呼ばれます。このパーティションを、ルートファイルシステムやユーザーデータのファイルシステムなどに使用してはいけません。ESPパーティションのEFIディレクトリには、NVRAMに保存されているエントリが指すアプリケーションが含まれています。

UEFIを備えたシステムにおいては、一般的にオペレーティングシステムの起動前に以下の処理が行われます。

1. POST（power-on self-test）処理は、マシンの電源がオンになるとすぐに、単純なハー

ドウェア診断を実行します。

2. UEFIは、ビデオ出力、キーボード、ストレージメディアなど、システムをロードするための基本コンポーネントを起動します。
3. UEFIのファームウェアは、NVRAMに格納されている設定情報に従って、ESPパーティションのファイルシステムに格納されているEFIアプリケーションを実行します。通常、実行されるEFIアプリケーションはブートローダーです。
4. 事前定義されたEFIアプリケーションがブートローダーの場合、それがカーネルをロードしてオペレーティングシステムを起動します。

UEFI規格は、Secure Boot と呼ばれる機能もサポートしています。この機能は、署名されたEFIアプリケーション、つまりハードウェアメーカーによって承認されたEFIアプリケーションの実行のみを許可します。この機能により、悪意のあるソフトウェアに対する保護が強化されますが、製造元の保証対象外のオペレーティングシステムのインストールが困難になることがあります。

ブートローダー

x86アーキテクチャでLinux用に最も人気のあるブートローダーはGRUB（Grand Unified Bootloader）です。BIOSまたはUEFIによって呼び出されるとすぐに、GRUBは起動可能なオペレーティングシステムのリストを表示します。リストが自動的に表示されない場合もありますが、BIOSからGRUBが呼び出されているときに`Shift`を押すとリストを呼び出すことができます。UEFIシステムでは、代わりに`Esc`キーを使用します。

GRUBメニューから、インストールされているカーネルのどれをロードするかを選択し、それに新しいパラメータを渡すことができます。ほとんどのカーネルパラメータは、`option=value` というパターンで指定します。使う機会が多いいくつかのパラメータを以下に示します:

`acpi`
ACPIサポートを有効/無効にします。 `acpi=off` は、ACPIのサポートを無効にします。

`init`
代替の初期化デーモンを指定します。たとえば、 `init=/bin/bash` は、Bashシェルを初期化デーモンに指定します。これにより、カーネルがブートすると直ぐにシェルセッションが開始されることになります。

`systemd.unit`
アクティブ化する systemd ターゲットを指定します。たとえば、`systemd.unit=graphical.target` などです。systemdは、 SysV Init で定義されているランレベルも受け入れます。たとえば、ランレベル1をアクティブにするには、カーネルパラメータとして数字 `1` または文字 `S`（“single” の略）を指定します。

mem
システムで使用可能なRAMの量を指定します。このパラメータは、仮想マシンゲストが使用できるRAMの量を制限する場合に便利です。例えば `mem=512M` を使用すると、そのゲストシステムで使用可能なRAMの量が512メガバイトに制限されます。

maxcpus
対称型マルチプロセッサマシンにおいて、システムが認識するプロセッサ（またはプロセッサコア）の数を制限します。仮想マシンにも役立ちます。値 `0` は、マルチプロセッサのサポートをオフにし、カーネルパラメータ `nosmp` と同じ効果があります。パラメータ `maxcpus=2` は、オペレーティングシステムで使用可能なプロセッサの数を2つに制限します。

quiet
ほとんどのブートメッセージを非表示にします。

vga
ビデオモードを選択します。パラメータ `vga=ask` は、選択可能なモードのリストを表示します。

root
ブートローダーで指定されたものとは異なるルートパーティションを設定します。たとえば、 `root=/dev/sda3` などです。

rootflags
ルートファイルシステムのマウントオプション。

ro
ルートファイルシステムを読み取り専用で初期マウントします。

rw
ルートファイルシステムを読み書き両用で初期マウントします。

通常はカーネルパラメータの変更は必要ありませんが、オペレーティングシステム関連の問題を検出して解決するのに役立つ場合があります。カーネルパラメータを、`/etc/default/grub` ファイルの `GRUB_CMDLINE_LINUX` 行に追加すると、再起動後にも有効となります。`/etc/default/grub` を変更したら、`grub-mkconfig -o /boot/grub/grub.cfg` コマンドを実行して、ブートローダーの新しい構成ファイルを再生成する必要があります。オペレーティングシステムが起動すると、そのセッションの開始時に指定されたカーネルパラメータを、`/proc/cmdline` ファイルから読み取ることができます。

NOTE GRUBの構成については、後のレッスンで詳しく説明します。

システムの初期化

オペレーティングシステムは、カーネルだけではなく、さまざな機能を提供する多数のコンポーネントから成っています。これらのコンポーネントの多くは、単純なシェルスクリプトから複雑なサービスプログラムまで、システムの初期化処理によって読み込まれます。システムの初期化中に実行されてすぐに終了する短期間のタスクの多くにはスクリプトが使用されます。オペレーティングシステムの本質的な側面を担当するサービス（デーモン(daemons）とも呼ばれる）は、常に実行され続けます。

さまざまな特徴があるスタートアップスクリプトやデーモンをLinuxディストリビューションに組み込む方法は、バラエティに富んだものでした。そのため、すべてのLinuxディストリビューションのメンテナーとユーザーの期待に応える決定的な解決方法は、長い期間存在しませんでした。とはいえ、ディストリビューションのメンテナが選択したツールは、少なくともシステムサービスを開始、停止、再起動できます。ソフトウェアの更新後に再起動などの操作が自動的に実行されることもありますが、構成ファイルに変更を加えた後のほとんどの場合には、システム管理者がサービスを手動で再起動する必要があります。

また、システム管理者が、状況に応じて特定のデーモン群をアクティブ化できると便利です。たとえば、システムメンテナンスタスクを実行するために、最小限のサービス群のみを実行するといった場面があります。

NOTE

厳密に言えば、オペレーティングシステムとは、ハードウェアを制御しすべてのプロセスを管理するカーネルとそのコンポーネントを指します。しかしながら一般的には、“オペレーティングシステム” という用語をより大きくとらえ、ユーザーが基本的な計算タスクを実行できるソフトウェア環境のさまざまなプログラムのグループ全体を指します。

ブートローダーがカーネルをRAMにロードした直後に、オペレーティングシステムの初期化が開始されます。CPUをカーネルが専有して、基本的なハードウェア構成やメモリアドレス指定など、オペレーティングシステムの基本的な機能のセットアップを行います。

次に、カーネルは initramfs （RAM上の初期ファイルシステム）を開きます。 initramfsは、ブート処理中に一時的なルートファイルシステムとして使用されるファイルシステムを格納したアーカイブです。initramfsファイルの主な目的は、カーネルがオペレーティングシステムの “実際の” ルートファイルシステムにアクセスするために必要なモジュールを提供することです。

ルートファイルシステムが利用可能になるとすぐに、カーネルは `/etc/fstab` で構成されたすべてのファイルシステムをマウントし、最初のプログラムである `init` という名前のユーティリティを実行します。`init` プログラムは、すべての初期化スクリプトとシステムデーモンの実行を担当します。systemd や Upstart など、従来のinitとは異なる初期化デーモンの実装もあります。initプログラムが起動すると、initramfsはRAMから削除されます。

標準的なSysVinit

標準的な SysVinitに基づくサービスマネージャーは、起動するデーモンとリソースを制御するために、ランレベル の概念を使用します。ランレベルには0から6の番号が付けられ、ディストリビューションのメンテナがそれぞれの番号に特定の目的を割り当てます。すべてのディストリビューション間で共通するランレベルは、ランレベル0、1、6のみです。

systemd

systemdは、SysVinitのコマンドとランレベルに対する互換性を備えた、最新のシステムとサービスのマネージャーです。systemdは、ソケットとD-Busを使用したサービスのアクティブ化、オンデマンドでのデーモン実行、 cgroups によるプロセス監視、スナップショットのサポート、システムセッションの回復、マウントポイントの制御、依存関係に基づくサービス制御などを、並行して行います。近年、主要なLinuxディストリビューションでは、デフォルトのシステムマネージャーとしてsystemdの採用が増えています。

Upstart

Upstartは、systemdと同様に、SysVinitの代わりになります。Upstartの目標は、システムサービスを並列的に起動することにより、ブート処理を高速化することです。Upstartは、以前のUbuntuベースのディストリビューションなどで使用されていましたが、今日ではsystemdに取って代わられました。

ブート処理の確認

起動処理の最中に、オペレーティングシステムを完全に停止するほどには重要ではないエラーが発生することがあります。そのようなエラーが、システムの所定の動作を損なうかもしれません。エラーメッセージには、エラーがいつどのように発生したかを示す貴重な情報が含まれているので、後の調査に役立ちます。エラーメッセージが発生しない時でも、起動処理中に収集された情報は、チューニングや設定に役立ちます。

カーネルは、起動時を含むメッセージを、カーネル・リング・バッファ と呼ぶメモリ領域に記録します。初期化処理中にアニメーションが表示されてメッセージが表示されない場合でも、メッセージはカーネルリングバッファに格納されます。ただし、システムの電源がオフになるか、`dmesg --clear` コマンドを実行すると、カーネルリングバッファ内のすべてのメッセージは消えてしまいます。オプションを指定しない `dmesg` コマンドは、カーネルリングバッファ内の現在のメッセージを表示します。

```
$ dmesg
[    5.262389] EXT4-fs (sda1): mounted filesystem with ordered data mode. Opts:
(null)
[    5.449712] ip_tables: (C) 2000-2006 Netfilter Core Team
[    5.460286] systemd[1]: systemd 237 running in system mode.
[    5.480138] systemd[1]: Detected architecture x86-64.
[    5.481767] systemd[1]: Set hostname to <torre>.
```

```
[    5.636607] systemd[1]: Reached target User and Group Name Lookups.
[    5.636866] systemd[1]: Created slice System Slice.
[    5.637000] systemd[1]: Listening on Journal Audit Socket.
[    5.637085] systemd[1]: Listening on Journal Socket.
[    5.637827] systemd[1]: Mounting POSIX Message Queue File System...
[    5.638639] systemd[1]: Started Read required files in advance.
[    5.641661] systemd[1]: Starting Load Kernel Modules...
[    5.661672] EXT4-fs (sda1): re-mounted. Opts: errors=remount-ro
[    5.694322] lp: driver loaded but no devices found
[    5.702609] ppdev: user-space parallel port driver
[    5.705384] parport_pc 00:02: reported by Plug and Play ACPI
[    5.705468] parport0: PC-style at 0x378 (0x778), irq 7, dma 3
[PCSPP,TRISTATE,COMPAT,EPP,ECP,DMA]
[    5.800146] lp0: using parport0 (interrupt-driven).
[    5.897421] systemd-journald[352]: Received request to flush runtime journal
from PID 1
```

`dmesg` の出力は数百行に達することがあるため、上記のリストには、カーネルがsystemdサービスマネージャーを呼び出す部分を抜粋して示しています。行の先頭の値は、カーネルのロードが開始された時から経過した秒数です。

systemdを使用するシステムでは、`journalctl` コマンドに `-b`、`--boot`、`-k`、ないし `--dmesg` オプションを指定すると、ブート時のメッセージを表示します。`journalctl --list-boots` コマンドは、現在のブートからの相対的なブート番号とそのハッシュ値、最初と最後のメッセージのタイムスタンプをリスト表示します。

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

systemdに基づくシステムでは過去のブートログも保持されるので、過去のブートにおけるオペレーティングシステムからのメッセージも調べることができます。`-b0` ないし `--boot=0` オプションを指定すると、現在のブート時のメッセージが表示されます。`-b -1` ないし `--boot=-1` オプションを指定すると、前回のブート時のメッセージを表示します。`-b -2` ないし `--boot=-2` オプションを指定すると、さらにその前のブート時のメッセー

ジを表示します。次の抜粋は、直前のブート時にカーネルがsystemdサービスマネージャーを呼び出した部分を示しています。

```
$ journalctl -b 0
oct 04 00:31:01 ubuntu-host kernel: EXT4-fs (sda1): mounted filesystem with ordered
data mode. Opts: (null)
oct 04 00:31:01 ubuntu-host kernel: ip_tables: (C) 2000-2006 Netfilter Core Team
oct 04 00:31:01 ubuntu-host systemd[1]: systemd 237 running in system mode.
oct 04 00:31:01 ubuntu-host systemd[1]: Detected architecture x86-64.
oct 04 00:31:01 ubuntu-host systemd[1]: Set hostname to <torre>.
oct 04 00:31:01 ubuntu-host systemd[1]: Reached target User and Group Name Lookups.
oct 04 00:31:01 ubuntu-host systemd[1]: Created slice System Slice.
oct 04 00:31:01 ubuntu-host systemd[1]: Listening on Journal Audit Socket.
oct 04 00:31:01 ubuntu-host systemd[1]: Listening on Journal Socket.
oct 04 00:31:01 ubuntu-host systemd[1]: Mounting POSIX Message Queue File System...
oct 04 00:31:01 ubuntu-host systemd[1]: Started Read required files in advance.
oct 04 00:31:01 ubuntu-host systemd[1]: Starting Load Kernel Modules...
oct 04 00:31:01 ubuntu-host kernel: EXT4-fs (sda1): re-mounted. Opts:
commit=300,barrier=0,errors=remount-ro
oct 04 00:31:01 ubuntu-host kernel: lp: driver loaded but no devices found
oct 04 00:31:01 ubuntu-host kernel: ppdev: user-space parallel port driver
oct 04 00:31:01 ubuntu-host kernel: parport_pc 00:02: reported by Plug and Play
ACPI
oct 04 00:31:01 ubuntu-host kernel: parport0: PC-style at 0x378 (0x778), irq 7, dma
3 [PCSPP,TRISTATE,COMPAT,EPP,ECP,DMA]
oct 04 00:31:01 ubuntu-host kernel: lp0: using parport0 (interrupt-driven).
oct 04 00:31:01 ubuntu-host systemd-journald[352]: Journal started
oct 04 00:31:01 ubuntu-host systemd-journald[352]: Runtime journal
(/run/log/journal/abb765408f3741ae9519ab3b96063a15) is 4.9M, max 39.4M, 34.5M free.
oct 04 00:31:01 ubuntu-host systemd-modules-load[335]: Inserted module 'lp'
oct 04 00:31:01 ubuntu-host systemd-modules-load[335]: Inserted module 'ppdev'
oct 04 00:31:01 ubuntu-host systemd-modules-load[335]: Inserted module 'parport_pc'
oct 04 00:31:01 ubuntu-host systemd[1]: Starting Flush Journal to Persistent
Storage...
```

ブート時だけではなく、オペレーティングシステムによって発行されたメッセージは、`/var/log/` ディレクトリ内のファイルに保存されます。カーネルとinitramfsをロードした後に重大なエラーが発生し、オペレーティングシステムが初期化プロセスを続行できない場合は、別のブートメディアを使用してシステムを起動して、エラー発生時に対応するファイルシステムにアクセスします。そして、そのファイルシステムの `/var/log/` の下にあるファイルを検索して、起動プロセスの中断の原因を探します。コマンド `journalctl` の `-D` ないし `--directory` オプションを使用すると、systemdログメッセージのデフォルトの場所である `/var/log/journal/` 以外のディレクトリにあるログメッセージを読み取ることができます。 systemdのログメッセージはプレーンテキストではないので、それらを読

み取るには `journalctl` コマンドを使用します。

演習

1. BIOSファームウェアを搭載したマシンで、ブートストラップのバイナリはどこにありますか？

2. UEFIファームウェアは、EFIアプリケーションと呼ばれる外部プログラムによる機能拡張をサポートします。ただし、これらのアプリケーションは所定の位置に置く必要があります。EFIアプリケーションはシステムのどこに配置されますか？

3. ブートローダーは、カーネルをロードする際にカスタムパラメータを渡すことができます。ルートファイルシステムの位置を誤って設定したためにシステムを起動できない場合に、カーネルに正しいルートファイルシステムの位置 `/dev/sda3` を指示するにはどうしますか？

4. Linuxマシンの起動プロセスが、次のメッセージで終了しました。

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

   この問題の原因として何が考えられますか？

発展演習

1. マシンに複数のオペレーティングシステムがインストールされている場合、ブートローダーは選択できるオペレーティングシステムのリストを表示します。新しくオペレーティングシステムをインストールしたときに、ハードディスクのMBRを上書きしてブートローダーの最初のステージを消去してしまい、他のオペレーティングシステムにアクセスできなくなることがあります。UEFIファームウェアを搭載したマシンでは、このような問題が発生しないのはなぜですか？

2. カスタムカーネルをインストールするときに、適切なinitramfsの提供を忘れるとどうなりますか？

3. ブート時のログは数百行の長さに達するので、読みやすいように `dmesg` コマンドの出力を、`less` コマンドなどのページャーにパイプすることがよくあります。ページャー使用しなくても済むように、自動的に出力をページに分割する `dmesg` コマンドのオプションは何ですか？

4. 停止中のマシンから、ファイルシステム全体を含むハードドライブを取り外して、稼働中のマシンのセカンダリドライブとして接続しました。そのマウントポイントが `/mnt/hd` であるとすると、`/mnt/hd/var/log/journal/` にあるジャーナルファイルの内容を、`journalctl` コマンドで調べるにはどうしますか？

まとめ

このレッスンでは、標準的なLinuxシステムのブートシーケンスについて説明しました。Linuxシステムのブート処理がどのように行われるかを知っていると、システムにアクセスできなくなるおそれがあるエラーを防ぐのに役立ちます。このレッスンでは、以下のトピックについて説明しました:

- BIOSとUEFIにおける起動方法の違い。
- システムの一般的な初期化手順。
- ブート時のメッセージを取り出す方法。

以下のコマンドと手順を取り上げました:

- よく使うカーネルパラメータ。
- ブートメッセージを取り出すコマンド: `dmesg` と `journalctl`。

演習の解答

1. BIOSファームウェアを搭載したマシンで、ブートストラップのバイナリはどこにありますか？

   BIOS構成で最初のストレージデバイスとして定義したデバイスのMBRの中にあります。

2. UEFIファームウェアは、EFIアプリケーションと呼ばれる外部プログラムによる機能拡張をサポートします。ただし、これらのアプリケーションは所定の位置に置く必要があります。EFIアプリケーションはシステムのどこに配置されますか？

   EFIアプリケーションは、任意のストレージブロック上のファイルシステム（通常はFAT32ファイルシステム）であるEFIシステムパーティション（ESP）に配置されます。

3. ブートローダーは、カーネルをロードする際にカスタムパラメータを渡すことができます。ルートファイルシステムの位置を誤って設定したためにシステムを起動できない場合に、カーネルに正しいルートファイルシステムの位置 `/dev/sda3` を指示するにはどうしますか？

   `root=/dev/sda3` のように、`root` パラメータを使用します。

4. Linuxマシンの起動プロセスが、次のメッセージで終了しました。

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

   この問題の原因として何が考えられますか？

   カーネルは、ルートファイルシステムとして指示されたデバイス `/dev/sda3` を見つけることができませんでした。

発展演習の解答

1. マシンに複数のオペレーティングシステムがインストールされている場合、ブートローダーは選択できるオペレーティングシステムのリストを表示します。新しくオペレーティングシステムをインストールしたときに、ハードディスクのMBRを上書きしてブートローダーの最初のステージを消去してしまい、他のオペレーティングシステムにアクセスできなくなることがあります。UEFIファームウェアを搭載したマシンでは、このような問題が発生しないのはなぜですか？

   UEFIマシンは、ブートローダーの最初のステージを格納するために、ハードディスクのMBRを使用しないからです。

2. カスタムカーネルをインストールするときに、適切なinitramfsの提供を忘れるとどうなりますか？

   ルートファイルシステムが、カーネルモジュールとしてコンパイルされているタイプの場合には、ルートファイルシステムにアクセスできません。

3. ブート時のログは数百行の長さに達するので、読みやすいように `dmesg` コマンドの出力を、`less` コマンドなどのページャーにパイプすることがよくあります。ページャー使用しなくても済むように、自動的に出力をページに分割する `dmesg` コマンドのオプションは何ですか？

   `dmesg -H` ないし `dmesg --human` コマンドでは、ページャーが有効になります。

4. 停止中のマシンから、ファイルシステム全体を含むハードドライブを取り外して、稼働中のマシンのセカンダリドライブとして接続しました。そのマウントポイントが `/mnt/hd` であるとすると、`/mnt/hd/var/log/journal/` にあるジャーナルファイルの内容を、`journalctl` コマンドで調べるにはどうしますか？

   `journalctl -D /mnt/hd/var/log/journal` ないし `journalctl --directory=/mnt/hd/var/log/journal` コマンドを使用します。

101.3 ランレベル/ブートターゲットを変更し、システムをシャットダウンまたは再起動する

LPI目標への参照

LPIC-1 v5, Exam 101, Objective 101.3

総重量

3

主な知識分野

- デフォルトのランレベルまたはブートターゲットを設定する。
- シングルユーザーモードを含むランレベル/ブートターゲット間の変更。
- コマンドラインからのシャットダウンと再起動。
- ランレベル/ブートターゲットやその他の主要なシステムイベントを、切り替える前にユーザに警告する 。
- プロセスを正しく終了する。
- acpidの知識。

用語とユーティリティ

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

- `/usr/lib/systemd/`
- `wall`

101.3 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	101 システムアーキテクチャ
Objective:	101.3 ランレベル／ブートターゲットを変更し、システムをシャットダウン/再起動する
Lesson:	1 of 1

はじめに

Unixの設計原則に従ったオペレーティングシステムに共通する機能は、システムの個々の機能を制御するために、専用のプロセスを用いることです。デーモン（一般的には　サービス）と呼ばれるこれらのプロセスは、ネットワークサービス（HTTPサーバー、ファイル共有、電子メールなど）、データベース、オンデマンド構成など、オペレーティングシステムの基盤となる拡張機能を担当します。Linuxはモノリシックカーネルですが、負荷分散やファイアウォールといったオペレーティングシステムの低レベルな機能の多くは、デーモンの手助けにより成り立ちます。

どのデーモンをアクティブにするかは、システムの目的によって異なります。アクティブなデーモンのセットを変更できるように、システム全体を再起動することなく、それぞれのサービスを開始・停止できることが必要です。そのために、主要なLinuxディストリビューションは、システムを制御するための何らかのサービス管理ユーティリティーを提供しています。

サービスは、シェルスクリプトや、プログラムとその設定ファイルによって制御します。System　V（訳注:旧世代のUNIXの名称）における　SysVinit　の実装が最初のサービスマネージャでした。続いて、Upstart　と　systemd　が実装されました。多くのLinuxディストリビューションではSysVinitベースのサービスマネージャーが伝統的に使用されてきま

したが、今日ではsystemdベースのサービスマネージャーが、ほとんどのLinuxディストリビューションで採用されるようになっています。サービスマネージャーは、ブートプロセス中にカーネルによって起動される最初のプログラムであるため、そのPID（プロセス識別番号）は常に `1` です。

SysVinit

SysVinitに基づいたサービスマネージャーは、 ランレベル （runlevel）と呼ばれるシステム状態の定義セットと、それに対応するサービススクリプトから成ります。ランレベルには `0` から `6` の番号が付けられ、通常は次の目的に割り当てられます。

ランレベル0
: システムのシャットダウン。

ランレベル1、sまたは single
: シングルユーザーモード。ネットワークおよびその他の必須ではない機能は起動しない（メンテナンスモード）。

ランレベル2、3または4
: マルチユーザーモード。ユーザーはコンソールまたはネットワークでログインできます。ランレベル2と4はあまり使用されません。

ランレベル5
: マルチユーザーモード。これは、3に加えて、グラフィカルモードのログインを有効化します。

ランレベル6
: システムの再起動。

ランレベルと、それに関連するデーモン/リソースの管理を担当するプログラムは `/sbin/init` です。システムの初期化中に、`init` プログラムは、カーネルパラメータないし `/etc/inittab` ファイルからデフォルトのランレベルを調べて、そのランレベルに関連付けられたスクリプト群を実行します。それぞれのランレベルには、関連付けられた多くのサービススクリプトがあります。それらのスクリプトは通常、`/etc/init.d/` ディレクトリに置かれます。SysVinitベースのディストリビューションでは、ディストリビューションによってそれぞれのランレベルの意味が異なるので、それぞれのランレベルの役割が明記されているのが普通です。

`/etc/inittab` ファイルでは、次の構文を使用します。

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

`id` は、エントリを識別するための4文字までのラベルです。`runlevels` フィールドは、そ

のエントリの `action` を実行する必要があるランレベル番号のリストです。`action` フィールドには、`init` がそのエントリの `process` を実行する方法を定義します。`action` に指定できるキーワードとその意味を以下に示します:

`boot`
: システムの初期化中に `process` を実行します。`runlevels` フィールドは無視されます。

`bootwait`
: システムの初期化中に `process` を実行し、終了するまで `init` は待機します。`runlevels` フィールドは無視されます。

`sysinit`
: ランレベルに関係なく、システムの初期化（boot）が完了した後に `process` を実行します。`runlevels` フィールドは無視されます。

`wait`
: 指定されたランレベルで `process` を実行し、終了するまで `init` は待機します。

`respawn`
: `process` が終了すると、再度 `process` を実行します。

`ctrlaltdel`
: キーボードで `Ctrl`+`Alt`+`Del`を押すと、initプロセスに `SIGINT` シグナルが送られますが、その時に `process` を実行します。

デフォルトのランレベル（カーネルパラメータに明示されていない場合にに選択されるもの）は、`/etc/inittab` の `id:x:initdefault` エントリで定義されます。`x` がデフォルトのランレベル番号です。この数値を `0` または `6` にすると、ブートプロセスが終了するとすぐにシステムがシャットダウンまたは再起動してしまうので、ここに指定することはできません。典型的な `/etc/inittab` ファイルを以下に示します:

```
# Default runlevel
id:3:initdefault:

# Configuration script executed during boot
si::sysinit:/etc/init.d/rcS

# Action taken on runlevel S (single user)
~:S:wait:/sbin/sulogin

# Configuration for each execution level
l0:0:wait:/etc/init.d/rc 0
```

```
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6

# Action taken upon ctrl+alt+del keystroke
ca::ctrlaltdel:/sbin/shutdown -r now

# Enable consoles for runlevels 2 and 3
1:23:respawn:/sbin/getty tty1 VC linux
2:23:respawn:/sbin/getty tty2 VC linux
3:23:respawn:/sbin/getty tty3 VC linux
4:23:respawn:/sbin/getty tty4 VC linux

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

`/etc/inittab` ファイルを変更した後に、`telinit q` コマンドを実行する必要があります。引数 `q`（または `Q`）は、initに設定ファイルをを再読み込みするように指示します。この手順は、`/etc/inittab` の設定を間違えたためにシステムを起動できないことを避けるために重要です。

`init` がランレベルを設定する際に使用するスクリプトは、`/etc/init.d/` ディレクトリに保存されています。それぞれのランレベルには、`/etc/` ディレクトリ内に、`/etc/rc0.d/`、`/etc/rc1.d/`、`/etc/rc2.d/` などの関連付けられたディレクトリがあり、そのランレベルに入るときにスクリプトが実行されます。1つのスクリプトを複数のランレベルで使用することがあるため、これらのディレクトリ内のファイルは `/etc/init.d/` にある実際のスクリプトへのシンボリックリンクです。また、それぞれのランレベルのディレクトリにあるリンク名の先頭文字が、対応するランレベルでサービスを開始するか終了するかを示します。文字 `K` で始まるリンク名は、ランレベルに入るとき（別のランレベルから切り替えられたとき）に、サービスが強制終了（kill）されることを示します。文字 `S` で始まるものは、ランレベルに入るときにサービスが開始（start）されます。たとえば、ランレベル1はネットワーク接続のないシングルユーザーのためのランレベルですから、文字 `K` で始まるネットワークスクリプトへのリンクが多数あります。

`runlevel` コマンドは、システムの現在のランレベルを表示します。`runlevel` コマンドは2つの値を表示します。1つ目は直前のランレベルで、2つ目は現在のランレベルです。

```
$ runlevel
N 3
```

出力の文字 `N` は、前回の起動以降にランレベルが変更されていないことを示しています。この例では、`runlevel 3` がシステムの現在のランレベルです。

`init` コマンドを使用して、実行中のシステムのランレベルを再起動せずに切り替えることができます。`telinit` コマンドを使用してもランレベルを切り替えることができます。たとえば、コマンド `telinit 1`、`telinit s`、`telinit S` は、システムをランレベル1に変更します。

systemd

現在では、システムリソースとサービスを管理するために systemdが最も広く使用されていて、ユニット（units）を使ってそれらを定義します。ユニットは、名前、タイプ、および対応する設定ファイルから構成されます。たとえば、httpd サーバープロセス（Apache Webサーバーなど）のユニットは、Red Hatベースのディストリビューションでは `httpd.service` になり、その構成ファイルは `httpd.service` です（Debianベースのディストリビューションでは、このユニットは `apache2.service` という名前です）。

systemdユニットには7つの異なるタイプがあります。

`service`
: 最も一般的なユニットタイプで、システムリソースの起動、停止、リロードを制御します。

`socket`
: ソケットユニットタイプには、ファイルシステムソケットとネットワークソケットがあります。すべてのソケットユニットには対応するサービスユニットがあり、ソケットが要求を受信したときにサービスユニットがロードされます。

`device`
: デバイスユニットは、カーネルが識別したハードウェアデバイスに関連付けられています。デバイスに対応するudevルールが存在する場合にのみ、デバイスとsystemdユニットが関連付けられます。デバイスユニットは、udevルールのプロパティをパラメータとして使用できるので、あるハードウェアが検出されたときに、依存関係から別のユニットを起動するといった目的のために使用できます。

`mount`
: マウントユニットは、ファイルシステムのマウントポイントの定義であり、`/etc/fstab` のエントリに似ています。

`automount`
: 自動マウントユニットもファイルシステムのマウントポイント定義ですが、自動的にマウントされます。すべての自動マウントユニットには対応するマウントユニットがあり、自動マウントポイントにアクセスすると開始されます。

`target`
ターゲットユニットは、いくつかのユニットをグループ化して単一のユニットとして管理します。

`snapshot`
スナップショットユニットはsystemdマネージャーの状態を保存するものです（すべてのLinuxディストリビューションで利用できるわけではありません）。

systemdユニットを制御するための主なコマンドは `systemctl` です。`systemctl` コマンドで、ユニットの起動、停止、実行、中断、監視などに関するすべてのタスクを実行します。架空の `unit.service` を例に取ると、次のようなアクションを `systemctl` で行えます。

`systemctl start unit.service`
`unit` を開始します。

`systemctl stop unit.service`
`unit` を停止します。

`systemctl restart unit.service`
`unit` を再起動します。

`systemctl status unit.service`
`unit` の状態（実行中かどうかなど）を表示します。

`systemctl is-active unit.service`
`unit` が実行中の場合は `active` を表示し、それ以外の場合は `inactive` を表示します。

`systemctl enable unit.service`
`unit` を有効にします。すなわち、システムの起動時に `unit` が自動起動されます。

`systemctl disable unit.service`
システム起動時に `unit` を自動起動しません。

`systemctl is-enabled unit.service`
`unit` がシステム起動時に自動起動するかどうかを確認します。答えはシェル変数 `$?` に保存されます（以下の注意を参照）。値 `0` は、`unit` がシステムで起動することを示し、値 `1` は、`unit` がシステムで自動起動しないことを示します。

NOTE

systemdの新しいバージョンでは、ユニットが enableであるか disableであるかを文字で表示します。例:

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

同名の（種別が異なる）ユニットがシステムに存在しない場合は、ドットの後のサフィックスを省略できます。たとえば、タイプ `service` の `httpd` ユニットしかない場合は、`systemctl` の引数にユニット名 `httpd` のみを指定すれば十分です。

`systemctl` コマンドで system targets （起動後のターゲット）を制御することもできます。たとえば、 `multi-user.target` ユニットは、マルチユーザーシステム環境に必要なすべてのユニットを組み合わせたもので、SysVを利用するシステムのランレベル番号3に相当します。

コマンド `systemctl isolate` は現在とは異なるターゲットに切り替えます。（訳注: isolate でターゲットを切り替えると、以前のターゲットで有効だったが切り替え後のターゲットで無効となるユニットは停止します）。したがって、手動でターゲットを `multi-user` に切り替えるには：

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

SysVinitのランレベルに対応する `runlevel0.target` から `runlevel6.target` までのターゲットが用意されています。ただし、systemdは `/etc/inittab` ファイルを使用しません。デフォルトのシステムターゲットを変更するには、カーネルパラメータにオプション `systemd.unit` を追加します。たとえば、 `multi-user.target` をデフォルトのターゲットとするには、カーネルパラメータに `systemd.unit=multi-user.target` を追加します。ブートローダーの構成を変更することで、すべてのカーネルパラメータを永続化できます。

デフォルトのターゲットを変更する別の方法は、シンボリックリンク `/etc/systemd/system/default.target` を変更して、目的のターゲットを指すようにすることです。リンクの再定義は、 `systemctl` コマンドで実行できます。

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

同様に、次のコマンドを使用して、システム起動時のデフォルトターゲットを調べられます。

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

デフォルトターゲットが、SysVinitを採用しているシステムにおけるランレベル0（シャットダウン）に相当する `shutdown.target` になることはありません。

ユニットの構成ファイルは、すべて `/lib/systemd/system/` ディレクトリにあります。コマンド `systemctl list-unit-files` は、使用可能なすべてのユニットをリスト表示し、それらのユニットがシステムの起動時に有効であるかどうかを示します。オプション `--type` は、`systemctl list-unit-files --type=service` や `systemctl list-unit-files --type=target` のように、指定したタイプのユニットのみを選択します。

`systemctl list-units` コマンドは、アクティブなユニットと現在のシステムセッション中に起動したユニットをリスト表示します。`list-unit-files` オプションと同様に、`systemctl list-units --type=service` コマンドは `service` ユニットのみを選択し、`systemctl list-units --type=target` コマンドは `target` ユニットのみを選択します。

systemdは、電源関連イベントのトリガーと応答も担当します。コマンド `systemctl suspend` は、システムを低電力モードにし、現在のデータをメモリに保持します。コマンド `systemctl hibernate` は、すべてのメモリデータをディスクにコピーして、システムの電源をオフにした後もシステムの現在の状態を回復できるようにします。このようなイベントに関連するアクションは、ファイル `/etc/systemd/logind.conf` またはディレクトリ `/etc/systemd/logind.conf.d/` 内のファイルで定義します。ただし、`acpid` デーモンなどの電源マネージャーがシステムで実行されている場合は、systemdのこの機能は使用できません。`acpid` デーモンはLinuxのメインの電源マネージャーであり、ラップトップの蓋を閉める、バッテリー残量少、バッテリーの充電レベルなど、電源関連のイベントに応じたアクションを細かく調整できます。

Upstart

Upstartが使用する初期化スクリプトは、`/etc/init/` ディレクトリにあります。システムサービスは、`initctl list` コマンドで一覧表示できます。このコマンドは、サービスの現在の状態と、実行中であればそのPID番号を表示します。

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

```
tty5 start/running, process 1764
failsafe stop/waiting
```

Upstart操作ごとに、専用のコマンドがあります。たとえば、6番目の仮想端末を有効にするには、startコマンドを使います。

```
# start tty6
```

リソースの状態は、`status` コマンドで確認できます。

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

そして、サービスの停止は `stop` コマンドで行います。

```
# stop tty6
```

Upstartは、ランレベルの定義に `/etc/inittab` ファイルを使用しませんが、SysVinitと同様に、コマンド `runlevel` と `telinit` を使用して、ランレベルを確認し、切り替えることができます。

NOTE Upstartは、Ubuntu Linuxディストリビューション用に、プロセスの並列起動を行うために開発されました。 しかし、Ubuntuは、2015年にsystemdに切り替えた後、Upstartを使用していません。

シャットダウンと再起動

システムをシャットダウンまたは再起動するための伝統的なコマンドは、当然のことながら `shutdown` といいます。`shutdown` コマンドは、電源オフの処理にいくつかの機能を追加します。ログインしているすべてのユーザーのシェルセッションに警告メッセージを送信し、新規ログインが禁止されます。コマンド `shutdown` は、SysVinitまたはsystemdによる処理に対する仲介役として機能します。つまり、システムが採用しているサービスマネージャの対応するアクションを呼び出して、要求されたアクションを実行します。

`shutdown` が実行されると、すべてのプロセスが `SIGTERM` シグナルを、続いて `SIGKILL` シグナルを受け取り、システムは停止するかランレベルを変更します。オプション `-h` または `-r` のどちらも使用されていない場合、デフォルトでシステムはランレベル1つまりシングルユーザーモードに切り替わります。 `shutdown` のデフォルト動作を変更するには、次の書式でコマンドを実行します。

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

引数として `time` のみが必須です。 `time` パラメータには、次の形式でいつアクションを実行するかを指定します。

`hh:mm`
: この形式では、実行時刻を時間と分で指定します。

`+m`
: この形式では、実行までに待機する分数を指定します。

`now` または `+0`
: この形式では、即時実行を指定します。

`message` パラメータには、ログインしているユーザーのすべてのターミナルセッションに送信する警告テキストを指定します。

SysVinitの実装では、`Ctrl`+`Alt`+`Del`を押してマシンを再起動できるユーザーを制限できます。これには、`/etc/inittab` ファイルの `ctrlaltdel` 行にある `shutdown` コマンドに、`-a` オプションを指定します。そうすると、`/etc/shutdown.allow` ファイルに名前が登録されているユーザーのみが、`Ctrl`+`Alt`+`Del`キーストロークでシステムを再起動できます。

systemdを使用しているシステムでは、`systemctl` コマンドを使用してマシンの電源をオフにしたり再起動することもできます。システムを再起動するには `systemctl reboot` コマンドを使用し、システムをオフにするには `systemctl poweroff` コマンドを使用します。一般ユーザーはこれらの手順を実行できず、どちらのコマンドも実行するにはroot権限が必要です。

NOTE

一部のLinuxディストリビューションでは、 個別のコマンドとして `poweroff` と `reboot` を `systemctl` にリンクしています。例えば:

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

すべてのメンテナンス作業に、システムの電源を切ったり再起動したりする必要があるわけではありません。しかし、システムの状態をシングルユーザーモードに変更する場合には、ログインしているユーザーの活動が突然の終了によって損害を受けないように、警告することが重要です。

`shutdown` コマンドと同様に、すべてのログイン中のユーザーのターミナルセッションにメッセージを送信するには、`wall` コマンドを使います。メッセージファイルを用意するか、コマンド `wall` のパラメータにメッセージを直接指定します。

演習

1. `telinit` コマンドを使用してシステムを再起動するにはどうすればよいですか？

2. システムがランレベル1に入ると、ファイル `/etc/rc1.d/K90network` に関連するサービスはどうなりますか？

3. `systemctl` コマンドを使用して `sshd.service` ユニットが実行中かどうかを確認するにはどうしますか？

4. systemdベースのシステムにおいて、システムの初期化中に `sshd.service` ユニットがアクティブになるようにするコマンドは何ですか？

発展演習

1. SysVinitベースのシステムにおいて、`/etc/inittab` で定義されているデフォルトのランレベルは3になっていますが、システムが常にランレベル1で起動します。考えられる原因は何ですか？

2. systemdベースのシステムでは、`/sbin/init` は別の実行可能ファイルへのシンボリックリンクです。`/sbin/init` が指すファイルは何ですか？

3. systemdベースのシステムにおいて、デフォルトのシステムターゲットを確認するにはどうしますか？

4. `shutdown` コマンドでスケジュールされたシステムの再起動をキャンセルするにはどうしますか？

まとめ

このレッスンでは、Linuxディストリビューションがサービスマネージャーとして使用するユーティリティについて説明しました。SysVinit、systemd、およびUpstartユーティリティは、それぞれ独自の方法でシステムサービスとシステム状態を制御します。レッスンでは、次のトピックについて説明しました:

- システムサービスとは何か、また、オペレーティングシステムにおけるの役割。
- SysVinit、systemd、Upstartコマンドの概念と基本的な使用法。
- システムサービスとシステム自体を、適切に起動、停止、再起動する方法。

以下のコマンドと手順を取り上げました:

- `init`、`/etc/inittab`、`telinit` などの、SysVinitに関連するコマンドとファイル。
- systemdの主要コマンド: `systemctl`。
- Upstartのコマンド: `initctl`、`status`、`start`、`stop`。
- `shutdown` などの伝統的な電源管理コマンドと `wall` コマンド。

演習の解答

1. `telinit` コマンドを使用してシステムを再起動するにはどうすればよいですか？

   `telinit 6` コマンドで、ランレベル6に切り替えます。これでシステムが再起動します。

2. システムがランレベル1に入ると、ファイル `/etc/rc1.d/K90network` に関連するサービスはどうなりますか？

   ファイル名の先頭が `K` なので、関連するサービスが停止します。

3. `systemctl` コマンドを使用して `sshd.service` ユニットが実行中かどうかを確認するにはどうしますか？

   `systemctl status sshd.service` または `systemctl is-active sshd.service` コマンドを使用します。

4. systemdベースのシステムにおいて、システムの初期化中に `sshd.service` ユニットがアクティブになるようにするコマンドは何ですか？

   rootになって、`systemctl enable sshd.service` コマンドを実行します。

発展演習の解答

1. SysVinitベースのシステムにおいて、`/etc/inittab` で定義されているデフォルトのランレベルは3になっていますが、システムが常にランレベル1で起動します。考えられる原因は何ですか？

   カーネルのパラメータリストに、`1` ないし `S` が指定されているかもしれません。

2. systemdベースのシステムでは、`/sbin/init` は別の実行可能ファイルへのシンボリックリンクです。`/sbin/init` が指すファイルは何ですか？

   systemdのメインプログラム: `/lib/systemd/systemd`。

3. systemdベースのシステムにおいて、デフォルトのシステムターゲットを確認するにはどうしますか？

   シンボリックリンク `/etc/systemd/system/default.target` は、デフォルトのターゲットを定義するユニットファイルを指します。`systemctl get-default` コマンドも使用できます。

4. `shutdown` コマンドでスケジュールされたシステムの再起動をキャンセルするにはどうすればよいですか？

   `shutdown -c` コマンドを使います。

Linux
Professional
Institute

課題 102: Linuxのインストールとパッケージ管理

102.1 ハードディスクレイアウトを設計する

LPI目標への参照

LPIC-1 v5, Exam 101, Objective 102.1

総重量

2

主な知識分野

- パーティションやディスクを分割し、ファイルシステムやスワップ領域に割り当てる。
- 設計をシステムの意図された利用目的に合わせる。
- /bootパーティションが、ブートに必要なハードウェアアーキテクチャの要件を満たしているか確認する。
- LVMの基本機能に関する知識。

用語とユーティリティ

- /(ルート)ファイルシステム
- /var ファイルシステム
- /home ファイルシステム
- /boot ファイルシステム
- EFIシステムパーティション(ESP)
- swapスペース
- マウントポイント
- パーティション

Linux Professional Institute

102.1 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	102 Linuxのインストールとパッケージ管理
Objective:	102.1 ハードディスクレイアウトの設計
Lesson:	1 of 1

はじめに

このレッスンの目標は、disks、partitions、filesystems、volumes の関係を理解することです。

ディスク（最近のデバイスには “disks” がまったく含まれていないので、storage device と呼ぶことも多い）をデータの “物理的な容器” と考えてください。

ディスクをコンピューターで使用する前に、ディスクをパーティション化する必要があります。パーティションとは、物理ディスクを論理的なサブセットに仕切ったようなものです。パーティショニングは、ディスクに保存されている情報を “区画化” する方法であり、たとえば、オペレーティングシステムデータをユーザーデータから分離します。

すべてのディスクには少なくとも1つのパーティションが必要ですが、必要に応じて複数のパーティションを持つことができ、それらに関する情報はパーティションテーブルに格納されます。このテーブル（表）には、パーティションの最初と最後のセクターとそのタイプ、ならびにパーティションそれぞれの情報が含まれています。

それぞれのパーティションには、ファイルシステムが置かれます。ファイルシステムとは、データを実際にディスクに保存する方法を規定している仕組みです。その中には、ディレクトリの編成方法、ディレクトリ間の関係、各ファイルのデータはどこにあるかなどが含まれます。

パーティションは複数のディスクにまたがることはできません。ただし、Logical Volume Manager（LVM）を使用すると、異なるディスクの複数のパーティションを組み合わせて、ひとつの論理ボリュームとすることができます。

LVMでは、物理デバイスの制限を抽象化し、従来のパーティションよりもはるかに柔軟な方法で組み合わせたり分散したりできる、ディスクスペースの“プール”（ボリュームグループ）を作成し、その中から論理ボリュームを切り出して使用します。LVMは、データをより大きなデバイスに移行することなく、パーティションにスペースを追加する必要がある状況で役立ちます。

このレッスンでは、Linuxシステムのディスクパーティションスキームを設計し、必要に応じてファイルシステムとスワップスペースを、パーティションやディスクに割り当てる方法を学習します。

パーティションとファイルシステムを作成したり管理する方法については、他のレッスンで説明します。このレッスンでLVMの概要を説明しますが、詳細には踏み込みません。

マウントポイント

Linuxでファイルシステムにアクセスするには、ファイルシステムをマウントする必要があります。これは、ファイルシステムをシステムのディレクトリツリー内の**マウントポイント**と呼ばれるあるディレクトリに取り付けることを意味します。

マウントされると、ファイルシステムの内容が、マウントポイントの下に現れて利用できるようになります。たとえば、ユーザーの個人データ（ホームディレクトリ）を含むパーティションがあり、ディレクトリ`/john`、`/jack`、`/carol`が含まれているとします。そのパーティションを`/home`の下にマウントすると、それらのディレクトリの内容が`/home/john`、`/home/jack`、`/home/carol`として利用可能になります。

ファイルシステムをマウントする前に、マウントポイントとなるディレクトリが存在している必要があります。例えば`/mnt/userdata`ディレクトリが存在しない場合は、そこにパーティションをマウントすることはできません。逆にディレクトリが存在していてファイルを含んでいる場合は、ファイルシステムをアンマウントするまで、それらのファイルにはアクセスできません。ディレクトリの内容を一覧表示すると、ディレクトリの元の内容ではなく、マウントされたファイルシステムにあるファイルが表示されます。

ファイルシステムはどこにでもマウントできます。ただし、システム管理を容易にするために従うべき慣習があります。

伝統的に`/mnt`には、取り外し可能な外部デバイスをマウントします。例えば、CD-ROMドライブ（`/mnt/cdrom`）やフロッピーディスク（`/mnt/floppy`）のように、一般的なデバイス用の**アンカーポイント**がその下に用意されています。

最近は（`/mnt`ではなく）`/media`がよく使われます。ユーザーによる取り外し可能なメディア（外部ディスク、USBフラッシュドライブ、メモリカードリーダ、光ディスクなど）を

システムに接続した場合の、デフォルトのマウントポイントになっています。

現在のほとんどのLinuxディストリビューションでは、リムーバブルデバイスを接続すると自動的に `/media/USER/LABEL` 以下にマウントされるようになっています。`USER` はユーザー名で、`LABEL` はデバイスを識別するボリューム名（ラベル）です。例えば、`FlashDrive` というラベルのUSBフラッシュドライブをユーザー `john` が接続すると、`/media/john/FlashDrive/` 以下にマウントされます。（デスクトップ環境によっては挙動が異なることがあります）。

とはいえ、ファイルシステムを 手動で マウントする必要がある場合は、`/mnt` の下にマウントすることがお勧めです。Linuxでファイルシステムのマウントとアンマウントを制御するコマンドについては、別のレッスンで取り上げします。

データを分離する

Linuxでは、専用のパーティションに保持した方が良いディレクトリがいくつかあります。さまざまな理由があり、たとえば、ブートローダー関連のファイル（`/boot`）を、専用のbootパーティション に置けば、ルートファイルシステムが破損した場合でもシステムを起動できます。

あるいは、ユーザー個人用のディレクトリ（`/home`）を、別のパーティションに分けておけば、誤ってユーザーデータに触れるリスクなしに、システムを簡単に再インストールできます。またあるいは、Webやデータベースなどのサーバー用データ（通常は `/var` の下）を別のパーティション（ないし別のディスク）に分けておけば、サーバー用にディスク領域を追加する必要がある場合のシステム管理が容易になります。

パフォーマンス上の理由から、あるディレクトリを別のパーティションに分けることもあります。たとえば、ルートファイルシステム（`/`）を高速なSSDユニットに保持し、`/home` や `/var` などの大きな容量を必要とするディレクトリを安価なハードディスクに保持すれば、パフォーマンスとコストの最適化を図ることができます。

ブートパーティション（`/boot`）

ブートパーティションには、ブートローダーがオペレーティングシステムをロードするために使用するファイルが含まれています。Linuxシステムのブートローダーは、通常はGRUB2であり、古いシステムではGRUB Legacyです。パーティションは通常 `/boot` の下にマウントされ、ブートローダーが参照するファイルは `/boot/grub` に保存されます。

ほとんどの場合、GRUBはルートパーティション（`/`）をマウントして、その `/boot` ディレクトリからファイルをロードできるため、技術的にはブートパーティションは必須ではありません。

しかし、安全性（ルートファイルシステムが破損してもシステムが起動できるようにするため）や、ルートパーティションにブートローダが理解できないファイルシステムを使用したい場合や、サポートされていない暗号化や圧縮方法を使用したい場合には、独立したブート

パーティションが必要になることがあります。

ブートパーティションは通常、ディスク上の最初のパーティションです。歴史的に、オリジナルのIBM PC BIOSで扱える最大ディスク容量が528MB（MS-DOSでは504MB）であったことから生じた慣習です。

IBM PC BIOSでは、シリンダー、ヘッド、セクター の3つの値（CHS）でディスクをアドレス指定しており、それぞれの最大値が1024シリンダー、256ヘッド、63セクターであったことに起因します。後に、別のディスクアドレス指定スキーム（LBA Logical Block Addressing）が登場するまでは、BIOSはディスクの先頭から528MBまでの範囲しか読み込むことができなかったのです。つまり（LBAが登場する以前の）レガシーマシンとの互換性を保つには、ブートパーティションをディスクの先頭に置き、シリンダー1024（528 MB）の前で終了することが必要です。（訳注: LBAが登場したのは1994年ですから、概ね1995年よりも前に製造されたマシンのみが対象になります。昔は色々な制約があり今では必要が無い慣習が残っている、とだけ覚えておけば十分です。）

ブートパーティションには、カーネル-イメージと初期RAMディスク、およびブートローダーに必要なファイルのみが格納されます。大した容量は必要なく、概ね300MBから1GB程度を確保すれば十分でしょう。

EFIシステムパーティション（ESP）

UEIF（Unified Extensible Firmware Interface）を搭載したマシンでは、インストールしたオペレーティングシステムのブートローダーとカーネルイメージを格納するために、EFIシステムパーティション（ESP）を使います。

このパーティションは、FATファイルシステムでフォーマットされています。パーティション分割には（MBRではなく）GUIDパーティションテーブルが使用されていて、`C12A7328-F81F-11D2-BA4B-00A0C93EC93B` というGUID（Globally Unique Identifier）が与えられています。ディスクには（互換性などのために）MBRも存在していて、そのパーティションIDは `0xEF` です。

Microsoft Windowsを実行しているマシンでは、ESPパーティションは通常、ディスク上の最初のパーティションですが、必須ではありません。ESPは、インストール時にオペレーティングシステムによって作成ないし設定され、Linuxシステムでは `/boot/efi` の下にマウントされます。

`/home` パーティション

システムの各ユーザーには個人のファイルや設定を保存するためのホームディレクトリが与えられ、そのほとんどは `/home` の下に置かれます。ホームディレクトリ名はユーザー名と同じであるのが風ですから、ユーザーJohnのディレクトリは `/home/john` になります。

ただし、例外もあり、たとえば rootユーザーのホームディレクトリは `/root` ですし、システムサービスのユーザーのホームディレクトリは別の位置であることががあります。

/home ディレクトリのパーティション（ホームパーティション）のサイズを決定する規則はありません。システムのユーザー数とその使用方法を考慮して決定します。たとえば、Webブラウジングとワードプロセッシングのみを行うユーザーは、ビデオ編集を行うユーザーよりも必要なスペースが少なくて済むでしょう。

可変データ（/var）

このディレクトリには、"可変データ”（variable data）、すなわち、システムログ（/var/log）、一時ファイル（/var/tmp）、アプリケーションデータのキャッシュ（/var/cache）など、システムが稼働中に読み書きするデータが含まれます。

たとえば、/var/www/html はApache Webサーバーのデータファイルを置くディレクトリですし、/var/lib/mysql はMySQLサーバーのデータベースファイルを置きます。ただし、どちらも別の位置に変更できます。

/var を別のパーティションに配置する理由の1つは、安定性です。多くのアプリケーションとプロセスは、/var と /var/log や /var/tmp などのサブディレクトリに書き込みを行います。プロセスの誤動作によって、ファイルシステムに空き領域がなくなるまでデータを書き込む可能性があります。

/var と / が同じファイルシステムにある場合、ルートファイルシステムの空き容量がなくなってしまい、カーネルパニックからファイルシステムが破損して、回復が困難な状況に陥る可能性があります。/var を別のパーティションに分けておけば、ルートファイルシステムは影響を受けません。

/home と同様に、システムの使用方法によって異なるため、/var のパーティションのサイズを決定する普遍的なルールはありません。家庭用のシステムでは数ギガバイトしか必要ないこともありますが、データベースやWebサーバーではもっと多くのスペースが必要になるでしょう。そのようなシナリオでは、ルートパーティションとは異なるパーティションに /var を配置して、ディスク障害に対する予防策を講じておくことがお勧めです。

スワップ

スワップパーティションは、必要に応じてメモリページをRAMからディスクにスワップするために使用されます。このパーティションは、専用のパーティションID（0x82）を持ち、使用する前に mkswap と呼ばれるユーティリティでセットアップする必要があります。

スワップパーティションは他のパーティションのようにマウントできません。つまり、通常のディレクトリのようにアクセスしてその内容を確認することはできません。

システムは（一般的ではありませんが）複数のスワップパーティションを持つことができ、Linuxではパーティションではなくスワップ ファイル の使用もサポートします。スワップファイルは、必要な時にスワップ領域をすばやく増やすのに役立ちます。

スワップパーティションのサイズは論争の的となる課題です。Linuxの初期の古いルール（“RAM容量の2倍”）は、システムの使用方法と物理RAMの量によって不適切な場合があります。

Red Hat Enterprise Linux 7のドキュメントでは、Red Hatは以下を推奨しています。

RAMの量	**推奨スワップサイズ**	**ハイバネーションする場合の推奨スワップサイズ**
RAMが2GB未満	RAM容量の2倍	RAM容量の3倍
RAM2-8GB	RAMと同容量	RAM容量の2倍
RAM8-64GB	少なくとも4G	RAM容量の1.5倍
RAMが64GB以上	少なくとも4G	推奨されない

もちろん、スワップ容量はワークロードに依存します。データベース、web、SAPサーバーなどの重要なサービスを実行しているマシンでは、これらのサービス（またはソフトウェアベンダー）のドキュメントで推奨事項を確認することを勧めます。

NOTE スワップパーティションとスワップファイルの作成と有効化については、LPIC-1のObjective 104.1を参照してください。

LVM

ディスクを1つまたは複数のパーティションに分割し、各パーティションにファイルと関連するメタデータを格納するファイルシステムを格納する方法についてはすでに説明しました。パーティション分割の欠点は、システム管理者がデバイスの空きディスク領域をどのように配分するかを事前に決定しなければならないことです。パーティションが当初の見込みよりも多くの領域を必要とする場合、後で問題になることがあります。もちろんパーティションのサイズは変更できますが、ディスク上に空き容量がない場合など、容量の拡大が不可能な場合があります。

LVM（Logical Volume Management）は、従来のパーティショニングよりも柔軟なディスク領域管理方法を提供する、一種のストレージ仮想化です。LVMの目標は、エンドユーザーのストレージニーズへの対応を容易にすることです。最も基本となる要素は　物理ボリューム（PhysicalVolume）といって、ディスクパーティションやRAIDアレイなどのブロックデバイスです。

いくつかの物理ボリュームを　ボリュームグループ（VolumeGroups）にグループ化して、それぞれのPV容量を合計したサイズのストレージとして抽象化した1つの論理デバイスとします。

ボリュームグループを作成する際に、それぞれの物理ボリュームが、エクステント　　と呼ばれる固定サイズ（デフォールトは4M）のピースに分割されます。物理ボリューム上のエクステントは 物理エクステント（Phisical Extents）と呼ばれ、後述する論理ボリューム上のエクステントは 論理エクステント（Logical Extents）と呼ばれます。通常、論理エクス

テントと物理エクステントは1対1でマップされますが、ディスクミラーリングなどの機能を使用する場合には、対応付けが変更されることがあります。

ボリュームグループから指定した数のエクステントを切り出して、論理ボリューム（Logical Volume）を作成します。論理ボリュームは、パーティションと同様に（ブロックデバイスとして）機能しますが、より柔軟性があります。

論理ボリュームのサイズは作成時に指定しますが、実際には物理エクステントのサイズ（デフォルトでは4MB）に、論理ボリューム上のエクステントの数を掛けたものです。つまり、論理ボリュームを拡張するには、ボリュームグループから未使用のエクステントを論理ボリュームに追加するだけです。同様に、エクステントを削除して論理ボリュームを縮小することもできます。

論理ボリュームが作成されると、オペレーティングシステムからは通常のブロックデバイスとして認識されます。デバイスは `/dev` に作成されます。名前は `/dev/VGNAME/LVNAME` です。ここで言う、`VGNAME` はボリュームグループの、`LVNAME` は論理ボリュームの名前です。

これらのデバイスは、標準的なユーティリティ（たとえば、`mkfs.ext4` など）を使って希望するファイルシステム種別でフォーマットし、通常の方法でマウントできます。すなわち、`mount` コマンドで手動で、あるいは `/etc/fstab` ファイルに追加することで自動的にマウントできます。

演習

1. Linuxシステムでは、GRUBブートローダーのファイルはどこに置かれますか？

2. 骨董PC（1994年以前の製造）が常にカーネルをロードできるようにするには、ブートパーティションをどこに配置する必要がありますか？

3. EFIパーティションは、通常どこにマウントされますか？

4. ファイルシステムを手動でマウントする場合、どのディレクトリにマウントするのが（慣例上）適当ですか？

発展演習

1. ボリュームグループの最小要素は何ですか？

2. 論理ボリュームの実際のサイズはどう定義されますか？

3. MBRパーティションスキームで、EFIシステムパーティションのIDは何ですか？

4. スワップパーティションに加えて、Linuxシステムのスワップ領域をすばやく増やすにはどうすればよいですか？

まとめ

このレッスンでは、パーティション分割について学習しました。どのディレクトリに専用のパーティション割り当てるべきかと、その理由を学びました。また、LVM（Logical Volume Management）の概要と、従来のパーティショニングと比較してデータとディスク容量をより柔軟に割り当てる仕組みについても説明しました。

次のファイル、用語、およびユーティリティについて説明しました。

/
Linuxルートファイルシステム。

/var
“可変データ” のディレクトリ。時間の経過とともにサイズが変化するデータ。

/home
一般ユーザーのホームディレクトリの標準的な親ディレクトリ。

/boot
ブートローダーが利用するファイル、Linuxカーネル、および初期RAMディスクを格納するディレクトリ。

EFIシステムパーティション（ESP）
UEFIを搭載したシステムにおける、ブートファイルを保存するために使用される。

スワップ領域
実装容量よりも多くのRAMが使用される場合に、メモリページをスワップアウトするために使用される。

マウントポイント
デバイス（ハードディスクなど）がマウントされるディレクトリ。

パーティション
ハードディスクの分割領域。

演習の解答

1. Linuxシステムでは、GRUBブートローダーのファイルはどこに置かれますか？

   `/boot/grub` の中。

2. 骨董PC（1994年以前の製造）が常にカーネルをロードできるようにするには、ブートパーティションをどこに配置する必要がありますか？

   シリンダー1024の前。

3. EFIパーティションは、通常どこにマウントされますか？

   `/boot/efi`。

4. ファイルシステムを手動でマウントする場合、どのディレクトリにマウントするのが（慣例上）適当ですか？

   `/mnt` の下。パーティションは任意のディレクトリにマウントできますので、あくまでも「慣例」です。

発展演習の解答

1. ボリュームグループ内の最小単位は何ですか

   エクステント。

2. 論理ボリュームの実際のサイズはどう定義されますか？

   物理エクステントのサイズに、論理ボリューム内のエクステントの数を掛けたもの。

3. MBRパーティションスキームで、EFIシステムパーティションのIDは何ですか？

   IDは `0xEF` です。

4. スワップパーティションに加えて、Linuxシステムのスワップスペースをすばやく増やすにはどうすればよいですか？

   スワップ用のファイルを作成して登録する。

102.2 ブートマネージャをインストールする

LPI目標への参照

LPIC-1 v5, Exam 101, Objective 102.2

総重量

2

主な知識分野

- 代替ブート場所とバックアップブートオプションを提供する。
- GRUB Legacyなどのブートローダをインストールして設定する。
- GRUB 2の基本的な設定・変更を実行する。
- ブートローダーと対話する。

用語とユーティリティ

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

102.2 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	102 Linuxのインストールとパッケージ管理
Objective:	102.2 ブートマネージャーをインストールする
Lesson:	1 of 1

はじめに

コンピュータの電源がオンになった時に、最初に実行するソフトウェアはブートローダーです。これは、オペレーティングシステムのカーネルをロードして、制御を渡すことを唯一の目的とするコードです。カーネルは必要なドライバーをロードし、ハードウェアを初期化してから、オペレーティングシステムの残りの部分をロードします。

GRUBは、ほとんどのLinuxディストリビューションで使用されているブートローダーです。LinuxカーネルやWindowsなどの他のオペレーティングシステムをロードすることができ、複数のカーネルイメージとそのパラメータを、別々のメニューエントリとして扱えます。起動するカーネルをキーボードから選択することができ、起動オプションやパラメータを編集するためのコマンドラインインターフェイスも使用できます。

ほとんどのLinuxディストリビューションは、インストール時にGRUB（実際にはGRUB　2）を自動的に構成するので、一般ユーザーがそれについて考える必要はありません。しかしシステム管理者は、たとえばカーネルのアップグレードに失敗した後に、ブート障害からシステムを回復できるように、ブートプロセスを制御する方法に対する知識が必要です。

このレッスンでは、GRUBをインストール、設定、操作する方法を学習します。

GRUB Legacy 対 GRUB 2

今では GRUB Legacy と呼ばれているオリジナルのGRUB（Grand Unified Bootloader）は、1995年にGNU Hurdプロジェクトの一部として開発されました。多くのLinuxディストリビューションが、それまでのLILOなどのブートローダーに代わるデフォルトのブートローダーとしてGRUBUを採用しました。

GRUB 2は、よりクリーンで安全かつ堅牢で強力であることを目指して、GRUBを完全に書き直したものです。GRUB Legacyと比較すると、柔軟な設定ファイル（スクリプト言語のように多くのコマンドや条件文が使える）、モジュール化された設計、優れたローカライズ/国際化など、多くの利点があります。

スプラッシュ画面を備えたテーマとグラフィカルなブートメニュー、ハードディスクにあるLiveCD ISOイメージの直接起動、非x86アーキテクチャのサポート、UUIDのユニバーサルサポート（ディスクとパーティションの選択を容易にする）など、他にもさまざまな利点があります。

GRUB Legacyの開発は既に終了しており（最後のリリースは2005年の0.97でした）、ほとんどの主要なLinuxディストリビューションはデフォルトのブートローダーとしてGRUB 2をインストールします。ただし、GRUB Legacyを使用しているシステムに出会う可能性もまだあるので、GRUB Legacyの使用方法と、GRUB 2との違いを理解しておくことが重要です。（訳注: 2010年代前半までにセットアップされたマシンは、GRUB Legacyを使用している可能性が高いです。）

ブートローダーはどこにあるか？

歴史的に、IBM PC互換システムのハードディスクは、1982年にIBM PC-DOS（MS-DOS）2.0用に作成されたMBRパーティションスキームを使用してパーティション分割されていました。

この方式では、ディスクの先頭セクターの512バイトをMBR（Master Boot Record）と呼び、そこにディスクのパーティション情報を記述したテーブル（パーティションテーブル）と、一次ブートローダーと呼ばれる起動用コードが置かれています。

コンピュータの電源を入れると、この非常に小さな（サイズ制限のため）起動用コードが読み取られて起動し、ディスク上の二次ブートローダ（通常はMBRと最初のパーティションの間の32KBスペースに位置する）に制御を渡し、その二次ブートローダーがオペレーティングシステムをロードします。

MBRパーティションディスクでは、GRUBのブートコード（一次ブートローダー）がMBRにインストールされます。これが、MBRと最初のパーティションの間にインストールされた “コア” イメージ（二次ブートローダー）をロードし、制御を渡します。この時点から、GRUBは残りの必要なリソース（メニュー定義、構成ファイル、追加モジュールなど）をディスクからロードできるようになります。

ただし、MBRには、パーティション数（初期には最大4つのプライマリパーティション、後

に最大3つのプライマリパーティションと1つの拡張パーティションならびに複数の論理パーティション）と、ディスクサイズ（最大2TB）の制限があります。これらの制限を克服するために、UEFI（ Unified Extensible Firmware Interface ）規格の一部であるGPT（ GUID Partition Table ）と呼ばれる新しいパーティション方式が作成されました。

GPTパーティションディスクは、PC BIOSを搭載した古いコンピューターでも、UEFIファームウェアを搭載したコンピューターのいずれでも使用できます。BIOSを搭載したマシンでは、特別なBIOSブートパーティションにGRUBの後半部分が置かれます。

UEFIファームウェアを搭載したシステムでは、ESP（EFI System Partition）と呼ばれるパーティションから、GRUBファイル `grubia32.efi`（32ビットシステム）ないし `grubx64.efi`（64ビットシステム）が、ファームウェアによってロードされます。

`/boot` パーティション

BIOSを搭載したマシンにLinuxをインストールする場合、ブートプロセスに必要なファイルを（独立した）ブートパーティションに保存し、それをルートファイルシステム直下の `/boot` ディレクトリにマウントするのが一般的でした。

GRUBのような高機能なブートローダは、ルートファイルシステムをマウントして `/boot` ディレクトリにアクセスすることができるので、現在のシステムではブートパーティションは必要ありません。しかし、ブートに必要なファイルを独立したファイルシステムに置いておくのは、（障害への対策として）良い習慣と言えます。

ブートパーティションを作成する場合には、ディスク上の最初のパーティションとするのが慣例です。オリジナルのIBM PC BIOSでは、ディスクの先頭528MB（MS-DOSでは504MB）までしかアクセスできなかったことから、このテクニックが生まれました。（訳注: 詳細は前のレッスンを参照してください。）

つまり、骨董PC（1996年以前に製造のもの）との互換性を確保するためには、`/boot` パーティションをディスクの先頭からシリンダー1024（528 MB）の間に配置し、BIOSがブートローダーをロードできるようにします。（訳注: BIOSによるブートファイルを置く物理範囲の制約は、ハードディスクインターフェイス規格改定により何度か変わっています。1996年のATA-1996規格以降、このような制約はほぼ無くなりました。もちろん、現在のマシンではブートパーティションをどこに置いても構いません）。現在のLinuxにおけるブートパーティションの推奨サイズは、概ね300MB以上です。

`/boot` を独立パーティションに置くその他の理由として、GRUB 2がサポートしていない暗号化やディスク圧縮を使用したり、あるいはルートファイルシステムを特殊なファイルシステムでフォーマットする場合などが挙げられます。

ブートパーティションの内容

`/boot` パーティションの内容は、システムアーキテクチャや使用中のブートローダーによって異なりますが、x86ベースのシステムでは通常以下のファイルがあるのが普通です。こ

れらのほとんどは、名前に `-VERSION` サフィックスが付けられています。ここで `-VERSION` は、Linuxカーネルのバージョンです。たとえば、Linuxカーネルバージョン `4.15.0-65-generic` の構成ファイルは `config-4.15.0-65-generic` になります。

構成ファイル

このファイルの名前は、通常 `config-VERSION`（上記の例を参照）です。Linuxカーネルの構成パラメータを格納しています。このファイルは、新しいカーネルをコンパイル、ないしインストールするときに自動的に生成されます。ユーザーが直接変更することはできません。

システムマップ

このファイルは、カーネルのシンボル名（変数や関数など）と、そのメモリ内の位置（アドレス）を対応付ける参照表です。カーネルパニック という重大なシステム障害をデバッグするときなどに参照し、障害が発生したときに参照していた変数や関数を知ることができます。設定ファイルと同様に、ファイルの名前は通常 `System.map-VERSION` です（例: `System.map-4.15.0-65-generic`）。

Linuxカーネル

これはオペレーティングシステムのカーネルそのものです。名前は通常 `vmlinux-VERSION` です（例: `vmlinux-4.15.0-65-generic`）。`vmlinux` ではなく `vmlinuz` という名前のこともあります。末尾の `z` は、ファイルが圧縮されていることを示しています。

初期RAMディスク

この名前は通常 `initrd.img-VERSION` です。RAMディスクにロードする、必要最小限のルートファイルシステムが収められています。その中には、カーネルが実際のルートファイルシステムをマウントするために必要なユーティリティとカーネルモジュールが含まれています。

ブートローダー関連ファイル

GRUBがインストールされているシステムでは、これらは通常 `/boot/grub` にあり、GRUB構成ファイル（GRUB 2の場合は `/boot/grub/grub.cfg`、GRUB Legacyの場合は `/boot/grub/menu.lst`）、モジュール（`/boot/grub/i386-pc/` など）、翻訳ファイル（`/boot/grub/locale/`）とフォント（`/boot/grub/fonts/`）が含まれています。

GRUB 2

GRUB 2のインストール

GRUB 2をインストールするには `grub-install` ユーティリティを使用します。システムが起動しない場合には、Live CDやレスキューディスクからシステムを起動し、本来の起動パーティションを見つけてマウントしてから、このユーティリティを実行します。

NOTE

以下のコマンドは、root権限でログインしていることを前提としています。一般ユーザーとしてログインしている場合は、まず `sudo su -` を実行

> しrootに“昇格”します。作業が終了したら、`exit` と入力してログアウトすると、通常のユーザーに戻ります。

システムの最初のディスクが ブートデバイス であることが普通です。ディスクに ブートパーティションが があるか調べるには、`fdisk` ユーティリティを使用します。マシンの最初のディスク上のすべてのパーティションを一覧表示するには、次のコマンドを実行します:

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

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

ブートパーティションには、ブート列のに `*` が表示されます。上の例では、`/dev/sda1` です。

（訳注: GPTパーティションのディスクでは、fdiskコマンドは原則として使用できません。代わりに partedコマンドを使用します。）

次に `/mnt` の下に一時ディレクトリを作成して、その下にパーティションをマウントします。

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

次に、ブート デバイス（パーティションでは ない）とブートパーティションがマウントされているディレクトリを指定して、`grub-install` を実行します。システムに専用のブートパーティションがある場合、コマンドは次のとおりです。

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

（訳注: MBRパーティションのディスクの場合のみです。GPTパーティション（UEFIシステム）の場合もgrub-installコマンドを使いますが、ブートローダーのインストール先をオ

プションで指定します。UEFIシステムでのブートプロセスはディストリビューションによって異なりますので、それぞれのディストリビューションのドキュメントを参照してください。）

ブートパーティションがなく、ルートファイルシステムの `/boot` ディレクトリだけがあるシステムにインストールする場合は、そのディレクトリを `grub-install` に指定します。次のようになります:

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

GRUB 2の設定

GRUB 2のデフォルトの設定ファイルは `/boot/grub/grub.cfg` です。このファイルは自動的に生成されるので、手動で編集することはお勧めしません。GRUBの設定を変更するには、ファイル `/etc/default/grub` を編集してから、`update-grub` コマンドを実行して設定ファイルを再生成する必要があります。

NOTE `update-grub` は通常、`grub-mkconfig -o /boot/grub/grub.cfg` へのショートカットであるため、同じ結果になります。

ファイル `/etc/default/grub` には、起動するデフォルトカーネルの指定や、タイムアウト時間の設定、追加のコマンドラインパラメータなど、GRUB 2の動作を制御するオプションがあります。最も重要なものを示します:

`GRUB_DEFAULT=`
: 起動するデフォルトのメニューエントリを指定します。エントリを指定する数値（`0`、`1` など）、エントリの名前（`debian` など）、`saved` といった値を取り、次に説明する `GRUB_SAVEDEFAULT=` と組み合わせて使用します。メニューエントリはゼロから始まるので、最初のメニューエントリは `0`、2番目は `1` になることに注意してください。

`GRUB_SAVEDEFAULT=`
: このオプションが `true` で、`GRUB_DEFAULT=` が `saved` の場合、最後に選択したメニューエントリがデフォルトになります。

`GRUB_TIMEOUT=`
: デフォルトのメニューエントリがタイムアウトで起動されるまでの時間（秒単位）。`0` に設定すると、メニューを表示せずにデフォルトのエントリを起動します。`-1` に設定すると、ユーザーがエントリを選択するまで待機し続けます。

`GRUB_CMDLINE_LINUX=`
: Linuxカーネルのエントリに追加するコマンドラインオプションを指定します。

GRUB_CMDLINE_LINUX_DEFAULT=

デフォルトでは、Linuxカーネルごとに2つのメニューエントリが生成されます。1つはデフォルトのオプションで、もう1つは保守用のエントリです。このオプションは、デフォルトのエントリにのみ追加するパラメータを指定します。

GRUB_ENABLE_CRYPTODISK=

`y` を指定すると、`grub-mkconfig`、`update-grub`、`grub-install` などのコマンドが、暗号化されたディスクを探して、起動時にそれらにアクセスするために必要なコマンドを追加します。これにより、ディスクにアクセスするためのパスフレーズが必要になるので、自動起動（`GRUB_TIMEOUT=` が `-1` 以外）が無効になります。

メニューエントリの管理

`update-grub` を実行すると、GRUB 2はマシン上のカーネルとオペレーティングシステムをスキャンして、対応するメニューエントリを `/boot/grub/grub.cfg` ファイルに生成します。独自のエントリを追加したい場合には、`/etc/grub.d` ディレクトリ内にスクリプトファイルを追加します。

それらのスクリプトファイルは実行可能である必要があり、`update-grub` によって番号順に処理されます。たとえば、`05_debian_theme` は `10_linux` の前に処理されます。独自のメニューエントリは、`40_custom` ファイルに追加するのが一般的です。

メニューエントリの基本的な構文を以下に示します:

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

最初の行は常に `menuentry` で始まり、`{` で終わります。引用符で囲まれたテキストが、GRUB 2ブートメニューのエントリラベルとして表示されます。

`set root` パラメータには、ルートファイルシステムが配置されているディスクとパーティションを定義します。GRUB 2では、ディスクにはゼロから番号が付けられるため、`hd0` が最初のディスク（Linuxでは `sda`）、`hd1` が2番目のディスクというようになります。また、パーティションには1から始まる番号が付けられます。上の例では、ルートファイルシステムは最初のディスク（`hd0`）の最初のパーティション（`,1`）、つまり `sda1` になります。

デバイスとパーティションを番号で指定するのではなく、ラベルやUUID（Universally Unique Identifier）を指定して、ファイルシステムを検索させることもできます。メニューエントリの `search --set=root` に続けて、`--label` とファイルシステムラベルを指定するか、`--fs-uuid` とファイルシステムのUUIDを指定します。

ファイルシステムのUUIDを見つけるには、次のようにします:

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

この例では、`/dev/sda1` のUUIDは `ae71b214-0aec-48e8-80b2-090b6986b625` です。GRUB 2でルートデバイスとして指定する場合は、`search --set=root --fs-uuid ae71b214-0aec-48e8-80b2-090b6986b625` という行をメニューエントリに置くことになります。

`search` を使用する場合には、GRUBがフロッピーディスクの検索に時間を浪費しないように、`--no-floppy` パラメータを追加するのが一般的です。

`linux` 行では、カーネルが置かれている位置を指定します（例では、ファイルシステムのルートにある `vmlinuz` ファイル）。その後に、カーネルのコマンドラインパラメータを置くことができます。

例では、ルートパーティション（`root=/dev/sda1`）を指定し、3つのカーネルパラメータを渡しています。ルートパーティションを読み取り専用でマウントし（`ro`）、ほとんどの起動時メッセージを表示せず（`quiet`）、スプラッシュ画面を表示します（`splash`）。

`initrd` 行では、初期RAMディスクが置かれている位置を指定します。例では、ファイルシステムのルートにある `initrd.img` ファイルです。

NOTE ほとんどのLinuxディストリビューションでは、カーネルとinitrdをルートファイルシステムのルートディレクトリに置きません。通常は、`/boot` ディレクトリないしパーティションに実際のファイルが置かれます。

メニューエントリの最後の行は、文字 `}` のみである必要があります。

GRUB 2との対話

システムを起動すると、GRUB 2のメニューが表示されます。矢印キーを使用してメニューエントリのひとつを選択し、`Enter` で起動します。

TIP カウントダウンだけが表示されて、メニューが表示されない場合は、`Shift` を押すとメニューが表示されます。

メニューエントリで指示したオプションを編集するには、矢印キーでオプションを選択してから `E` を押します。すると、`/boot/grub/grub.cfg` で定義されている `menuentry` の内容を含むエディターウィンドウが表示されます。

オプションを編集したら、`Ctrl`+`X`ないし`F10`を押して起動するか、`Esc` を押してメニューに戻ります。

GRUB 2シェルに入るには、メニュー画面で `C`（編集ウィンドウでは `Ctrl`+`C`）を押します。次のようなコマンドプロンプトが表示されます: `grub>`

`help` と入力して、すべてのコマンドのリストを表示するか、`Esc`を押してシェルを終了してメニュー画面に戻ります。

NOTE `/etc/default/grub` で `GRUB_TIMEOUT` を `0` に設定した場合には、このメニューが表示されないことに注意してください。

GRUB 2シェルからのブート

メニューエントリの設定ミスによりシステムが起動しない場合は、GRUB 2シェルを使用してシステムを起動できます。

まず、ブートパーティションを見つけます。（GRUBをシェルの）`ls` コマンドを使用すると、GRUB2が検出したディスクとパーティションの一覧を表示します。

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

上の例はシンプルで、ディスク `(hd0)` が1つ、そのパーティションも1つだけです: `(hd0,msdos1)`。

もちろん、表示されるディスクとパーティションはシステムによって異なります。この例では、ディスクがMBRパーティションで分割されているため、`hd0` の最初のパーティションは `msdos1` になります。GPTパーティションで分割されている時は `gpt1` になります。

Linuxを起動するには、カーネルと初期RAMディスク（initrd）が必要です。`(hd0,msdos1)` の内容を確認しましょう。（訳注: 末尾の `/` に注意してください）。

```
grub> ls (hd0,msdos1)/
lost+found/ swapfile etc/ media/ bin/ boot/ dev/ home/ lib/ lib64/ mnt/ opt/ proc/
root/ run/ sbin/ srv/ sys/ tmp/ usr/ var/ initrd.img initrd.img.old vmlinuz cdrom/
```

Linuxのlsコマンドと同様に、`-l` オプションを追加して、長いリストを取得できます（訳注:GRUB2の構成によってはオプションが使えないことがあります）。`Tab`を入力すること

で、ディスク、パーティション、ファイルの名前を補完することもできます。。

この例では、パーティションのルートディレクトリにカーネル（`vmlinuz`）とinitrd（`initrd.img`）があることに注意してください。無い場合には、`list (hd0,msdos1)/boot/` で、`/boot` の内容を確認するとよいでしょう。

次に、ブートパーティションを設定します。

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

`linux` コマンドで、Linuxカーネルをロードすることを指示します。カーネルのパスと、カーネルにルートファイルシステムの位置を知らせる `root=` オプションを指定します。

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

初期RAMディスク `initrd` をロードすることを指示します。initrdコマンドに、`initrd.img` ファイルへのフルパスを指定します。

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

最後に、`boot` を使用してシステムを起動します。

レスキューシェルからのブート

起動に失敗した場合、GRUB 2は簡易版のレスキューシェルをロードすることがあります。コマンドプロンプトが、`grub rescue>` になるので分かります。

このシェルからシステムを起動する手順は、前に示したものとほぼ同じです。ただし、動作させるには、いくつかのGRUB 2モジュールをロードする必要があります。

（前の例のように `ls` を使って）ブートパーティションを見つけたら、`set prefix=` コマンドで、GRUB 2のモジュールファイルを含むディレクトリへのフルパスを指示します。通常は `/boot/grub` です。取り上げている例では、次のようになります:

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

次に、`insmod` コマンドを使って、モジュール `normal` と `linux` をロードします。

```
grub rescue> insmod normal
```

```
grub rescue> insmod linux
```

続いて、前の例と同様に、カーネルのパスと `set root=` でブートパーティションを指定した、`linux` コマンドと、初期RAMディスクのパスを指定する `initrd` コマンドを実行し、最後に `boot` コマンドでブートを試みます。

GRUB レガシー

稼働中のシステムにGRUBレガシーをインストールする

稼働中のシステムのディスクにGRUBレガシーをインストールするには、（GRUB 2と同様に）`grub-install` ユーティリティを使用します。基本的なコマンドは `grub-install DEVICE` です。ここで、`DEVICE` にはGRUBレガシーをインストールするディスクを指定します。次の例では `/dev/sda` です。

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

`/dev/sda1` のようなパーティションではなく `/dev/sda/` のように、GRUBレガシーをインストールする デバイス を指定することに注意してください。

GRUBのデフォルトでは、指定されたデバイスの `/boot` ディレクトリに、必要なファイルをコピーします。別のディレクトリにコピーする場合は、`--boot-directory=` オプションに、ファイルのコピー先へのフルパスを指定します。

GRUBシェルからGRUBレガシーをインストールする

何らかの理由でシステムを起動できず、GRUBレガシーを再インストールする必要がある場合は、GRUBレガシーブートディスクのGRUBシェルから再インストールできます。

GRUBシェル（ブートメニューで `c` を入力して `grub>` プロンプトが表示されている状態）からの最初の手順は、`/boot` ディレクトリを含むブートデバイスを設定することです。たとえば、`/boot` ディレクトリが、1番目のディスクの最初のパーティションにある場合、コマンドは次のようになります。

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

`/boot` ディレクトリを含んでいるデバイスがわからない場合は、次のように `find` コマンドを使用してGRUBに検索させることができます。

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

デバイスが判明したら、前述のように（rootコマンドで）ブートパーティションを指定して、 `setup` コマンドでGRUBレガシーをMBRにインストールすると共に、必要なファイルをディスクにコピーします。

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

これで、システムを再起動すれば、正常に起動するはずです。

GRUBレガシーとそのメニューエントリを設定する

GRUBレガシーでは、`/boot/grub/menu.lst` ファイルに、設定とメニューリストが保存されます。これは、コマンドとパラメータのリストを含む単純なテキストファイルであり、好みのテキストエディターで直接編集できます。

`#` で始まる行はコメントと見なされ、空白行は無視されます。

それぞれのメニューエントリには、少なくとも3つのコマンドがあります。最初の `title` は、メニュー画面に表示するオペレーティングシステムの名称などを指定します。2番目の `root` は、GRUBレガシーが起動するデバイスないしパーティションを指示します。

3番目のエントリ `kernel` には、そのエントリが選択されたときにロードするカーネルイメージへのフルパスを指定します。このパスは、`root` パラメータに指定したデバイスのルートディレクトリからの相対パスであることに注意してください。

簡単な例を次に示します。

```
# This line is a comment
title My Linux Distribution
root (hd0,0)
kernel /vmlinuz root=/dev/hda1
```

GRUB 2とは異なり、GRUBレガシーでは、ディスクとパーティションの両方を、ゼロからの番号で数えます。したがって、`root(hd0,0)` コマンドでは、最初のディスク（`hd0`）の最初のパーティション（`0`）が、ブートパーティションになります。

`kernel` コマンドに、ブートデバイスとカーネルパスの両方を続けて指定することができ、その場合には `root` ステートメントを省略できます。次のようになります。

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

これは以下の例と同等です:

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

どちらも、最初のディスクの最初のパーティション（`hd0,0`）の、ルートディレクトリ（`/`）から `vmlinuz` ファイルをロードします。

`kernel` コマンドのパラメータ `root=/dev/hda1` では、どのパーティションをルートファイルシステムとして使用するかを、Linuxカーネルに指示しています。これはLinuxカーネルのパラメータであり、GRUBレガシーのコマンドではありません。

NOTE

カーネルパラメータの詳細については、https://www.kernel.org/doc/html/VERSION/admin-guide/kernel-parameters.html を参照してください。（VERSIONはカーネルのバージョン番号です。）

多くの場合、`initrd` コマンドで初期RAMディスクイメージの位置を指定する必要があります。ファイルのフルパス指定は `kernel` コマンドと同様で、パスの前にデバイスやパーティションを指定することもできます。例:

```
# This line is a comment
title My Linux Distribution
root (hd0,0)
kernel /vmlinuz root=/dev/hda1
initrd /initrd.img
```

GRUBレガシーはモジュラー設計になっていて、追加のモジュールをロードして、特殊なハードウェアやファイルシステム、新しい圧縮アルゴリズムへの対応などを追加できます。なお、モジュールは（一般的なPCの場合）`/boot/grub/i386-pc` ディレクトリある `.mod` ファイルです。

モジュールを追加ロードするには、`module` コマンドに続けて、`.mod` ファイルへのフルパスを指定します。kernelやinitrdと同様に、`root` コマンドで指定されたデバイスからの相対パスで指定することに注意してください。

以下は、Intel 800または900シリーズのビデオチップセット（2004年発表）を搭載したシステムで、フレームバッファの解像度を正しく設定するためのモジュール `915resolution` をロードする例です。

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

他のオペレーティングシステムのチェーンロード

GRUBレガシーには、WindowsなどのGRUBレガシーがサポートしていないオペレーティングシステムをロードするための、チェーンロード　　と呼ばれる機能を備えています。起動したGRUBレガシーが、目的のオペレーティングシステムのブートローダーをロードして実行します。

Windowsをチェーンロードするため例を示します:

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

コマンドを順に見ていきましょう。Linuxの場合と同様に、`root(hd0,1)`　にロードしたいオペレーティングシステムのブートローダーがあるデバイスとパーティションを指定します。この例では、最初のディスクの 2番目の パーティションです。

`makeactive`
: アクティブなパーティションであることを示すフラグをセットします。MBRのプライマリパーティションでのみ有効です。

`chainload +1`
: GRUBにブートパーティションの最初のセクターをロードすることを指示します。そこには通常、ブートローダーが置かれています。

`boot`
: （ロードした）ブートローダーを実行して、対応するオペレーティングシステムをロードします。

演習

1. GRUB 2のデフォルトの設定ファイルはどこにありますか？

2. GRUB 2の設定を変更するにはどうしますか？

3. GRUB 2に独自のメニューエントリを追加するには、どのファイルを変更しますか？

4. GRUBレガシーのメニューエントリはどのファイルに保存されていますか？

5. GRUB 2ないしGRUBレガシーのメニュー画面から、GRUBシェルに入るにはどうしますか？

発展演習

1. GRUBレガシーで、1番目のディスクの2番目のパーティションから起動するように、次のようなカスタムメニューエントリを書きました。

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

   しかし、システムが起動しません。なにが問題ですか？

2. MBRパーティションのデバイス `/dev/sda` には複数のパーティションがあります。システムのブートパーティションを見つけるにはどうしますか？

3. パーティションのUUIDを調べるにはどうしますか？

4. GRUB 2で、次のエントリがあります:

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

   システムが、UUIDが `5dda0af3-c995-481a-a6f3-46dcd3b6998d` のディスクから起動するように変更してください。

5. GRUB 2を、デフォルトのメニューエントリを起動する前に10秒間待機するように設定するにはどうしますか？

6. GRUBレガシーシェルから、2番目のディスクの最初のパーティションにGRUBをインストールするにはどうしますか？

まとめ

このレッスンでは、以下の事柄を学びました:

- ブートローダーとは何か。
- GRUBレガシーとGRUB 2の違い。
- ブートパーティションとは何で、そこに何が含まれるか。
- GRUBレガシーとGRUB 2をインストールする方法。
- GRUBレガシーとGRUB 2を設定する方法。
- GRUBレガシーとGRUB 2にカスタムメニューエントリを追加する方法。
- GRUBレガシーとGRUB 2のメニュー画面とシェルを操作する方法。
- GRUBレガシーないしGRUB　2のシェル、ないしはレスキューシェルからシステムを起動する方法。

このレッスンでは、次のコマンドを説明しました:

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

演習の解答

1. GRUB 2のデフォルトの設定ファイルはどこにありますか？

   `/boot/grub/grub.cfg`

2. GRUB 2の設定を変更するにはどうしますか？

   `/etc/default/grub` ファイルを変更してから、`update-grub` で設定を更新します。

3. GRUB 2に独自のメニューエントリを追加するには、どのファイルを変更しますか？

   `/etc/grub.d/40_custom`

4. GRUBレガシーのメニューエントリはどのファイルに保存されていますか？

   `/boot/grub/menu.lst`

5. GRUB 2ないしGRUBレガシーのメニュー画面から、GRUBシェルに入るにはどうしますか？

   メニュー画面で `c` を押します。

発展演習の解答

1. GRUBレガシーで、1番目のディスクの2番目のパーティションから起動するように、次のようなカスタムメニューエントリを書きました。

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

   しかし、システムが起動しません。なにが問題ですか？

   ブートパーティションの指定方法が間違っています。GRUBレガシーでは、パーティションを　ゼロ　からカウントすることに注意してください（GRUB　2では1からです）。したがって、1番目のディスクの2番目のパーティションは、`root(hd0,1)` になります。

2. MBRパーティションのデバイス　`/dev/sda`　には複数のパーティションがあります。システムのブートパーティションを見つけるにはどうしますか？

   `fdisk　-l　/dev/sda`　を使用します。ブートパーティションには、アスタリスク（`*`）が付けられます。

3. パーティションのUUIDを調べるにはどうしますか？

   `ls -la /dev/disk/by-uuid/` を使用して、パーティションを指すUUIDを探します。

4. GRUB 2で、次のエントリがあります:

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

   システムが、UUIDが　`5dda0af3-c995-481a-a6f3-46dcd3b6998d`　のディスクから起動するように変更してください。

   `set　　root`　　行を変更します。ディスクとパーティションを指定するのではなく、目的のUUIDを持つパーティションを探すように指示します。

```
menuentry "Default OS" {
    search --set=root --fs-uuid 5dda0af3-c995-481a-a6f3-46dcd3b6998d
```

```
    linux /vmlinuz root=/dev/sda1 ro quiet splash
    initrd /initrd.img
}
```

5. GRUB　2を、デフォルトのメニューエントリを起動する前に10秒間待機するように設定するにはどうしますか？

   `/etc/default/grub` に、`GRUB_TIMEOUT=10` を 追加します。

6. GRUBレガシーシェルから、2番目のディスクの最初のパーティションにGRUBをインストールするにはどうしますか？

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

102.3 共有ライブラリを管理する

LPI目標への参照

LPIC-1 v5, Exam 101, Objective 102.3

総重量

1

主な知識分野

- 共有ライブラリを識別する。
- システムライブラリの一般的な場所を特定する。
- 共有ライブラリをロードする。

用語とユーティリティ

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

Linux Professional Institute

102.3 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	102 Linuxのインストールとパッケージ管理
Objective:	102.3 共有ライブラリの管理
Lesson:	1 of 1

はじめに

このレッスンでは、共有ライブラリ（共有オブジェクト　　と呼ぶこともある）について解説します。これは、さまざまなプログラムから利用される、関数やクラスなどの再利用可能なコンパイル済みコードのことです。

まず、共有ライブラリとは何で、どこにあるのか、どうやって見分けるのかについて説明します。次に、それらの保存場所を設定する方法を説明します。最後に、プログラムが利用する共有ライブラリを調べる方法を示します。

共有ライブラリの概念

ソフトウェア・ライブラリは、物理的な部品と同様に、様々なプログラムで利用されるコードの集合体です。例えれば、様々な人に利用されるように、図書館が本や資料を保管しているのと同じです。

プログラムのソースコードから実行可能ファイルを作成するには、2つのステップが必要です。まず、コンパイラ　　がソースコードをマシンコードに変換して、オブジェクトファイル　に保存します。次に、リンカー　がオブジェクトファイルとライブラリを　リンク（取りまとめ）して、最終的な実行可能ファイルを生成します。このリンクには、静的　　な方法と、動的　　な方法があります。静的リンクの時には静的ライブラリの話になりますし、動的リンクの時には共有ライブラリの話になります。それらの違いを説明しましょう。

静的ライブラリ

静的ライブラリはリンク時にプログラムと結合されます。ライブラリコードのコピーがプログラムに埋め込まれて、プログラムの一部になります。つまり、プログラムとライブラリコードが一体化しているので、プログラムの実行時にはライブラリファイルを必要としません。使用されているライブラリファイルが利用可能であることを確認する必要がなく、プログラム単体で実行できることが利点となります。対して、静的にリンクされたプログラムのファイルサイズが大きくなることが欠点です。

共有（動的）ライブラリ

共有ライブラリの場合、リンカはプログラムがライブラリを正しく参照できるように調整するだけです。リンカはライブラリコードをプログラムファイルに埋め込みません。そのため、プログラムの実行時には、共有ライブラリを参照してプログラムに欠けた部分を補う必要があります。これにより、プログラムファイルのサイズが小さくなりますし、複数のプログラムが同じライブラリを使用している場合でもメモリにロードされるのは1つのライブラリコードだけですから、システムリソースを効率的に利用することができます。

共有オブジェクトファイルの命名規則

共有ライブラリの名前は、次の3つの要素で構成されます。

- ライブラリ名（通常はプレフィックスが `lib`）
- `so`（“shared object” の略）
- ライブラリのバージョン番号

例を示します: `libpthread.so.0`

対する静的ライブラリの名前は `.a` で終わります（例: `libpthread.a`）。

NOTE

プログラムの実行には共有ライブラリのファイルが必要であるため、ほとんどすべてのLinuxシステムには共有ライブラリが含まれています。静的ライブラリは、プログラムをリンクする時にのみ必要なので、エンドユーザー用のシステムに含まれないのが一般的です。

`glibc`（GNU Cライブラリ）は共有ライブラリの良い例です。Debian GNU/Linux 9.9システムでは、そのファイルの名前は `libc.so.6` です。このようなシンプルなファイル名は通常、ライブラリファイルの実体を指すシンボリックリンクです。ライブラリファイルの実体の名前には、正確なバージョン番号が含まれています。`glibc` の例では、次のようなシンボリックリンクです。

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

このように、正確なバージョンを含む名前の共有ライブラリファイルを、より一般的なファイル名で参照するのはよくあることです。

共有ライブラリの他の例には、`libreadline`（入力行の編集機能を提供する。Emacsモードとviモードがある）や、`libcrypt`（暗号、ハッシュ、エンコードなどの関数を含む）、`libcurl`（マルチプロトコルファイル転送ライブラリ）などが挙げられます。

Linuxシステムでは、一般的に次のディレクトリに共有ライブラリを置きます。

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

NOTE 共有ライブラリの概念はLinuxだけのものではありません。たとえば、WindowsではDLL（ダイナミックリンクライブラリ）と呼びます。

共有ライブラリパスの設定

動的にリンクされたプログラムを実行する場合には、動的リンカー（`ld.so` ないし `ld-linux.so`）が、プログラムに埋め込まれたライブラリへの参照を解決します。動的リンカーは、ライブラリパス と呼ばれるいくつかのディレクトリでライブラリを探します。ライブラリパスは、`/etc/ld.so.conf` ファイル、ならびに `/etc/ld.so.conf.d` ディレクトリにあるファイルで定義します。通常、前者には、後者の `*.conf` ファイル群を読み取るための `include` 行が1つだけ含まれています。

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

`/etc/ld.so.conf.d` ディレクトリには、いくつかの `*.conf` ファイルが置かれています。

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

これらの `*.conf` ファイルには、共有ライブラリが置かれているディレクトリへの絶対パスが含まれます。

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

```
/lib/x86_64-linux-gnu
/usr/lib/x86_64-linux-gnu
```

`ldconfig` コマンドは、これらの設定ファイルを読み取って、ライブラリの検索に役立つ一群のシンボリックリンクを作成し、さらにキャッシュファイル `/etc/ld.so.cache` を更新します。つまり、設定ファイルを追加ないし更新した場合には、`ldconfig` を実行する必要があります。

`ldconfig` の主要なオプションを示します:

`-v`, `--verbose`
ライブラリのバージョン番号、ディレクトリ名、作成したリンクを表示する。

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

この例では `libfuse.so.2` が、共有ライブラリファイルの実体 `libfuse.so.2.9.7` を参照していることが分かります。

`-p`, `--print-cache`
キャッシュに保存されているライブラリとディレクトリのリストを表示します。

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

キャッシュがリンクの完全なso名をどのように利用しているかに着目してください。

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

`ls` のロングオプション（-l）で `/lib/x86_64-linux-gnu/libfuse.so.2` を表示すると、同じディレクトリにある共有ライブラリファイル `libfuse.so.2.9.7` へのリンクであることがわかります。

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

NOTE

rootが所有する `/etc/ld.so.cache` への書き込みアクセスが必要なため、`sudo` を使用して rootとして `ldconfig` を実行する必要があります。`ldconfig` のオプションの詳細については、マニュアルを参照してください。

上記の設定ファイルに加えて、`LD_LIBRARY_PATH` 環境変数で、共有ライブラリの検索パスを一時的に追加できます。環境変数には、ライブラリを検索するディレクトリをコロン（`:`）で区切って指定します。たとえば、現在のシェルセッションのライブラリパスに `/usr/local/mylib` を追加するには、次のようにします。

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

次のように値を確認できます。

```
$ echo $LD_LIBRARY_PATH
/usr/local/mylib
```

`LD_LIBRARY_PATH` 環境変数を削除するには、次のようにします。

```
$ unset LD_LIBRARY_PATH
```

変更を永続的にするには、Bashの初期化スクリプトに次の行を追加します。

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

`/etc/bash.bashrc` や `~/.bashrc` など、初期化スクリプトのいずれか1つに追加してください。

NOTE LD_LIBRARY_PATH は共有ライブラリに対する環境変数で、PATH は実行可能ファイルに対する環境変数です。環境変数とシェルの設定については、レッスン103.1を参照してください。

実行可能ファイルの依存関係を調べる

あるプログラムが必要とする共有ライブラリを調べるには、ldd コマンドにプログラムファイルへの絶対パスを与えます。共有ライブラリファイルのパスと、ファイルがロードされるメモリアドレスが16進数で表示されます。

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

同様に、ldd を使用して、共有ライブラリの依存関係を調べることもできます。

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

ldd に -u（--unused）オプションを指定すると、プログラムではコードを使用していないが、依存している共有ライブラリを表示します。（訳注: プログラムをビルドする際の問題であり、一般ユーザーやシステム管理者が気にする必要はほとんどありません。）

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

未使用ライブラリへの不必要な依存関係は、バイナリを構築するときにリンカに指定するオプションが不適切な際に発生します。readelf や objdump などのコマンドを使用して調査できます。これらのコマンドを、発展演習で使用します。

演習

1. 次の共有ライブラリ名をそれぞれのパーツに分けてみましょう。

ファイル名	ライブラリ名	SO サフィックス	バージョン番号
`linux-vdso.so.1`			
`libprocps.so.6`			
`libdl.so.2`			
`libc.so.6`			
`libsystemd.so.0`			
`ld-linux-x86-64.so.2`			

2. ソフトウェアを開発したので、その共有ライブラリディレクトリ（`/opt/lib/mylib`）をシステムに追加したいとします。その絶対パスを `mylib.conf` というファイルに書き込みました。
   - このファイルをどのディレクトリに置きますか？
   - 変更を有効にするには、どのコマンドを実行しますか？

3. `kill` コマンドが必要とする共有ライブラリを表示するコマンドはどうなりますか？

発展演習

1. `objdump` は、オブジェクトファイルの情報を表示するユーティリティです。`which objdump` を使用して、システムにインストールされていることを確認してください。インストールされていない場合は、インストールしてください。
   - `objdump` の `-p`（`--private-headers`）オプションと `grep` を使用して、 `glibc` の依存関係を表示してください:
   - `objdump` の `-p`（`--private-headers`）オプションと `grep` を使用して、 `glibc` のso名を表示してください:
   - `objdump` の `-p`（`--private-headers`）オプションと `grep` を使用して、Bashの依存関係を表示してください:

まとめ

このレッスンでは、次のことを学びました。

- 共有（動的）ライブラリとは何か。
- 共有ライブラリと静的ライブラリの違い。
- 共有ライブラリの名前（so名）。
- `/lib` や `/usr/lib` など、Linuxシステムで共有ライブラリが置かれるディレクトリ。
- ダイナミックリンカー `ld.so`（`ld-linux.so`）の役割。
- `/etc` にある `ld.so.conf` ファイルや、`ld.so.conf.d` ディレクトリで、共有ライブラリパスを設定する方法。
- `LD_LIBRARY_PATH` 環境変数を使用して共有ライブラリパスを設定する方法。
- 実行可能ファイルと共有ライブラリの依存関係を調べる方法。

このレッスンでは以下のコマンドを使用しました:

`ls`
ディレクトリの内容を一覧表示します。

`cat`
ファイルを連結し、標準出力に表示します。

`sudo`
スーパーユーザーの権限でコマンドを実行します。

`ldconfig`
共有ライブラリに必要なリンクを作成します。

`echo`
環境変数の値を表示します。

`export`
子プロセスに引き渡す環境変数を指定します。

`unset`
環境変数を削除します。

`ldd`
プログラムが依存する共有ライブラリを表示します。

`readelf`
ELF（executable and linkable format）ファイルに関する情報を表示します。

`objdump`
オブジェクトファイルに関する情報を表示します。

演習の解答

1. 次の共有ライブラリ名をそれぞれのパーツに分けてみましょう。

ファイル名	ライブラリ名	SO サフィックス	バージョン番号
`linux-vdso.so.1`	`linux-vdso`	`so`	`1`
`libprocps.so.6`	`libprocps`	`so`	`6`
`libdl.so.2`	`libdl`	`so`	`2`
`libc.so.6`	`libc`	`so`	`6`
`libsystemd.so.0`	`libsystemd`	`so`	`0`
`ld-linux-x86-64.so.2`	`ld-linux-x86-64`	`so`	`2`

2. ソソフトウェアを開発したので、その共有ライブラリディレクトリ（`/opt/lib/mylib`）をシステムに追加したいとします。その絶対パスを `mylib.conf` というファイルに書き込みました。
   - このファイルをどのディレクトリに置きますか？

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

   - 変更を有効にするには、どのコマンドを実行しますか？

     `ldconfig`

3. `kill` コマンドが必要とする共有ライブラリを表示するコマンドはどうなりますか？

   `ldd /bin/kill`

発展演習の解答

1. `objdump` は、オブジェクトファイルからの情報を表示するコマンドラインユーティリティです。 `which objdump` を使用してシステムにインストールされているかどうかを確認してください。そうでない場合は、インストールしてください。
   - `objdump` の `-p`（`--private-headers`）オプションと `grep` を使用して、 `glibc` の依存関係を表示してください:

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

   - `objdump` の `-p`（`--private-headers`）オプションと `grep` を使用して、 `glibc` のso名を表示してください:

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

   - `objdump` の `-p`（`--private-headers`）オプションと `grep` を使用して、Bashの依存関係を表示してください:

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

102.4 Debianパッケージ管理を利用する

LPI目標への参照

LPIC-1 v5, Exam 101, Objective 102.4

総重量

3

主な知識分野

- Debianバイナリパッケージのインストール、アップグレード、アンインストール。
- インストールされている場合とインストールされていない場合がある時に、特定のファイルまたはライブラリを含むパッケージを検索する。
- バージョン、コンテンツ、依存性、パッケージの完全性、インストール状態(パッケージのインストールの有無など)などのパッケージ情報を取得する。
- aptの知識。

用語とユーティリティ

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

102.4 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	102 Linuxのインストールとパッケージ管理
Objective:	102.4 Debianパッケージ管理
Lesson:	1 of 1

はじめに

その昔、Linuxがまだ揺籃期にあった頃、ソフトウェアを配布する最も一般的な方法は、ソースコードを含む圧縮ファイル（通常は `.tar.gz` アーカイブ）で、それを展開して自分でコンパイルしていました。

しかし、LinuxカーネルやXサーバーのような大規模なプロジェクトをコンパイルするための時間や能力などのリソースを誰もが持っているわけではありません。ですから、ソフトウェアの量と複雑さが増すにつれて、コンパイル済みのソフトウェアを配布することの必要性が増していきました。

やがて、これらのソフトウェア “パッケージ” の配布方法を標準化する取り組みが進められ、最初のパッケージマネージャーが誕生しました。このツールの誕生により、システムへのソフトウェアのインストールや設定、削除が、より簡単に行えるようになりました。

その中の１つが、Debianパッケージ形式（`.deb`）とそのパッケージツール（`dpkg`）でした。現在では、それらはDebian自体だけでなく、Ubuntuやそこから派生したディストリビューションなどでも広く使用されています。

Debianベースのシステムでポピュラーなもう1つのパッケージ管理ツールは `apt`（Advanced Package Tool）です。これは、パッケージのインストール、保守、削除など多くの側面を効率化し、操作を簡単にします。

このレッスンでは、`dpkg` と `apt` の両方を使用して、DebianベースのLinuxシステムでソフトウェアを取得、インストール、保守、削除する方法を学習します。

Debianパッケージツール（dpkg）

`dpkg`（Debian Package）ツールは、Debianベースのシステムにソフトウェアパッケージをインストール、構成、保守、削除するために不可欠なユーティリティです。最も基本的な操作は、`.deb` パッケージをインストールすることです。次の方法で実行します。

```
# dpkg -i PACKAGENAME
```

ここで、`PACKAGENAME` はインストールする `.deb` ファイルの名前です。

パッケージのアップグレードも同じ方法で処理できます。パッケージをインストールする前に、`dpkg` は以前のバージョンがシステムにすでに存在するかどうかをチェックします。以前のバージョンが存在する場合、パッケージは新しいバージョンにアップグレードされます。そうでない場合は、新規インストールされます。

依存関係への対処

パッケージが正しく動作するために、他のパッケージを必要とする（依存する）ことがよくあります。たとえば、画像エディタがJPEGファイルを開くためのライブラリを必要としたり、GUIのユーティリティがそのユーザーインターフェイスを実現するためにQtやGTKなどのウィジェットツールキットを必要とすることなどがあります。

`dpkg` は必要なパッケージがシステムにインストールされているかどうかをチェックして、インストールされていない場合はパッケージのインストールが失敗します。この時 `dpkg` は、不足しているパッケージをリストします。ただし、`dpkg` が依存関係を解決することはできません。必要な `.deb` パッケージを見つけてインストールするのは、ユーザーの責任です。

以下の例は、OpenShotビデオエディターパッケージのインストールを試みたけれども、依存関係から必要となるいくつかのパッケージが無かったことを示しています。

```
# dpkg -i openshot-qt_2.4.3+dfsg1-1_all.deb
(Reading database ... 269630 files and directories currently installed.)
Preparing to unpack openshot-qt_2.4.3+dfsg1-1_all.deb ...
Unpacking openshot-qt (2.4.3+dfsg1-1) over (2.4.3+dfsg1-1) ...
dpkg: dependency problems prevent configuration of openshot-qt:
 openshot-qt depends on fonts-cantarell; however:
  Package fonts-cantarell is not installed.
 openshot-qt depends on python3-openshot; however:
  Package python3-openshot is not installed.
```

```
 openshot-qt depends on python3-pyqt5; however:
  Package python3-pyqt5 is not installed.
 openshot-qt depends on python3-pyqt5.qtsvg; however:
  Package python3-pyqt5.qtsvg is not installed.
 openshot-qt depends on python3-pyqt5.qtwebkit; however:
  Package python3-pyqt5.qtwebkit is not installed.
 openshot-qt depends on python3-zmq; however:
  Package python3-zmq is not installed.

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

上の例では、OpenShotは `fonts-cantarell`、`python3-openshot`、`python3-pyqt5`、`python3-pyqt5.qtsvg`、`python3-pyqt5.qtwebkit`、`python3-zmq` パッケージに依存しています。OpenShotをインストールするには、これらすべてをインストールする必要があります。

パッケージの削除

パッケージを削除するには、`dpkg` に `-r` オプションを指定し、その後にパッケージ名を渡します。たとえば、次のコマンドはシステムから `unrar` パッケージを削除します。

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

削除操作では依存関係がチェックされて、削除しようとするパッケージに依存する他のすべてのパッケージが削除されない限り、そのパッケージを削除することはできません。行おうとすると、次のようなエラーメッセージが表示されます。

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

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

複数のパッケージ名を `dpkg -r` に渡せば、それらがすべて一度に削除されます。

パッケージが削除されても、対応する設定ファイルがシステムに残ります。パッケージに関連する すべての ファイルを削除する場合は、`-r` ではなく `-P` （purge）オプションを使用します。

NOTE

依存関係が満たされていない場合でも、`dpkg -i --force PACKAGENAME` のように `--force` パラメータを追加すれば、`dpkg` にパッケージのインストールや削除を強制することができます。ただし、そうするとインストールされたパッケージや、さらにはシステムが壊れた状態になることがあります。何が起きるのか確信が持てない限り、`--force` は使用しないほうが良いでしょう。

パッケージ情報を見る

バージョン、アーキテクチャ、メンテナ、依存関係などの `.deb` パッケージに関する情報を取得するには、`dpkg` コマンドに `-I` オプションを指定し、その後に調べるパッケージのファイル名を渡します。

```
# dpkg -I google-chrome-stable_current_amd64.deb
 new Debian package, version 2.0.
 size 59477810 bytes: control archive=10394 bytes.
    1222 bytes,    13 lines      control
   16906 bytes,   457 lines   *  postinst             #!/bin/sh
   12983 bytes,   344 lines   *  postrm               #!/bin/sh
    1385 bytes,    42 lines   *  prerm                #!/bin/sh
 Package: google-chrome-stable
 Version: 76.0.3809.100-1
 Architecture: amd64
 Maintainer: Chrome Linux Team <chromium-dev@chromium.org>
 Installed-Size: 205436
 Pre-Depends: dpkg (>= 1.14.0)
 Depends: ca-certificates, fonts-liberation, libappindicator3-1, libasound2 (>=
1.0.16), libatk-bridge2.0-0 (>= 2.5.3), libatk1.0-0 (>= 2.2.0), libatspi2.0-0 (>=
2.9.90), libc6 (>= 2.16), libcairo2 (>= 1.6.0), libcups2 (>= 1.4.0), libdbus-1-3
(>= 1.5.12), libexpat1 (>= 2.0.1), libgcc1 (>= 1:3.0), libgdk-pixbuf2.0-0 (>=
2.22.0), libglib2.0-0 (>= 2.31.8), libgtk-3-0 (>= 3.9.10), libnspr4 (>= 2:4.9-2~),
libnss3 (>= 2:3.22), libpango-1.0-0 (>= 1.14.0), libpangocairo-1.0-0 (>= 1.14.0),
```

```
libuuid1 (>= 2.16), libx11-6 (>= 2:1.4.99.1), libx11-xcb1, libxcb1 (>= 1.6),
libxcomposite1 (>= 1:0.3-1), libxcursor1 (>> 1.1.2), libxdamage1 (>= 1:1.1),
libxext6, libxfixes3, libxi6 (>= 2:1.2.99.4), libxrandr2 (>= 2:1.2.99.3),
libxrender1, libxss1, libxtst6, lsb-release, wget, xdg-utils (>= 1.0.2)
 Recommends: libu2f-udev
 Provides: www-browser
 Section: web
 Priority: optional
 Description: The web browser from Google
  Google Chrome is a browser that combines a minimal design with sophisticated
technology to make the web faster, safer, and easier.
```

インストールされているパッケージとパッケージの内容の一覧

システムにインストールされているすべてのパッケージを一覧表示するには、`dpkg --get-selections` のように、`--get-selections` オプションを使用します。次の例のように、`dpkg` に `-L PACKAGENAME` オプションを渡せば、そのパッケージによってインストールされたすべてのファイルのリストを取得できます。

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

ファイルが属するパッケージを見つける

システム内のファイルが、どのパッケージに所属しているのかを知りたいことがあります。これには、`dpkg-query` コマンドに `-S` オプションと、調べたいファイルのパスを指定します。

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

インストールされたパッケージの再構成

パッケージをインストールする最後に、post-install と呼ばれるスクリプトが実行されて、アクセス許可や構成ファイルの配置など、ソフトウェアの実行に必要なすべてのものがセットアップされます。この時に、ソフトウェアの実行方法に関する設定を行うために、ユーザーに問いあわせが行われる場合があります。

時には、設定ファイルが破損してしまったり、異常が発生しているので、パッケージの設定を “新鮮な” 状態に戻したいことや、初期設定の質問に対する回答を変更したいことがあります。このような場合は、パッケージ名を指定して `dpkg-reconfigure` コマンドを実行します。

このユーティリティは、古い設定ファイルをバックアップし、新しいファイルを適切なディレクトリに展開してから、インストール時と同様に post-install スクリプトを実行します。次の例は、`tzdata` パッケージの再設定を行います。

```
# dpkg-reconfigure tzdata
```

apt（Advanced Package Tool）

apt（Advanced Package Tool）は、複数のツールから成るパッケージ管理システムであり、パッケージのインストール、アップグレード、削除、および管理を大幅に簡素化します。APTは、高度な検索機能や、依存関係の自動的な解決などの機能も備えています。

APTは `dpkg` の “代用品” ではありません。APTは “フロントエンド” であり、操作を効率化し、依存関係の解決などの `dpkg` には備わっていない機能とのギャップを埋めるものと考えればよいでしょう。

APTは、インストール可能なパッケージを保管する「ソフトウェアリポジトリ」と連携して動作します。リポジトリは、ローカルないしリモートのサーバー、あるいは（あまり一般的ではありませんが）CD-ROMディスクの場合もあります。

DebianやUbuntuなどのLinuxディストリビューションでは、ディストリビュータ自身が（公式）リポジトリを運用しています。公式リポジトリには無いソフトウェアを提供するために、開発者やコミュニティが別のリポジトリを運用していることもあります。

APTには多くのユーティリティがありますが、主なものを以下に示します:

`apt-get`
: パッケージをダウンロード、インストール、アップグレード、削除します。

`apt-cache`
: パッケージのインデックスを用いて、検索などを行います。

`apt-file`
パッケージ内のファイルを検索します。

また、`apt-get` と `apt-cache` の使用頻度が高いオプションを1つのユーティリティにまとめた“とっつきやすい”ユーティリティである `apt` というシンプルな名前のコマンドもあります。`apt` コマンドのオプション（サブコマンド）の多くは、`apt-get` コマンドと同じで、ほとんど同様に使えます。ただし、`apt` コマンドは（比較的新しいため）インストールされていないシステムもありますから、`apt-get` と `apt-cache` を使う方法を学んでおくことが推奨されています。

NOTE `apt` と `apt-get` は、パッケージとパッケージインデックスをリモートサーバーからダウンロードすることがあるので、ネットワーク接続が必要なことがあります。

パッケージインデックスの更新

APTでソフトウェアをインストールないしアップグレードする前に、パッケージ情報を更新するために、まずパッケージインデックスを更新することがお勧めです。これには、`apt-get` コマンドに `update` サブコマンドを指定します。

```
# apt-get update
Ign:1 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:2 https://repo.skype.com/deb stable InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu disco InRelease
Hit:4 http://repository.spotify.com stable InRelease
Hit:5 http://dl.google.com/linux/chrome/deb stable Release
Hit:6 http://apt.pop-os.org/proprietary disco InRelease
Hit:7 http://ppa.launchpad.net/system76/pop/ubuntu disco InRelease
Hit:8 http://us.archive.ubuntu.com/ubuntu disco-security InRelease
Hit:9 http://us.archive.ubuntu.com/ubuntu disco-updates InRelease
Hit:10 http://us.archive.ubuntu.com/ubuntu disco-backports InRelease
Reading package lists... Done
```

TIP `apt-get update` の代わりに、`apt update` も使用できます。

パッケージのインストールと削除

パッケージインデックスを更新したら、パッケージをインストールできます。これには、`apt-get install` の後に、インストールするパッケージの名前を指定します。

```
# apt-get install xournal
Reading package lists... Done
Building dependency tree
```

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

同様に、パッケージを削除するには、`apt-get remove` の後に、パッケージ名を指定します：

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

パッケージをインストールまたは削除するときに、APTは依存関係を自動的に解決します。つまり、インストールするパッケージが必要とするパッケージがすべてインストールされますし、削除するパッケージに依存しているパッケージはすべて削除されます。APTは、処理の続行を尋ねる前に、インストールまたは削除されるパッケージの一覧を表示します。

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

パッケージを削除しても、その設定ファイルは消されずに残ることに注意してください。パッケージと設定ファイルを削除するには、`remove` ではなく `purge` サブコマンドを使用するか、`remove` サブコマンドに `--purge` オプションを追加します。

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

ないし

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

TIP `apt install` と `apt remove` を使用することもできます。

壊れた依存関係の修正

システムを使っていると “壊れた依存関係” が発生することがあります。これは、インストールされている1つ以上のパッケージが、インストールされていないか存在しなくなった別のパッケージに依存していることを意味します。APT実行時のエラーや、手動でインストールしたパッケージなどが原因となって発生することがあります。

これを解決するには、`apt-get install -f` コマンドを使用します。これは、欠けているパッケージをインストールすることで、壊れた依存関係を “修正” して、すべてのパッケージの整合性を保ちます。

TIP `apt install -f` を使用することもできます。

パッケージのアップグレード

APTでは、インストールされているパッケージを、自動的にリポジトリから入手できる最新バージョンにアップグレードできます。これは、`apt-get upgrade` コマンドで行いますが、その前に `apt-get update` でパッケージインデックスをで更新しておきます。

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

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

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

出力の末尾にあるサマリーには、アップグレードされるパッケージの数、インストール、削除、保持されるパッケージの数、合計ダウンロードサイズ、必要なディスク容量が表示されます。アップグレードを完了するには、`Y` と答えて、`apt-get` がタスクを完了するのを待ちます。

1つのパッケージのみをアップグレードするには、`apt-get upgrade` にパッケージ名を指定します。`dpkg` と同様に、`apt-get` はまずパッケージがインストールされているかどうかをチェックし、インストールされていればリポジトリが提供している最新バージョンにアップグレードします。インストールされていなければ、新規にインストールします。

TIP `apt update` と `apt uprade` を使用することもできます。

ローカルキャッシュ

パッケージをインストールまたは更新する際には、まずパッケージに対応する `.deb` ファイルがローカルキャッシュディレクトリにダウンロードされます。デフォルトでは、キャッシュは `/var/cache/apt/archives` ディレクトリです。部分的にダウンロードされたファイルは `/var/cache/apt/archives/partial/` にコピーされます。

パッケージをインストールしたりアップグレードしたりすると、キャッシュディレクトリが非常に大きくなることがあります。容量を再利用するには、`apt-get clean` コマンドを使用してキャッシュを空にします。これで、`/var/cache/apt/archives` と `/var/cache/apt/archives/partial/` ディレクトリの内容が削除されます。

TIP `apt clean` を使用することもできます。

パッケージの検索

`apt-cache` ユーティリティは、パッケージを検索したり、ファイルが含まれているパッケージを探すなど、パッケージインデックスに対する操作を行います。

パッケージを検索するには、`apt-cache search` に続けて検索パターンを指定します。パッケージ名、説明、提供されるファイルのいずれかがパターンに一致するパッケージの一覧が表示されます。

```
# apt-cache search p7zip
liblzma-dev - XZ-format compression library - development files
liblzma5 - XZ-format compression library
forensics-extra - Forensics Environment - extra console components (metapackage)
```

```
p7zip - 7zr file archiver with high compression ratio
p7zip-full - 7z and 7za file archivers with high compression ratio
p7zip-rar - non-free rar module for p7zip
```

この例では、パターンに一致しているように見えない `liblzma5 - XZ-format compression library` が出力されています。このような場合は、`show` サブコマンドで、パッケージの説明を含む詳細な情報を表示すると、そこにパターンが見つかります。

```
# apt-cache show liblzma5
Package: liblzma5
Architecture: amd64
Version: 5.2.4-1
Multi-Arch: same
Priority: required
Section: libs
Source: xz-utils
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Jonathan Nieder <jrnieder@gmail.com>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 259
Depends: libc6 (>= 2.17)
Breaks: liblzma2 (<< 5.1.1alpha+20110809-3~)
Filename: pool/main/x/xz-utils/liblzma5_5.2.4-1_amd64.deb
Size: 92352
MD5sum: 223533a347dc76a8cc9445cfc6146ec3
SHA1: 8ed14092fb1caecfebc556fda0745e1e74ba5a67
SHA256: 01020b5a0515dbc9a7c00b464a65450f788b0258c3fbb733ecad0438f5124800
Homepage: https://tukaani.org/xz/
Description-en: XZ-format compression library
 XZ is the successor to the Lempel-Ziv/Markov-chain Algorithm
 compression format, which provides memory-hungry but powerful
 compression (often better than bzip2) and fast, easy decompression.
 .
 The native format of liblzma is XZ; it also supports raw (headerless)
 streams and the older LZMA format used by lzma. (For 7-Zip's related
 format, use the p7zip package instead.)
```

検索パターンに 正規表現 を使って、高度で複雑な検索が行えますが、その詳細はこのレッスンの範囲外です。（訳注: レッスン103.7を参照してください。）

TIP `apt-cache search` の代わりに `apt search` を、`apt-cache show` の代わりに `apt show` を使用することもできます。

ソースリスト

APTは、パッケージの入手先を把握するために　ソースリスト　を利用します。`/etc/apt` ディレクトリにある　`sources.list`　ファイルが、そのリストです。このファイルは、`vi`、`pico`、`nano`　などのテキストエディタで直接編集したり、`aptitude`　や　`synaptic`　などのグラフィカルツールで編集できます。

`sources.list` には、次のような行を設定します。

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

この行には、以下の構成要素があります:　アーカイブタイプ、URL、ディストリビューション、1つ以上のコンポーネント:

アーカイブタイプ
: 実行可能ソフトウェア（バイナリパッケージ、`deb`）と、ソフトウェアのソースコード（ソースパッケージ、`deb-src`）の2種類があります。例はバイナリパッケージです。

URL
: リポジトリのURL。

ディストリビューション
: パッケージが提供されるディストリビューションの名前（ないしコードネーム）。1つのリポジトリが複数のディストリビューションのパッケージをホストすることがあります。例の `disco` は、Ubuntu 19.04 のコードネーム Disco Dingo です。

コンポーネント
: コンポーネントは、パッケージのセットを意味します。コンポーネントの構成は、ディストリビューションによって異なります。例えば、Ubuntuとその派生版では、以下のようになります。:

  `main`
  : 公式にサポートされているパッケージが含まれます。

  `restricted`
  : 公式にサポートされているクローズドソースのソフトウェアが含まれます。例えば、グラフィックカード用のデバイスドライバなど。

  `universe`
  : コミュニティによってメンテナンスされている、オープンソースソフトウェアが含まれます。

multiverse
サポート外の、クローズドソースや特許に絡んだソフトウェアが含まれます。
別の例として、Debianでは以下のような主要コンポーネントがあります:

main
Debianフリーソフトウェアガイドライン（DFSG: Debian Free Software Guidelines）に準拠していて、DFSGではないソフトウェアに依存すること無く実行できるパッケージで構成されています。ここに含まれるパッケージは、Debianディストリビューションの一部であるとみなされます。

contrib
DFSGに準拠したパッケージであるが、`main` に含まれない（DFSGに準拠していない）パッケージに依存しているものが含まれます。

non-free
DFSGに準拠していないパッケージが含まれます。

security
セキュリティアップデートが含まれます。.

backports
`main` に含まれるパッケージの、より新しいバージョンが含まれます。 Debianの安定版の開発サイクルは非常に長い（約2年）ので、コアリポジトリである `main` を変更しなくても、最新版のパッケージを入手できるようになります。

NOTE
Debianフリーソフトウェアガイドラインの詳細は、https://www.debian.org/social_contract#guidelines をご覧ください。

パッケージを取得するためのリポジトリを追加するには、リポジトリに対応する行（通常はリポジトリメンテナが提供します）を、`sources.list` の末尾に追加してから、`apt-get update` でパッケージインデックスを再読み込みします。すると、`apt-get install` で、新しいリポジトリのパッケージをインストールできるようになります。

文字 `#` で始まる行はコメントと見なされ、無視されます。

`/etc/apt/sources.list.d` ディレクトリ

メインの `/etc/apt/sources.list` ファイルを変更しなくても、`/etc/apt/sources.list.d` ディレクトリにファイルを置くことで、APTが使用するリポジトリを追加できます。それらは、`sources.list` と同じ構文のテキストファイルで、サフィックスは `.list` です。

以下に、`/etc/apt/sources.list.d/buster-backports.list` というファイルの例を示します。

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

パッケージ内容の表示とファイルの検索

`apt-file` ユーティリティでは、パッケージの内容を表示したり、あるファイルを含むパッケージを検索したりするなど、パッケージインデックスを利用する多くの操作を実行できます。このユーティリティは、デフォルトではシステムにインストールされていないことがあります。その場合は、`apt-get` を使用してインストールします。

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

インストール後に、`apt-file` でパッケージキャッシュを更新します。

```
# apt-file update
```

処理は数秒で完了し、`apt-file` が使用できるようになります。

パッケージの内容を表示するには、`list` サブコマンドにパッケージ名を指定します。

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

TIP `apt-file list` の代わりに `apt list` を使用することもできます。

`search` サブコマンドにファイル名を指定することで、すべてのパッケージからファイルを検索できます。たとえば、`libSDL2.so` というファイルが、どのパッケージから提供されているかを知りたい場合は、以下のようにします。

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

パッケージ `libsdl2-dev` が、ファイル `/usr/lib/x86_64-linux-gnu/libSDL2.so` を提供していることが分かります。

`apt-file search` と `dpkg-query` の違いは、`apt-file search` はパッケージインデックス

からパッケージを検索するのに対して、`dpkg-query` はインストールされたパッケージしか検索しないことです。

演習

1. `dpkg` で `package.deb` という名前のパッケージをインストールするにはどうしますか？

2. `dpkg-query` で `7zr.1.gz` という名前のファイルが含まれているパッケージを見つけてください。

3. パッケージ `file-roller` が `unzip` パッケージに依存している場合に、`dpkg -r unzip` コマンドで、システムから `unzip` というパッケージを削除できるでしょうか？ できない場合、どのように削除すればよいでしょう？

4. `apt-file` で、ファイル `/usr/bin/unrar` を含むパッケージを見つけるにはどうしますか？

5. `apt-cache` で、パッケージ `gimp` の情報を表示するにはどうしますか？

発展演習

1. Debian `xenial` ディストリビューション用のソースパッケージが、リポジトリ `http://us.archive.ubuntu.com/ubuntu/` にホストされています。ここには `universe` コンポーネントのパッケージも含まれています。`/etc/apt/sources.list` にどのような行を追加すればよいですか？

2. プログラムをコンパイルしている時に、ヘッダファイル `zzip-io.h` が存在しないというエラーメッセージが表示されましたた。このファイルを提供しているパッケージを調べるにはどうしますか？

3. 依存しているパッケージがある場合に、依存関係の警告を無視して、`dpkg` でパッケージを削除するにはどうしますか？

4. `apt-cache` で `midori` と呼ばれるパッケージの詳細情報を得るにはどうしますか？

5. `apt-get` でパッケージのインストールや更新を行う前に、パッケージインデックスを最新にするにはどうしますか？

まとめ

このレッスンでは、次のことを学びました。

- `dpkg` を使用してパッケージをインストールおよび削除する方法。
- インストール済みのパッケージと、パッケージの内容を表示する方法。
- インストール済みのパッケージを再構成する方法。
- `apt` とは何か、それを使用してパッケージをインストール、アップグレード、削除する方法。
- `apt-cache` を使用してパッケージを検索する方法。
- `/etc/apt/sources.list` ファイルのしくみ。
- `apt-file` を使用してパッケージの内容を表示する方法、あるいはあるファイルを含んでいるパッケージを見つける方法。

次のコマンドについて説明しました。

`dpkg -i`
1つないし複数のパッケージファイルをインストールします。

`dpkg -r`
1つないし複数のインストール済みパッケージを削除します。

`dpkg -I`
パッケージファイルを調べて、インストールするソフトウェアと依存関係の詳細を表示します。

`dpkg --get-selections`
システムにインストール済みのパッケージを一覧表示します。

`dpkg -L`
パッケージがインストールしたすべてのファイルのリストを出力します。

`dpkg-query`
ファイル名を指定し、そのファイルをインストールしたパッケージ名を出力します。

`dpkg-reconfigure`
パッケージの post-install スクリプトを再実行して、インストール時の初期設定を再実行します。

`apt-get update`

`/etc/apt/` ディレクトリに設定されているリポジトリと一致するように、ローカルのパッケージインデックスを更新します。

`apt-get install`

リモートリポジトリからパッケージをダウンロードし、その依存関係とともにインストールします。また、既にダウンロードされているDebianパッケージファイルをインストールすることもできます。

`apt-get remove`

指定されたパッケージをシステムからアンインストールします。

`apt-cache show`

パッケージインデックスに基づき、指定したパッケージの詳細を表示します。

`apt-cache search`

指定したパッケージを、ローカルのパッケージインデックスから検索します。

`apt-file update`

パッケージインデックスを更新し、`apt-file` コマンドがその内容を照会できるようにします。

`apt-file search`

パッケージに含まれているファイルから、パターンに一致するものを検索して表示します。

`apt-file list`

このコマンドは、パッケージの内容を表示します。

演習の解答

1. `dpkg` で `package.deb` という名前のパッケージをインストールするにはどうしますか？

   `dpkg` に `-i` オプションを指定します。

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

2. `dpkg-query` で `7zr.1.gz` という名前のファイルが含まれているパッケージを見つけてください。

   `dpkg-query` に `-S` オプションを指定します。

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

3. パッケージ `file-roller` が `unzip` パッケージに依存している場合に、`dpkg -r unzip` コマンドで、システムから `unzip` というパッケージを削除できるでしょうか？ できない場合、どのように削除すればよいでしょう？

   削除できません。`dpkg` は依存関係を解決しないため、インストールされている別のパッケージが削除しようとしているパッケージに依存している場合は、パッケージを削除することはできません。この例では、まず `file-roller` を削除してから（他には依存しないと仮定して）、次に `unzip` を削除するか、次のように両方を同時に削除します。

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

4. `apt-file` で、ファイル `/usr/bin/unrar` を含むパッケージを見つけるにはどうしますか？

   `search` オプションにパス（ないしファイル名）を指定します。

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

5. `apt-cache` で、パッケージ `gimp` の情報を表示するにはどうしますか？

   `show` オプションにパッケージ名を指定します。

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

発展演習の回答

1. Debian　　xenial　　ディストリビューション用のソースパッケージが、リポジトリ http://us.archive.ubuntu.com/ubuntu/　にホストされています。ここには　universe　コンポーネントのパッケージも含まれています。/etc/apt/sources.list　　にどのような行を追加すればよいですか？

   ソースパッケージは deb-src タイプであるため、次のようになります。

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

   この行を、/etc/apt/sources.list.d/　内の　.list　ファイルに追加することもできます。名前は何でも構いませんが、xenial_sources.list　　のようにわかりやすい名前にしましょう。

2. プログラムをコンパイルしている時に、ヘッダファイル　zzip-io.h　が存在しないというエラーメッセージが表示されましたた。このファイルを提供しているパッケージを調べるにはどうしますか？

   システムに存在しないファイルを含むパッケージを探すには、apt-file　search　を使用します:

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

3. 依存しているパッケージがある場合に、依存関係の警告を無視して、dpkg　　でパッケージを削除するにはどうしますか？

   --force　オプションが使用できますが、システムが一貫性のない状態、あるいは “壊れた”　　状態になるリスクが高いので、何をやろうとしているのかを正確に理解していない限り、実行しない方が良いでしょう。

4. apt-cache で midori と呼ばれるパッケージの詳細情報を得るにはどうしますか？

   apt-cache show の後にパッケージ名を指定します。

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

5. apt-get　でパッケージのインストールや更新を行う前に、パッケージインデックスを最新にするにはどうしますか？

   apt-get　update　を実行します。これにより、/etc/apt/sources.list　ファイルまたは /etc/apt/sources.list.d/　ディレクトリに記述されているリポジトリから、最新のパッ

ケージインデックスがダウンロードされます。

102.5 RPMとYUMパッケージ管理を使用する

LPI目標への参照

LPIC-1 v5, Exam 101, Objective 102.5

総重量

3

主な知識分野

- RPM，YUMとZypperを使用してパッケージをインストール、再インストール、アップグレード、および削除する。
- バージョン、ステータス、依存関係、整合性、署名などのRPMパッケージに関する情報を取得する。
- パッケージが提供するファイルを特定し、特定のファイルがどのパッケージから得られるかを調べる。
- dnfの知識。

用語とユーティリティ

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

102.5 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	102 Linuxのインストールとパッケージ管理
Objective:	102.5 RPMおよびYUMパッケージ管理を使用する
Lesson:	1 of 1

はじめに

その昔、Linuxがまだ揺籃期にあった頃、ソフトウェアを配布する最も一般的な方法は、ソースコードを含む圧縮ファイル（通常は `.tar.gz` アーカイブ）で、それを展開して自分でコンパイルしていました。

しかし、Linuxカーネルや Xサーバーのような大規模なプロジェクトをコンパイルするための時間や能力などのリソースを誰もが持っているわけではありません。ですから、ソフトウェアの量と複雑さが増すにつれて、コンパイル済みのソフトウェアを配布することの必要性が増していきました。

やがて、これらのソフトウェア “パッケージ” の配布方法を標準化する取り組みが進められ、最初のパッケージマネージャーが誕生しました。このツールの誕生により、システムへのソフトウェアのインストールや設定、削除が、より簡単に行えるようになりました。

その中の1つが、Red Hatによって開発された RPMパッケージマネージャ とそのツール（`rpm`）です。現在、それらはRed Hat Enterprise Linux（RHEL）自体だけでなく、Fedora、CentOS、Oracle Linuxなどの子孫や、openSUSEなどのディストリビューション、さらにはIBMのAIXなど他のオペレーティングシステムにも広く使用されています。

Red Hat互換ディストリビューションでポピュラーなパッケージ管理ツールには、`yum`

（YellowDog Updater Modified）、`dnf`（Dandified YUM:しゃれたYUMの意）、`zypper`などがあります。これらは、パッケージのインストール、保守、削除など多くの側面を効率化し、操作を簡単にします。

このレッスンでは、Linuxシステム上でソフトウェアの入手、インストール、管理、削除を行うために、`rpm`、`yum`、`dnf`、`zypper`を使用する方法を学びます。

NOTE

パッケージ形式が同じでも、ディストリビューションごとに内部的な違いがあるため、openSUSE用に作成されたパッケージはRHELシステムでは動作しないことがありますし、逆も同様です。パッケージを検索するときは、互換性を確認し、できるだけお使いのディストリビューション用のパッケージを使用してください。

RPMパッケージマネージャ（rpm）

RPMパッケージマネジャ（`rpm`）は、Red Hatベース（または派生）システムで、ソフトウェアパッケージを管理するために不可欠なツールです。

パッケージのインストール、アップグレード、削除

最も基本的な操作は、パッケージをインストールすることです。次のように行います。

```
# rpm -i PACKAGENAME
```

ここで、`PACKAGENAME`はインストールする`.rpm`ファイルの名前です。

システムに既に（古い）パッケージがインストールされている場合は、`-U`オプションで新しいバージョンにアップグレード（更新）できます。

```
# rpm -U PACKAGENAME
```

`PACKAGENAME`がまだインストールされていない場合には、インストールされます。インストール済みパッケージのみをアップグレードするには、`-F`オプションを使用します。

どちらの操作でも、`-v`オプションを追加すると詳細情報を出力し、`-h`オプションを追加すると進行状況を視覚的に表示するハッシュ記号（#）を出力します。（Linuxの多くのコマンドと同様に）複数のパラメータを1つに組み合わせることができます。つまり`rpm -i -v -h`は`rpm -ivh`と同じです。

インストールされたパッケージを削除するには、`rpm`に`-e`オプション（“erase”の意）と、削除するパッケージ名を指定します。

```
# rpm -e wget
```

インストールされている別のパッケージが、削除しようとしたパッケージに依存している場合は、次のエラーメッセージが表示されます。

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

削除したいパッケージに依存するパッケージ（例では `file-roller`）を先に削除すれば、削除することができます。複数のパッケージ名を `rpm -e` に渡して、一度に複数のパッケージを削除できます。

依存関係への対処

パッケージが正しく動作するために、他のパッケージを必要とする（依存する）ことがよくあります。たとえば、画像エディタがJPEGファイルを開くためのライブラリを必要としたり、GUIのユーティリティがそのユーザーインターフェイスを実現するためにQtやGTKなどのウィジェットツールキットを必要とすることなどがあります。

`rpm` は必要なパッケージがシステムにインストールされているかどうかをチェックして、インストールされていない場合はパッケージのインストールが失敗します。この場合、`rpm` は不足しているパッケージをリストします。ただし、それ自体で依存関係を解決することはできません。

以下の例では、GIMPイメージエディターパッケージのインストールを試みたけれども、依存関係から必要となるいくつかのパッケージが無かったことを示しています。

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

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

依存しているパッケージの `.rpm` パッケージを見つけてインストールすることは、ユーザーの責任です。`yum`、`zypper`、`dnf` などのパッケージマネージャーでは、必要なファイルを提供しているパッケージを調べる機能が備わっています。これらについては、レッスンの後半で説明します。

インストールされているパッケージを調べる

システムにインストールされているすべてのパッケージのリストを取得するには、`rpm -qa` を使います（“query all” と覚えましょう）。

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

パッケージ情報を見る

バージョン番号、アーキテクチャ、インストール日、パッケージャ、概要など、インストールされている パッケージに関する情報を取得するには、`rpm -qi`（“query info” と覚えましょう）にパッケージ名を指定します。例を示します:

```
# rpm -qi unzip
Name        : unzip
Version     : 6.0
Release     : 19.el7
Architecture: x86_64
Install Date: Sun 25 Aug 2019 05:14:39 PM EDT
Group       : Applications/Archiving
Size        : 373986
License     : BSD
Signature   : RSA/SHA256, Wed 25 Apr 2018 07:50:02 AM EDT, Key ID 24c6a8a7f4a80eb5
```

```
Source RPM  : unzip-6.0-19.el7.src.rpm
Build Date  : Wed 11 Apr 2018 01:24:53 AM EDT
Build Host  : x86-01.bsys.centos.org
Relocations : (not relocatable)
Packager    : CentOS BuildSystem <http://bugs.centos.org>
Vendor      : CentOS
URL         : http://www.info-zip.org/UnZip.html
Summary     : A utility for unpacking zip files
Description :
The unzip utility is used to list, test, or extract files from a zip
archive. Zip archives are commonly found on MS-DOS systems. The zip
utility, included in the zip package, creates zip archives. Zip and
unzip are both compatible with archives created by PKWARE(R)'s PKZIP
for MS-DOS, but the programs' options and default behaviors do differ
in some respects.

Install the unzip package if you need to list, test or extract files from
a zip archive.
```

インストールされている　　パッケージ内含まれるファイルのリストを取得するには、`-ql`（“query list”と覚えましょう）とパッケージ名を指定します。

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

まだインストールされていないパッケージの情報やファイルリストを調べたい時は、上記のコマンドに　`-p`　オプションを追加し、その後にRPMファイルの名前（`FILENAME`）を指定します。具体的には、`rpm -qi PACKAGENAME` を `rpm -qip FILENAME` としたり、`rpm -ql PACKAGENAME` を `rpm -qlp FILENAME` とします。

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

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

(□□)
```

ファイルが属するパッケージを見つける

ファイルがどのパッケージに属しているかを調べるには、`-qf`（“query file” と覚えましょう）にファイルのフルパスを指定します。

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

この例では、ファイル `/usr/bin/unzip` は、`unzip-6.0-19.el7.x86_64` パッケージに属していることがわかります。

YellowDog Updater Modified（YUM）

`yum` は元々、Yellow Dog Linuxディストリビューションのパッケージ管理用ツールであるYellow Dog Updater（YUP）として開発されました。その後、Fedora、CentOS、Red Hat Enterprise Linux、Oracle Linuxなど、RPMベースの他のシステムでパッケージを管理できるよう進化してきました。

機能的には、Debianベースのシステムの `apt` ユーティリティに似ており、パッケージを検索、インストール、更新、削除し、依存関係を自動的に処理することができます。`yum` を使って、1つのパッケージをインストールしたり、システム全体を一度にアップグレードしたりできます。

パッケージの検索

パッケージをインストールするにはその名前を調べる必要がありますが、`yum search PATTERN` で探せます。ここで `PATTERN` は検索するパッケージの名前です。パッケージ名ないし説明にPATTERNが含まれているパッケージのリストが表示されます。たとえば、7Zip圧縮ファイル（拡張子が `.7z`）用のユーティリティを探す時には、次のようにします:

```
# yum search 7zip
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.ufscar.br
 * epel: mirror.globo.com
 * extras: mirror.ufscar.br
 * updates: mirror.ufscar.br
============================ N/S matchyutr54ed: 7zip ============================
p7zip-plugins.x86_64 : Additional plugins for p7zip
p7zip.x86_64 : Very high compression ratio file archiver
p7zip-doc.noarch : Manual documentation and contrib directory
p7zip-gui.x86_64 : 7zG - 7-Zip GUI version

  Name and summary matches only, use "search all" for everything.
```

パッケージのインストール、アップグレード、削除

`yum` を使用してパッケージをインストールするには、コマンド `yum install PACKAGENAME` を使用します。ここで指定する `PACKAGENAME` は、パッケージの名前です。`yum` は、インストールしようとしているパッケージが必要とする（依存する）パッケージををオンラインリポジトリから取得し、すべての必要なパッケージをインストールします。

```
# yum install p7zip
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.ufscar.br
 * epel: mirror.globo.com
 * extras: mirror.ufscar.br
 * updates: mirror.ufscar.br
Resolving Dependencies
--> Running transaction check
---> Package p7zip.x86_64 0:16.02-10.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package        Arch            Version               Repository     Size
================================================================================
Installing:
 p7zip          x86_64          16.02-10.el7          epel          604 k

Transaction Summary
================================================================================
Install  1 Package

Total download size: 604 k
Installed size: 1.7 M
Is this ok [y/d/N]:
```

インストール済みのパッケージをアップグレードするには、`yum update PACKAGENAME` を使います。ここで指定する `PACKAGENAME` は、アップグレードするパッケージの名前です。例を示します:

```
# yum update wget
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.ufscar.br
 * epel: mirror.globo.com
 * extras: mirror.ufscar.br
 * updates: mirror.ufscar.br
Resolving Dependencies
--> Running transaction check
---> Package wget.x86_64 0:1.14-18.el7 will be updated
---> Package wget.x86_64 0:1.14-18.el7_6.1 will be an update
```

```
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package     Arch          Version                    Repository      Size
================================================================================
Updating:
 wget        x86_64        1.14-18.el7_6.1            updates        547 k

Transaction Summary
================================================================================
Upgrade  1 Package

Total download size: 547 k
Is this ok [y/d/N]:
```

パッケージ名を省略すると、アップデート可能なすべてのインストール済みパッケージを更新します。

あるパッケージがアップデート可能であるかどうかを確認するには、`yum check-update PACKAGENAME` を使います。`yum update` と同様に、パッケージ名を省略するとインストール済みのすべてのパッケージの更新をチェックします。

インストールされているパッケージを削除するには、`yum remove PACKAGENAME` を使います。ここで指定する `PACKAGENAME` は、削除するパッケージの名前です。

ファイルが属するパッケージを見つける

先の `gimp` 画像エディタを（rpmで）インストールする例では、依存関係が満たされないのでインストールに失敗しました。この時、`rpm` は不足しているファイルを表示しますが、それを提供するパッケージ名は表示しません。

先の例で不足している依存関係の1つに `libgimpui-2.0.so.0` がありました。このような場合に、このファイルが所属するパッケージを調べるには、`yum whatprovides` を使い、検索したいファイルの名前を指定します。

```
# yum whatprovides libgimpui-2.0.so.0
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.ufscar.br
 * epel: mirror.globo.com
 * extras: mirror.ufscar.br
 * updates: mirror.ufscar.br
```

```
2:gimp-libs-2.8.22-1.el7.i686 : GIMP libraries
Repo        : base
Matched from:
Provides    : libgimpui-2.0.so.0
```

この例で、そのファイルを提供するパッケージは `gimp-libs-2.8.22-1.el7.i686` であることがわかりました。`yum install gimp-libs` コマンドで、そのパッケージをインストールできます。

このテクニックは、システムにすでに存在するファイルに対しても有効です。たとえば、ファイル `/etc/hosts` がどこから来たのかを知りたい場合は、次のコマンドを実行します。

```
# yum whatprovides /etc/hosts
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.ufscar.br
 * epel: mirror.globo.com
 * extras: mirror.ufscar.br
 * updates: mirror.ufscar.br
setup-2.8.71-10.el7.noarch : A set of system configuration and setup files
Repo        : base
Matched from:
Filename    : /etc/hosts
```

`/etc/host` ファイルは、`setup-2.8.71-10.el7.noarch` パッケージが提供していることがわかります。

パッケージ情報の取得

バージョン、アーキテクチャ、説明、サイズなど、パッケージに関する情報を取得するには、`yum info PACKAGENAME` を使用します。ここで指定する `PACKAGENAME` は、情報を調べたいパッケージの名前です。

```
# yum info firefox
Last metadata expiration check: 0:24:16 ago on Sat 21 Sep 2019 02:39:43 PM -03.
Installed Packages
Name         : firefox
Version      : 69.0.1
Release      : 3.fc30
Architecture : x86_64
Size         : 268 M
Source       : firefox-69.0.1-3.fc30.src.rpm
Repository   : @System
```

```
From repo    : updates
Summary      : Mozilla Firefox Web browser
URL          : https://www.mozilla.org/firefox/
License      : MPLv1.1 or GPLv2+ or LGPLv2+
Description  : Mozilla Firefox is an open-source web browser, designed
             : for standards compliance, performance and portability.
```

リポジトリの管理

`yum` が使用する “repos”（訳注：リポジトリの意）は、`/etc/yum.repos.d/` ディレクトリに格納されています。`CentOS-Base.repo` などの `.repo` ファイルが、それぞれのリポジトリを定義します。

リポジトリを追加したい時は、上記のディレクトリに `.repo` ファイルを追加するか、または `/etc/yum.conf` の末尾に追加します。ただし、リポジトリの追加や管理には、`yum-config-manager` ツールを使うことが推奨されています。

リポジトリを追加するには、`--add-repo` オプションに `.repo` ファイルへのURLを指定します。

```
# yum-config-manager --add-repo https://rpms.remirepo.net/enterprise/remi.repo
Loaded plugins: fastestmirror, langpacks
adding repo from: https://rpms.remirepo.net/enterprise/remi.repo
grabbing file https://rpms.remirepo.net/enterprise/remi.repo to
/etc/yum.repos.d/remi.repo
repo saved to /etc/yum.repos.d/remi.repo
```

利用可能な（登録されている）リポジトリのリストを取得するには、`yum repolist all` を使います。次のように出力されます。

```
# yum repolist all
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.ufscar.br
 * epel: mirror.globo.com
 * extras: mirror.ufscar.br
 * updates: mirror.ufscar.br
repo id                      repo name                    status
updates/7/x86_64             CentOS-7 - Updates           enabled:  2,500
updates-source/7             CentOS-7 - Updates Sources   disabled
```

▯▯ になっているリポジトリは、ソフトウェアのインストールやアップグレード時に無視さ

れます。リポジトリの有効/無効を切り替えるには、`yum-config-manager` ユーティリティで、リポジトリIDを指定します。

上の出力例では、リポジトリIDが各行の先頭列（`repo id`）に表示されています。`/` よりも前の部分がIDですから、`CentOS-7 - Updates` リポジトリのIDは `updates` で、`updates/7/x86_64` ではありません。

```
# yum-config-manager --disable updates
```

このコマンドは、`updates` リポジトリを無効にします。再度、有効にするには、次のコマンドを使います。

```
# yum-config-manager --enable updates
```

NOTE

Yumは、ダウンロードしたパッケージとそのメタデータを、キャッシュディレクトリ（通常は `/var/cache/yum`）に保存します。システムがアップグレードされ、新しいパッケージがインストールされると、このキャッシュのファイルサイズが非常に大きくなることがあります。キャッシュをクリーンアップしてディスク領域を再利用するには、`yum clean` コマンドで何をクリーンアップするかを指定します。よく使用するのは、ダウンロードしたパッケージを削除する `packages`（`yum clean packages`）と、関連するメタデータを削除する `metadata`（`yum clean metadata`）です。詳細は、`yum` のマニュアルページを参照（`man yum`）してください。

DNF

`dnf`（Dandified YUM:「しゃれた」YUMの意）は、`yum` から分岐した改良版であり、事実上の後継です。コマンドやオプションの多くは共通です。この節では、`dnf` の概要を簡単に説明します。

パッケージの検索

`dnf search PATTERN` で、`PATTERN` を検索します。たとえば、`dnf search unzip` は、名前または説明に `unzip` という単語を含むすべてのパッケージを表示します。

パッケージに関する情報の取得

`dnf info PACKAGENAME`

パッケージのインストール

`dnf install PACKAGENAME` で、`PACKAGENAME` はインストールしたいパッケージの名前です。この名前は検索で見つけられます。

パッケージの削除

`dnf remove PACKAGENAME`

パッケージのアップグレード

`dnf upgrade PACKAGENAME` は、1つのパッケージのみを更新します。パッケージ名を省略すると、インストール済みのすべてのパッケージを更新します。

指定のファイルを提供するパッケージを見つける

`dnf provides FILENAME`

システムにインストールされているすべてのパッケージのリストを取得する

`dnf list --installed`

パッケージの内容を一覧表示する

`dnf repoquery -l PACKAGENAME`

NOTE

`dnf` にはヘルプシステムが組み込まれていて、各サブコマンドの詳細情報（追加のパラメータなど）を表示します。`dnf help` に続けて、サブコマンドを `dnf help install` のように指定します。

リポジトリの管理

`yum` や `dnf`（後述の `zypper` 含む）は、リポジトリと連携しています。それぞれのディストリビューションにはデフォルトのリポジトリのリストがあり、管理者は必要に応じてリポジトリを追加ないし削除できます。

使用可能なすべてのリポジトリのリストを取得するには、`dnf repolist` を使います。有効なリポジトリのみを一覧表示するには `--enabled` オプションを追加し、無効なリポジトリのみを一覧表示するには `--disabled` オプションを追加します。

```
# dnf repolist
Last metadata expiration check: 0:20:09 ago on Sat 21 Sep 2019 02:39:43 PM -03.
repo id                    repo name                                       status
*fedora                    Fedora 30 - x86_64                              56,582
*fedora-modular            Fedora Modular 30 - x86_64                         135
*updates                   Fedora 30 - x86_64 - Updates                    12,774
*updates-modular           Fedora Modular 30 - x86_64 - Updates                145
```

リポジトリを追加するには、`dnf config-manager --add_repo URL` を使います。ここで指定する `URL` は、リポジトリへの完全なURLです。リポジトリを有効にするには、`dnf config-manager --set-enabled REPO_ID` を使います。

同様に、リポジトリを無効にするには `dnf config-manager --set-disabled REPO_ID` を使い

ます。いずれの場合も `REPO_ID` は、リポジトリの一意のIDであり、それを見つけるには `dnf repolist` を使います。追加したリポジトリは、デフォルトで有効になります。

リポジトリは、ディレクトリ `/etc/yum.repos.d/` の `.repo` ファイルに保存されます。 `yum` が使用する構文とまったく同じです。

Zypper

`zypper` は、SUSE Enteprise LinuxやopenSUSEで使用されるパッケージ管理ツールです。機能的には `apt` や `yum` と同様であり、依存関係を自動的に解決して、パッケージをインストール、更新、削除できます。

パッケージインデックスの更新

他のパッケージ管理ツールと同様に、`zypper` はパッケージとメタデータを含むリポジトリと連携します。`zypper` がパッケージの更新を認識できるように、メタデータを随時更新する必要があります。次のコマンドで行います

```
# zypper refresh
Repository 'Non-OSS Repository' is up to date.
Repository 'Main Repository' is up to date.
Repository 'Main Update Repository' is up to date.
Repository 'Update Repository (Non-Oss)' is up to date.
All repositories have been refreshed.
```

`zypper` には、リポジトリごとに有効化できる自動更新機能があります。つまり、（クエリやインストールする際に）自動的に更新されるリポジトリもあれば、手動で更新する必要があるリポジトリもあります。この機能を制御する方法は、後で述べます。

パッケージの検索

パッケージを検索するには、`search`（`se`）サブコマンドを使い、パッケージ名を続けます。

```
# zypper se gnumeric
Loading repository data...
Reading installed packages...

S | Name           | Summary                              | Type
--+----------------+--------------------------------------+--------
  | gnumeric       | Spreadsheet Application              | package
  | gnumeric-devel | Spreadsheet Application              | package
  | gnumeric-doc   | Documentation files for Gnumeric     | package
  | gnumeric-lang  | Translations for package gnumeric    | package
```

searchサブコマンドで、インストール済みのパッケージのリストを取得することもできます。この場合、`zypper se -i` のように、パッケージ名なしで `-i` オプションを使用します。

あるパッケージがインストールされているかどうかを確認するには、上記のコマンドにパッケージ名を追加します。たとえば、次のコマンドは、インストール済みパッケージから、名前に “firefox” を含むパッケージを検索します。

```
# zypper se -i firefox
Loading repository data...
Reading installed packages...

S | Name                                | Summary                  | Type
--+-------------------------------------+--------------------------+--------
i | MozillaFirefox                      | Mozilla Firefox Web B-> | package
i | MozillaFirefox-branding-openSUSE    | openSUSE branding of -> | package
i | MozillaFirefox-translations-common | Common translations f-> | package
```

インストールされていない パッケージのみを検索するには、`se` サブコマンドに `-u` オプションを追加します。

パッケージのインストール、アップグレード、削除

ソフトウェアパッケージをインストールするには、`install`（`in`）サブコマンドに続けてパッケージ名を指定し、次のように実行します:

```
# zypper in unrar
zypper in unrar
Loading repository data...
Reading installed packages...
Resolving package dependencies...

The following NEW package is going to be installed:
  unrar

1 new package to install.
Overall download size: 141.2 KiB. Already cached: 0 B. After the operation,
additional 301.6 KiB will be used.
Continue? [y/n/v/...? shows all options] (y): y
Retrieving package unrar-5.7.5-lp151.1.1.x86_64
                                    (1/1), 141.2 KiB (301.6 KiB unpacked)
Retrieving: unrar-5.7.5-lp151.1.1.x86_64.rpm .....................[done]
Checking for file conflicts: .....................................[done]
```

```
(1/1) Installing: unrar-5.7.5-lp151.1.1.x86_64 .....................[done]
```

`zypper` では、ローカルのRPMパッケージファイルをインストールすることができ、その際にはリポジトリのパッケージを利用して依存関係が自動的に満たされます。例えば、`zypper in /home/john/newpackage.rpm` のように、パッケージ名ではなくパッケージファイルのフルパスを指定します。

システムにインストールされているパッケージを更新するには、`zypper update` を使います。インストールの場合と同様に、インストール/アップグレードするパッケージのリストが表示されて、続行するかどうかを尋ねられます。

何もインストールせずに、利用できるアップデートを一覧表示したい時は、`zypper list-updates` を使います。

パッケージを削除するには、`remove`（`rm`）サブコマンドを使用し、その後にパッケージ名を続けます。

```
# zypper rm unrar
Loading repository data...
Reading installed packages...
Resolving package dependencies...

The following package is going to be REMOVED:
  unrar

1 package to remove.
After the operation, 301.6 KiB will be freed.
Continue? [y/n/v/...? shows all options] (y): y
(1/1) Removing unrar-5.7.5-lp151.1.1.x86_64 .....................[done]
```

パッケージを削除すると、それに依存しているすべてのパッケージが削除されることに注意してください:

```
# zypper rm libgimp-2_0-0
Loading repository data...
Warning: No repositories defined. Operating only with the installed resolvables.
Nothing can be installed.
Reading installed packages...
Resolving package dependencies...

The following 6 packages are going to be REMOVED:
  gimp gimp-help gimp-lang gimp-plugins-python libgimp-2_0-0
  libgimpui-2_0-0
```

```
6 packages to remove.
After the operation, 98.0 MiB will be freed.
Continue? [y/n/v/...? shows all options] (y):
```

ファイルが所属するパッケージを見つける

あるファイルを含んでいるパッケージを調べるには、searchサブオプションに続けて `--provides` オプションと、調べたいファイルの名前（ないしフルパス）を指定します。たとえば、`/usr/lib64/` にあるファイル `libgimpmodule-2.0.so.0` が所属しているパッケージを調べたい場合は、次のようになります。

```
# zypper se --provides /usr/lib64/libgimpmodule-2.0.so.0
Loading repository data...
Reading installed packages...

S | Name          | Summary                                          | Type
--+---------------+--------------------------------------------------+--------
i | libgimp-2_0-0 | The GNU Image Manipulation Program - Libra-> | package
```

パッケージ情報を見る

パッケージのメタデータを表示するには、`info` サブコマンドに続けてパッケージ名を指定します。提供しているリポジトリ、パッケージ名、バージョン、アーキテクチャ、ベンダー、インストールサイズ、インストールされているかどうか、ステータス（最新かどうか）、ソースパッケージ、簡単な説明が表示されます。

```
# zypper info gimp
Loading repository data...
Reading installed packages...

Information for package gimp:
 ------------------------------
Repository     : Main Repository
Name           : gimp
Version        : 2.8.22-lp151.4.6
Arch           : x86_64
Vendor         : openSUSE
Installed Size : 29.1 MiB
Installed      : Yes (automatically)
Status         : up-to-date
Source package : gimp-2.8.22-lp151.4.6.src
Summary        : The GNU Image Manipulation Program
```

```
Description    :
    The GIMP is an image composition and editing program, which can be
    used for creating logos and other graphics for Web pages. The GIMP
    offers many tools and filters, and provides a large image
    manipulation toolbox, including channel operations and layers,
    effects, subpixel imaging and antialiasing, and conversions, together
    with multilevel undo. The GIMP offers a scripting facility, but many
    of the included scripts rely on fonts that we cannot distribute.
```

リポジトリの管理

ソフトウェアリポジトリの管理にも、`zypper` を使用します。システムに登録されているリポジトリの一覧を表示するには、`zypper repos` を使います。

```
# zypper repos
Repository priorities are without effect. All enabled repositories share the same
priority.

#  | Alias                     | Name                                | Enabled | GPG
Check | Refresh
---+---------------------------+-------------------------------------+---------+
----------+--------
 1 | openSUSE-Leap-15.1-1      | openSUSE-Leap-15.1-1                | No      |
----      | ----
 2 | repo-debug                | Debug Repository                    | No      |
----      | ----
 3 | repo-debug-non-oss        | Debug Repository (Non-OSS)          | No      |
----      | ----
 4 | repo-debug-update         | Update Repository (Debug)           | No      |
----      | ----
 5 | repo-debug-update-non-oss | Update Repository (Debug, Non-OSS) | No      |
----      | ----
 6 | repo-non-oss              | Non-OSS Repository                  | Yes     | (r
) Yes  | Yes
 7 | repo-oss                  | Main Repository                     | Yes     | (r
) Yes  | Yes
 8 | repo-source               | Source Repository                   | No      |
----      | ----
 9 | repo-source-non-oss       | Source Repository (Non-OSS)         | No      |
----      | ----
10 | repo-update               | Main Update Repository              | Yes     | (r
) Yes  | Yes
11 | repo-update-non-oss       | Update Repository (Non-Oss)         | Yes     | (r
```

```
) Yes   | Yes
```

`Enabled` 列で、リポジトリの有効/無効を確認できます。有効/無効を切り替えるには、`modifyrepo` サブコマンドの後に、`-e`（有効）ないし `-d`（無効）オプションと、リポジトリのエイリアス名（上記出力の2列目）を指定します。

```
# zypper modifyrepo -d repo-non-oss
Repository 'repo-non-oss' has been successfully disabled.

# zypper modifyrepo -e repo-non-oss
Repository 'repo-non-oss' has been successfully enabled.
```

前述したように、`zypper` では、リポジトリごとに auto refresh（自動更新）を有効化することができます。この機能を有効にしたリポジトリでは、指定した処理を行う前に `zypper` が自動的にメタデータを更新します（`zypper refresh` と同じ）。この機能の有効/無効を切り替えるには、`modifyrepo` サブコマンドの `-f`（有効）ないし `-F`（無効）オプションで指定します。

```
# zypper modifyrepo -F repo-non-oss
Autorefresh has been disabled for repository 'repo-non-oss'.

# zypper modifyrepo -f repo-non-oss
Autorefresh has been enabled for repository 'repo-non-oss'.
```

リポジトリの追加と削除

`zypper` が使用するソフトウェアリポジトリを追加するには、次のように、`addrepo` サブコマンドに続けて、リポジトリのURLとリポジトリ名を指定します。

```
# zypper addrepo http://packman.inode.at/suse/openSUSE_Leap_15.1/ packman
Adding repository 'packman' .....................................[done]
Repository 'packman' successfully added

URI         : http://packman.inode.at/suse/openSUSE_Leap_15.1/
Enabled     : Yes
GPG Check   : Yes
Autorefresh : No
Priority    : 99 (default priority)

Repository priorities are without effect. All enabled repositories share the same
priority.
```

リポジトリを追加するときに `-f` オプションを指定すると、自動更新が有効になります。追加したリポジトリはデフォルトで有効になりますが、`-d` オプションを指定するとリポジトリは追加されますが同時に無効化されます。

リポジトリを削除するには、`removerepo` サブコマンドに続けてリポジトリ名（エイリアス）を指定します。上の例で追加したリポジトリを削除するコマンドは、次のようになります。

```
# zypper removerepo packman
Removing repository 'packman' .....................................[done]
Repository 'packman' has been removed.
```

演習

1. Red Hat Enterprise Linuxシステムで `rpm` を使って、インストール中に進行状況バーを表示しながら、パッケージ `file-roller-3.28.1-2.el7.x86_64.rpm` をインストールするにはどうしますか？

2. `rpm` を使って、ファイル `/etc/redhat-release` が含まれているパッケージを見つけてください。

3. `yum` を使って、システム内のすべてのパッケージの更新を確認するにはどうしますか？

4. `zypper` を使って、repo-extrasというリポジトリを無効にするにはどうしますか？

5. DNFに新しいリポジトリを記述した `.repo` ファイルを認識させるには、その `repo` ファイルをどこに置きますか？

発展演習

1. `zypper` を使って、ファイル `/usr/sbin/swapon` が所属するパッケージを見つけるにはどうしますか？

2. `dnf` を使って、システムにインストールされているすべてのパッケージの一覧を表示するにはどうしますか？

3. `dnf` を使って、リポジトリ `https://www.example.url/home:reponame.repo` をシステムに追加するコマンドは何ですか？

4. `zypper` を使って、パッケージ `unzip` がインストールされているかどうかを調べるにはどうしますか？

5. `yum` を使って、ファイル `/bin/wget` が所属するパッケージを見つけてください。

まとめ

このレッスンでは、次のことを学びました。

- `rpm` を使用してパッケージをインストール、アップグレード、削除する方法。
- `yum`、`zypper`、`dnf` の使い方。
- パッケージの情報を表示する方法。
- パッケージ内容のリストを表示する方法。
- ファイルがどのパッケージに属しているかを調べる方法。
- ソフトウェアリポジトリを一覧表示、追加、削除、有効化、無効化する方法。

次のコマンドについて説明しました。

- `rpm`
- `yum`
- `dnf`
- `zypper`

演習の解答

1. Red Hat Enterprise Linuxシステムで `rpm` を使って、インストール中に進行状況バーを表示しながら、パッケージ `file-roller-3.28.1-2.el7.x86_64.rpm` をインストールするにはどうしますか？

   パッケージをインストールするために `-i` オプションを、進行状況を示す "hash marks" を表示するために `-h` オプションを指定します。つまり `rpm -ih file-roller-3.28.1-2.el7.x86_64.rpm` になります。

2. `rpm` を使用して、ファイル `/etc/redhat-release` が含まれているパッケージを見つけてください。

   ファイルに関する情報を調べるために `-qf` オプションを使用します。つまり `rpm -qf /etc/redhat-release` です。

3. `yum` を使って、システム内のすべてのパッケージの更新を確認するにはどうしますか？

   パッケージ名を指定しない `check-update` サブコマンドを使います。`yum check-update`

4. `zypper` を使って、repo-extrasというリポジトリを無効にするにはどうしますか？

   リポジトリのパラメータを変更するために `modifyrepo` サブコマンドを使い、`-d` オプションで無効化します。`zypper modifyrepo -d repo-extras`

5. DNFに新しいリポジトリを記述した `.repo` ファイルを認識させるには、その `repo` ファイルをどこに置きますか？

   DNFは、YUMと同じ場所 `/etc/yum.repos.d/` にある `repo` ファイルを参照します。

発展演習の解答

1. `zypper` を使って、ファイル `/usr/sbin/swapon` が所属するパッケージを見つけるにはどうしますか？

   `se`（検索）サブコマンドと `--provides` オプションを指定します。`zypper se --provides /usr/sbin/swapon`

2. `dnf` を使って、システムにインストールされているすべてのパッケージの一覧を表示するにはどうしますか？

   `list` サブコマンドを使用し、`--installed` オプションを指定します。`dnf list --installed`

3. `dnf` を使って、リポジトリ `https://www.example.url/home:reponame.repo` をシステムに追加するコマンドは何ですか？

   リポジトリの操作は “設定の変更” であるため、`config-manager` の `--add_repo` オプションを指定します。`dnf config-manager --add_repo https://www.example.url/home:reponame.repo`

4. `zypper` を使って、パッケージ `unzip` がインストールされているかどうかを調べるにはどうしますか？

   インストールされている（`-i`）パッケージを検索（`se`）します。`zypper se -i unzip`

5. `yum` を使用して、ファイル `/bin/wget` を提供するパッケージを見つけてください。

   ファイルが所属するパッケージを見つけるには、`whatprovides` とファイル名を指定します。 `yum whatprovides /bin/wget`

102.6 仮想化のゲストOSとしてのLinux

LPI目標への参照

LPIC-1, Exam 101, Objective 102.6

総重量

1

主な知識分野

- 仮想マシンとコンテナの基本的な概念の理解
- コンピュータのインスタンス、ブロックストレージ、ネットワークなどのIaaSクラウドでの、仮想マシンの共通の要素についての理解
- システムを複製したりテンプレートとして利用する際に変更しなくてはならない、Linuxシステム特有の設定項目についての理解
- システムのイメージが、どのようにして仮想マシン・クラウド・コンテナにデプロイされるかの理解
- 仮想化製品により統合されるLinuxにおいての、Linuxの拡張機能ついての理解
- cloud-initの知識

用語とユーティリティ

- 仮想マシン
- Linuxコンテナ
- アプリケーションコンテナ
- ゲストドライバー
- SSHホストキー
- D-Busマシンid

102.6 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	102 Linuxのインストールとパッケージ管理
Objective:	102.6 仮想化ゲストとしてのLinux
Lesson:	1 of 1

はじめに

Linuxの大きな強みの1つは、汎用性にあります。この汎用性の中には、完全に分離された安全な環境として、他のオペレーティングシステムないし個々のアプリケーションをホストする手段として、Linuxを使用できることです。このレッスンでは、仮想化とコンテナ技術の概念に焦点を当て、クラウドプラットフォームに仮想マシンを配備する際に考慮すべきいくつかの技術的な詳細を示します。

仮想化の概要

仮想化は、ハイパーバイザー　　と呼ばれるソフトウェアプラットフォームが、完全にエミュレートされたコンピューターシステムを、プロセスとして実行できるようにする技術です。ハイパーバイザーは、個々の仮想マシンが使用できるハードウェアリソースを管理します。これらの仮想マシンは、ハイパーバイザーの　ゲスト　と呼ばれます（ハイパーバイザーを「ホスト」と呼ぶこともあります）。仮想マシンでは、システムBIOSやハードドライブのディスクコントローラーなど、物理コンピューターの多くのパーツをソフトウェアでエミュレートしています。仮想マシンは、多くの場合は単なるファイルとして保存されているハードディスクイメージを使用し、ハイパーバイザーを介してホストマシンのRAMとCPUにアクセスします。ハイパーバイザーが、ゲストのホストマシンのハードウェアリソースへのアクセスを分離するので、1台のホストマシンで、複数のオペレーティングシステムを実行できるのです。

Linuxで一般的に使用されるハイパーバイザーには、次のようなものがあります:

Xen
: Xenはオープンソースの、Type-1ハイパーバイザーです。Type-1のハイパーバイザーとは、基盤となるオペレーティングシステムに依存せずに、ハードウェア上で直接実行されるものです。この種のハイパーバイザーは、コンピューターから直接起動されるので、ベアメタルハイパーバイザー とも呼ばれます。

KVM
: KVM（Kernel Virtual Machine）は、仮想化用のLinuxカーネルモジュールです。KVMは、汎用のLinuxオペレーティングシステム上で動作しますが、ホストLinuxと統合された完全なハイパーバイザーとして動作します。そのため、Type-1ハイパーバイザーと、Type-2（ホスト型）ハイパーバイザーとのハイブリッド型とも言えます。KVMにデプロイする仮想マシンは、`libvirt` デーモンと、そのユーティリティを使用して作成および管理されます。

VirtualBox
: 簡単に仮想マシンの作成や管理ができる人気のデスクトップアプリケーションです。Oracle VM VirtualBoxはクロスプラットフォームであり、Linux、macOS、およびMicrosoft Windowsで動作します。VirtualBoxの実行には、基盤となるオペレーティングシステムが必要なため、Type-2のハイパーバイザーです。

一部のハイパーバイザーでは、仮想マシンの動的な再配置が可能です。仮想マシンをあるハイパーバイザーから別のハイパーバイザーに移動するプロセスを マイグレーション（移行） と呼びますが、その手法はハイパーバイザーによって異なります。マイグレーションには、仮想マシンがシャットダウンされているときにのみ実行できるものと、稼働中でも実行できるもの（ライブマイグレーション）があります。マイグレーションの技術は、ハイパーバイザーのメンテナンスや、ハイパーバイザーが機能しなくなった時に仮想マシンを別のハイパーバイザーに移動して実行を継続する場合などに役立ちます。

仮想マシンの種類

仮想マシンには、完全仮想化 ゲスト、準仮想化 ゲスト、ハイブリッド ゲストの、主に3種類があります。

完全仮想化
: 完全仮想化ゲストとは、ゲストOS（またはHardwareVM）が、仮想マシンインスタンスとして実行されていることを意識する必要がないゲストです。ゲストOSが実行するすべての命令は、仮想マシン内部で実行されます。つまり、完全仮想化では、命令をシミュレートしたり、実際のハードウェアを操作するための追加のドライバをゲスト内にインストールする必要はありません。このタイプの仮想化をx86ベースのハードウェアで実行するには、ハイパーバイザーをインストールするシステムでIntel VT-xまたはAMD-V CPU拡張機能を有効にする必要があります。これは、BIOSまたはUEFIファームウェア構成メニューから実行できます。

準仮想化

準仮想化ゲスト（PVM）は、ゲストOSが仮想マシンのインスタンスとして実行されていることを認識しているゲストです。このタイプのゲストでは、変更されたカーネルと、ゲストOSがハイパーバイザーの機能やハードウェアリソースを利用するための特別なドライバー（ゲストドライバー）が必要です。これらの仕組みによって、準仮想化ゲストのパフォーマンスが、完全仮想化ゲストのパフォーマンスよりも優れているケースがよくあります。

ハイブリッド

準仮想化と完全仮想化の組み合わせで、完全仮想化ゲストにおいても準仮想化と同様のドライバーを使用することで、オペレーティングシステム自体に変更を加えること無く、ネイティブに近いI/Oパフォーマンスを実現するものです。ディスクとネットワークのI/Oパフォーマンスを強化する、ストレージドライバとネットワークドライバが提供されるのが一般的です。

仮想化プラットフォームでは、パッケージされた仮想化OS用のゲストドライバーが提供されているのが普通です。KVMでは Virtio プロジェクトのドライバーを利用し、Oracle VM VirtualBoxではISO CD-ROMイメージファイルで提供される Guest Additions を利用します。

`libvirt` 仮想マシンの例

`libvirt` によって管理され、KVMハイパーバイザーを使用する仮想マシンの例を見ていきます。多くの場合、仮想マシンはファイルのグループで構成されます。特に重要なのは、仮想マシンを 定義 するXMLファイル（ハードウェア構成、ネットワーク接続、表示機能など）と、オペレーティングシステムやソフトウェア含む、ハードディスクイメージファイルです。

まず、仮想マシンとそのネットワーク環境を定義するXMLファイルの例を見ていきましょう。

```
$ ls /etc/libvirt/qemu
total 24
drwxr-xr-x 3 root root 4096 Oct 29 17:48 networks
-rw------- 1 root root 5667 Jun 29 17:17 rhel8.0.xml
```

NOTE

ディレクトリパスにある `qemu` とは、KVMベースの仮想マシンを実現する基盤となるソフトウェアを指します。QEMUプロジェクトは、仮想マシンが使用するハードウェアデバイス（ディスクコントローラー、ホストCPU、ネットワークカードなど）を、ハイパーバイザーがエミュレートするためのソフトウェアを提供します。

`networks` という名前のディレクトリがあることに着目しましょう。このディレクトリには、仮想マシンが使用するネットワークの定義ファイル（XML形式）が含まれています。

この例では、ハイパーバイザーは1つのネットワークだけを使用しているので、仮想ネットワークセグメントの構成を定義するファイルは1つだけです。

```
$ ls -l /etc/libvirt/qemu/networks/
total 8
drwxr-xr-x 2 root root 4096 Jun 29 17:15 autostart
-rw------- 1 root root  576 Jun 28 16:39 default.xml
$ sudo cat /etc/libvirt/qemu/networks/default.xml
<!--
WARNING: THIS IS AN AUTO-GENERATED FILE. CHANGES TO IT ARE LIKELY TO BE
OVERWRITTEN AND LOST. Changes to this xml configuration should be made using:
  virsh net-edit default
or other application using the libvirt API.
-->

<network>
  <name>default</name>
  <uuid>55ab064f-62f8-49d3-8d25-8ef36a524344</uuid>
  <forward mode='nat'/>
  <bridge name='virbr0' stp='on' delay='0'/>
  <mac address='52:54:00:b8:e0:15'/>
  <ip address='192.168.122.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.122.2' end='192.168.122.254'/>
    </dhcp>
  </ip>
</network>
```

この定義には、クラスCプライベートネットワークと、そのネットワークの（デフォルト）ルーターとして機能するエミュレートされたハードウェアデバイスが含まれています。このネットワークを使用する仮想マシンに割り当てられるDHCPサーバーが使用する、IPアドレスの範囲も定義されています。このネットワークでは、NAT（network address translation）を利用して、ホスト（ハイパーバイザーが）が属するLANなどのネットワークにパケットを転送できます。

次に、Red Hat Enterprise Linux 8の仮想マシン定義ファイルを見てみましょう。（着目すべき箇所を太字で示しています。）

```
$ sudo cat /etc/libvirt/qemu/rhel8.0.xml
<!--
WARNING: THIS IS AN AUTO-GENERATED FILE. CHANGES TO IT ARE LIKELY TO BE
OVERWRITTEN AND LOST. Changes to this xml configuration should be made using:
  virsh edit rhel8.0
```

```
or other application using the libvirt API.
-->

<domain type='kvm'>
  <name>rhel8.0</name>
  <uuid>fadd8c5d-c5e1-410e-b425-30da7598d0f6</uuid>
  <metadata>
    <libosinfo:libosinfo
xmlns:libosinfo="http://libosinfo.org/xmlns/libvirt/domain/1.0">
      <libosinfo:os id="http://redhat.com/rhel/8.0"/>
    </libosinfo:libosinfo>
  </metadata>
  <memory unit='KiB'>4194304</memory>
  <currentMemory unit='KiB'>4194304</currentMemory>
  <vcpu placement='static'>2</vcpu>
  <os>
    <type arch='x86_64' machine='pc-q35-3.1'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <vmport state='off'/>
  </features>
  <cpu mode='host-model' check='partial'>
    <model fallback='allow'/>
  </cpu>
  <clock offset='utc'>
    <timer name='rtc' tickpolicy='catchup'/>
    <timer name='pit' tickpolicy='delay'/>
    <timer name='hpet' present='no'/>
  </clock>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <pm>
    <suspend-to-mem enabled='no'/>
    <suspend-to-disk enabled='no'/>
  </pm>
  <devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='qcow2'/>
      <source file='/var/lib/libvirt/images/rhel8'/>
      <target dev='vda' bus='virtio'/>
```

```
      <address type='pci' domain='0x0000' bus='0x04' slot='0x00' function='0x0'/>
    </disk>
    <controller type='usb' index='0' model='qemu-xhci' ports='15'>
      <address type='pci' domain='0x0000' bus='0x02' slot='0x00' function='0x0'/>
    </controller>
    <controller type='sata' index='0'>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1f' function='0x2'/>
    </controller>
    <controller type='pci' index='0' model='pcie-root'/>
    <controller type='pci' index='1' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='1' port='0x10'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'
multifunction='on'/>
    </controller>
    <controller type='pci' index='2' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='2' port='0x11'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x1'/>
    </controller>
    <controller type='pci' index='3' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='3' port='0x12'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x2'/>
    </controller>
    <controller type='pci' index='4' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='4' port='0x13'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x3'/>
    </controller>
    <controller type='pci' index='5' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='5' port='0x14'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x4'/>
    </controller>
    <controller type='pci' index='6' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='6' port='0x15'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x5'/>
    </controller>
    <controller type='pci' index='7' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='7' port='0x16'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x6'/>
    </controller>
```

```
    <controller type='virtio-serial' index='0'>
      <address type='pci' domain='0x0000' bus='0x03' slot='0x00' function='0x0'/>
    </controller>
    <interface type='network'>
      <mac address='52:54:00:50:a7:18'/>
      <source network='default'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000' bus='0x01' slot='0x00' function='0x0'/>
    </interface>
    <serial type='pty'>
      <target type='isa-serial' port='0'>
        <model name='isa-serial'/>
      </target>
    </serial>
    <console type='pty'>
      <target type='serial' port='0'/>
    </console>
    <channel type='unix'>
      <target type='virtio' name='org.qemu.guest_agent.0'/>
      <address type='virtio-serial' controller='0' bus='0' port='1'/>
    </channel>
    <channel type='spicevmc'>
      <target type='virtio' name='com.redhat.spice.0'/>
      <address type='virtio-serial' controller='0' bus='0' port='2'/>
    </channel>
    <input type='tablet' bus='usb'>
      <address type='usb' bus='0' port='1'/>
    </input>
    <input type='mouse' bus='ps2'/>
    <input type='keyboard' bus='ps2'/>
    <graphics type='spice' autoport='yes'>
      <listen type='address'/>
      <image compression='off'/>
    </graphics>
    <sound model='ich9'>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1b' function='0x0'/>
    </sound>
    <video>
      <model type='virtio' heads='1' primary='yes'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x0'/>
    </video>
    <redirdev bus='usb' type='spicevmc'>
      <address type='usb' bus='0' port='2'/>
    </redirdev>
    <redirdev bus='usb' type='spicevmc'>
```

```
      <address type='usb' bus='0' port='3'/>
    </redirdev>
    <memballoon model='virtio'>
      <address type='pci' domain='0x0000' bus='0x05' slot='0x00' function='0x0'/>
    </memballoon>
    <rng model='virtio'>
      <backend model='random'>/dev/urandom</backend>
      <address type='pci' domain='0x0000' bus='0x06' slot='0x00' function='0x0'/>
    </rng>
  </devices>
</domain>
```

このファイルには、ゲストに割り当てられるRAM容量、CPUコア数、ハードディスクイメージファイル（`disk`　節の中）、表示機能（SPICEプロトコル経由）、USBデバイス、エミュレートされたキーボードとマウスからの入力などのハードウェア設定が定義されています。

仮想マシンのストレージ例

この仮想マシンのハードディスクイメージは、`/var/lib/libvirt/images/rhel8`　です。ホストマシンでのファイルの実体は次のとおりです。

```
$ sudo ls -lh /var/lib/libvirt/images/rhel8
-rw------- 1 root root 5.5G Oct 25 15:57 /var/lib/libvirt/images/rhel8
```

このディスクイメージは、ホストマシン上でわずか5.5GBしか消費していません。しかし、次のコマンド出力からわかるように、実行中の仮想マシンのオペレーティングシステムでは23.3GBと表示されます。

```
$ lsblk
NAME          MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda           252:0    0 23.3G  0 disk
▯▯vda1        252:1    0    1G  0 part /boot
▯▯vda2        252:2    0 22.3G  0 part
  ▯▯rhel-root 253:0    0   20G  0 lvm  /
  ▯▯rhel-swap 253:1    0  2.3G  0 lvm  [SWAP]
```

この違いは、このゲストに使用されているディスクの割り当て方法によるものです。仮想マシンが使用できるディスクイメージには何種類かありますが、次の2つが主要なものです。

COW

Copy-On-Write　や　thin-provisioning、sparse　images　と呼ばれるディスク種別

は、上限サイズのみを定義してディスクファイルを作成する方法です。ディスクイメージファイルのサイズは、新しいデータがディスクに書き込まれた時にのみ増加します。例に示したように、ゲストオペレーティングシステムでは定義したとおりの23.3GBのディスクが見えていますが、ディスクイメージには5.5GBのデータしか書き込まれていません。この仮想マシンに使用されているディスクイメージの形式は、QEMU　COWファイル形式である `qcow2` です。

RAW

raw　または　full　というディスク種別では、指定した容量すべてを事前に割り当てたファイルが作成されます。たとえば、10GBのrawディスクイメージファイルは、ホスト上で10GBのディスク容量を消費します。この種類のディスクでは、必要なディスクスペースがあらかじめ割り当てられているので、ハイパーバイザーが容量を確認して割り当てるという処理を行わなくても書き込みを行えるという、パフォーマンス上のメリットがあります。

Red Hat Enterprise Virtualization や oVirt などの仮想環境管理プラットフォームでは、ゲスト用のバッキングストレージ（仮想ディスク）として、物理ディスクを使用することができます。これらのシステムでは、（仮想マシンは）SAN（ストレージエリアネットワーク）やNAS（ネットワーク接続ストレージ）のデータを読み書きし、ハイパーバイザーはどの仮想マシンがどのストレージを利用しているかを管理します。これらのストレージシステムでは、LVM（論理ボリューム管理）などのテクノロジーを用いて必要に応じて仮想マシンのストレージサイズを拡大/縮小したり、ストレージスナップショットの作成や管理を実現しています。

仮想マシンのテンプレート

ハイパーバイザーから見れば、仮想マシンは単なる実行可能ファイルですから、シナリオに合わせてカスタマイズした　テンプレート　を簡単に作成できます。オペレーティングシステムと基本的なパッケージをインストールして、ロケールや起動のための認証設定を行った仮想マシンを用意しておけば、繰り返し行う作業を減らして新しいシステムの構築にかかる時間を削減できます。

このような仮想マシンのテンプレートをコピーして、新しいゲストシステムを作成することができます。この場合、新しいゲストのホスト名を変更し、ネットワークインターフェイスに新しいMACアドレスを生成し、さらに使用目的に応じた変更を行います。

D-BusマシンID

多くのLinuxディストリビューションでは、インストール時に生成した D-Bus machine ID をマシンの識別IDとして使用します。仮想マシンを　クローン　して別の仮想マシンのテンプレートとする場合には、ハイパーバイザーがそれぞれのゲストシステムに適切なシステムリソースを割り当てられるように、新しいD-BusマシンIDを生成する必要があります。

次のコマンドを使用して、実行中のシステムがD-BusマシンIDを持っていることを確認できます。

```
$ dbus-uuidgen --ensure
```

エラーメッセージが表示されない場合は、マシンIDが既に存在します。現在のD-Busマシン IDを表示するには、次のコマンドを実行します。

```
$ dbus-uuidgen --get
17f2e0698e844e31b12ccd3f9aa4d94a
```

表示された文字列が、現在のマシンIDです。ハイパーバイザーで実行される複数のLinuxシステムが、同じD-BusマシンIDを持つことはできません。

D-BusマシンIDは `/var/lib/dbus/machine-id` にあり、`/etc/machine-id` にシンボリックリンクされています。システムが不安定になりクラッシュする可能性があるため、稼働中のシステムでこのマシンIDを変更することはお勧めしません。複数の仮想マシンが同じD-BusマシンIDを持っている場合は、以下の手順で新しいD-BusマシンIDを生成します。

```
$ sudo rm -f /etc/machine-id
$ sudo dbus-uuidgen --ensure=/etc/machine-id
```

`/var/lib/dbus/machine-id` が `/etc/machine-id` へのシンボリックリンクでない場合は、`/var/lib/dbus/machine-id` を削除する必要があります。

仮想マシンをクラウドにデプロイする

ハイパーバイザーを提供し、組織の仮想ゲストイメージを展開できるIaaS（infrastructure as a service）プロバイダーが多数あります。ほとんどのプロバイダーは、さまざまなLinuxディストリビューションをベースに、カスタム仮想マシンを構築、配備、構成するツールを備えています。これらのプロバイダーの多くは、顧客が組織内で構築した仮想マシンを、展開・移行するためのシステムも導入しています。

IaaS環境におけるLinuxシステムの導入を評価する場合にポイントとなる重要な要素がいくつかあります。

コンピューティングインスタンス

多くのクラウドプロバイダーは、“コンピューティングインスタンス”、つまりクラウドベースのシステム基盤が使用するCPU時間に基づいて利用料金を請求します。アプリケーションが必要とする処理時間を慎重に計画することは、クラウドソリューションのコスト管理に役立ちます。

コンピューティングインスタンスは、クラウド環境に配備した仮想マシンの数を指すこともあります。ここでも、同時に実行するシステムのインスタンスが増えると、全体的

なCPU時間に影響して請求額が増大します。

ブロックストレージ

クラウドプロバイダーは、さまざまなレベルのブロックストレージも用意しています。ウェブベースのネットワークストレージとして利用できるものもあれば、クラウドに配備された仮想マシンがファイル保存に使用するストレージとして利用できるものもあります。

これらサービスのコストは、使用するストレージ容量や、データセンター内のストレージ速度によって異なります。ストレージへのアクセスが高速になるとコストが高くなり、逆に“保存中”のデータ（アーカイブストレージなど）は非常に安くなります。

ネットワーキング

クラウドソリューションプロバイダーが提供する重要なコンポーネントの1つに、仮想ネットワークの構成方法があります。多くのIaaSプロバイダーでは、Webベースのユーティリティを使って、さまざまなネットワーク経路、サブネット、ファイアウォール構成などの設計と実装を行います。DNSソリューションを提供して、インターネットに接続されたシステムにアクセス可能なFQDN（完全修飾ドメイン名）を割り当てることができるものもあります。また、既存のオンプレミス型ネットワークインフラとクラウドベースのインフラをVPN（仮想プライベートネットワーク）で接続して、両方のインフラを一体化できる「ハイブリッド型」ソリューションが用意されていることもあります。

クラウドのゲストに安全にアクセスする

クラウドプラットフォーム上のリモート仮想ゲストにアクセスするには、OpenSSHソフトウェアを使うのが一般的です。クラウド上のLinuxシステムでは、OpenSSHサーバーを実行します。管理者は、公開鍵とOpenSSHクライアントを使ってリモートアクセスします。

管理者は、まず次のコマンドを（ローカルマシンで）実行します。

```
$ ssh-keygen
```

プロンプトに従って、SSHの公開鍵と秘密鍵のペアを作成します。秘密鍵をローカルシステム（`~/.ssh/`）に保存し、公開鍵をリモートのクラウドシステムにコピーします。企業LAN上のネットワークマシンで作業する場合とまったく同じです。

次に、管理者は次のコマンドを（ローカルマシンで）実行します。

```
$ ssh-copy-id -i <public_key> user@cloud_server
```

これで、生成した公開鍵が、リモートのクラウドサーバーにコピーされます。公開鍵はクラウドサーバーの `~/.ssh/authorized_keys` ファイルに保存されて、適切なパーミッションが設定されます。

NOTE

`~/.ssh/` ディレクトリに公開鍵ファイルが1つしかない場合は、`-i` オプションを省略することができます。`ssh-copy-id` コマンドは、ディレクトリ内の公開鍵ファイル（通常はサフィックスが `.pub`）をデフォルトで使用します。

クラウドプロバイダーによっては、新しいLinuxシステムを配備するときに、鍵ペアを自動的に生成します。管理者は、新しいシステムの秘密鍵をクラウドプロバイダーからダウンロードして、ローカルシステムに保存します。SSH鍵のパーミッションは、秘密鍵は `0600`、公開鍵は `0644` とすることに注意してください。

クラウドシステムの事前構成

クラウドへの仮想マシンの展開を簡素化する便利なツールに、`cloud-init` ユーティリティがあります。このコマンドは、あらかじめ構成した仮想マシンイメージと構成ファイルに基づいて、多数のIaaSプロバイダーにLinuxゲストを配置するための、ベンダーニュートラルな方法です。プレーンテキストのYAML（YAML Ain't Markup Language）ファイルに、ネットワーク設定、ソフトウェアパッケージの選択、SSHキーの設定、ユーザーアカウントの作成、ロケールの設定、など多数のオプションを事前に定義しておくことで、新しいシステムをすばやく構築できます。

システムを初めて起動する時に、`cloud-init` がYAMLファイルから設定を読み込んで、システムに適用します。この処理は、システムの初期設定時のみに実行され、クラウドプロバイダーのプラットフォーム上に新しいシステムを簡単に配備できます。

`cloud-init` で使用するYAMLファイルは、cloud-config 形式で定義します。`cloud-config` ファイルの例を示します。

```
#cloud-config
timezone: Africa/Dar_es_Salaam
hostname: test-system

# Update the system when it first boots up
apt_update: true
apt_upgrade: true

# Install the Nginx web server
packages:
 - nginx
```

最初の行のハッシュ記号（#）と `cloud-config` の間に空白がないことに注意してください。

NOTE

`cloud-init` は仮想マシンだけではなく、コンテナ（LXD Linuxコンテナなど）の事前構成にも利用できます。

コンテナ

コンテナテクノロジーは仮想マシンと似ている部分があり、分離された環境にアプリケーションを簡単に配備することができます。仮想マシンではコンピューター全体がエミュレートされますが、コンテナはアプリケーションを実行するのに必要なソフトウェアだけを使用します。これにより、オーバーヘッドを大幅に削減することができます。

コンテナは仮想マシンよりも柔軟です。仮想マシンをあるハイパーバイザーから別のハイパーバイザーに移行できるのと同じように、アプリケーションコンテナをあるホストから別のホストに移行することができます。仮想マシンを移行するにはあらかじめ電源をオフにしておくことが必要な場合がありますが、コンテナでは実行中のアプリケーションを移行することができます。また、コンテナでは、既存のバージョンと並行して、新しいバージョンのアプリケーションを簡単に配備できます。ユーザーが実行中のコンテナとのセッションを閉じると、コンテナ　オーケストレーション　ソフトウェアがコンテナを自動的にシステムから削除して新しいバージョンに置き換えるので、ダウンタイムが短縮されます。

NOTE

Linuxでは、Docker　、Kubernetes、LXD/LXC、systemd-nspawn、OpenShift　など、多数のコンテナ技術が利用できます。コンテナソフトウェアの具体的な実装は、LPIC-1試験の範囲外です。

コンテナは、Linuxカーネルの control groups（cgroups）機構を利用します。cgroupとは、メモリ、CPU時間、ディスク、ネットワーク帯域幅などのシステムリソースを、個々のアプリケーションに分割する機能です。管理者はcgroupを使用して、1つのアプリケーション、ないし1つのcgroupに含まれるアプリケーショングループに、システムリソース制限をセットできます。つまり、コンテナソフトウェアは、管理者のためにcgroupsの管理と展開を容易にするツールを提供するものです。

NOTE

ここでcgroupの概念を説明したのは、アプリケーションをグループに分割することによって、それぞれが互いに影響せずにリソースを使えるようになる概念を紹介するためです。現在、LPIC-1試験に合格するためにcgroupの知識は必要ありません。

演習

1. 完全仮想化ゲストを実行するx86ベースのハードウェアプラットフォームに必要な、CPU拡張は何ですか？

2. 最大のパフォーマンスを必要とするミッションクリティカルなサーバーのインストールでは、どのタイプの仮想化を使用することが望ましいですか？

3. 同じテンプレートから複製され、D-Busを利用する2つの仮想マシンが不安定です。ホスト名とネットワーク設定は異なっています。それぞれの仮想マシンが異なるD-BusマシンIDを持っているかどうかを判断するにはどうしますか？

発展演習

1. 次のコマンドを実行して、仮想マシンを実行するためのCPU拡張機能がシステムで有効になっているかどうかを確認してください（結果はCPUによって異なります）。

   ```
   grep --color -E "vmx|svm" /proc/cpuinfo
   ```

   出力では、`vmx`（Intel VT-xが有効なCPUの場合）か、`svm`（AMD SVMが有効なCPUの場合）が強調表示されます。結果が得られない場合は、BIOSまたはUEFIファームウェアのマニュアルを見て、プロセッサの仮想化を有効にしてください。

2. プロセッサが仮想化をサポートしていたら、KVMハイパーバイザーを実行する方法を、ディストリビューションのドキュメントから探します。
   - KVMハイパーバイザーを実行するために必要なパッケージをインストールしてください。
   - グラフィカルデスクトップ環境を使用している場合は、KVMを利用するためのグラフィカルフロントエンドである `virt-manager` アプリケーションもインストールしてください。仮想マシンのインストールと管理に役立ちます。
   - 好みのLinuxディストリビューションのISOイメージをダウンロードして、そのドキュメントに従ってそのISOから新しい仮想マシンを作成してください。

まとめ

このレッスンでは、仮想マシンとコンテナの基本概念と、これらのテクノロジーをLinuxで使用する方法について説明しました。

次のコマンドについて簡単に説明しました。

`dbus-uuidgen`
DBusマシンIDを確認する。

`ssh-keygen`
リモートシステムにアクセスするときに使う、sshの公開鍵と秘密鍵のペアを生成する。

`ssh-copy-id`
リモート認証のために、SSHの公開鍵をリモートシステムにコピーする。

`cloud-init`
仮想マシンとコンテナを構成し、クラウド環境への配備を支援する

演習の解答

1. 完全に仮想化されたゲストを実行するx86ベースのハードウェアプラットフォームに必要なCPU拡張は何ですか？

   Intel CPUではVT-x、AMD CPUではAMD-V。

2. 最大のパフォーマンスを必要とするミッションクリティカルなサーバーのインストールでは、どのタイプの仮想化を使用することが望ましいですか？

   Xenなどで準仮想化ゲストを利用する場合、ハイパーバイザーと連携するソフトウェアドライバーを使用することで、ハードウェア資源をより有効に活用できます。

3. 同じテンプレートから複製され、D-Busを利用する2つの仮想マシンが不安定です。ホスト名とネットワーク設定は異なっています。それぞれの仮想マシンが異なるD-BusマシンIDを持っているかどうかを判断するにはどうしますか？

   それぞれのマシンで `dbus-uuidgen --get` を実行します。

発展演習の解答

1. 次のコマンドを実行して、仮想マシンを実行するためのCPU拡張機能がシステムで有効になっているかどうかを確認してください（結果はCPUによって異なります）。

   `grep --color -E "vmx|svm" /proc/cpuinfo`。

   出力では、`vmx`（Intel VT-xが有効なCPUの場合）か、`svm`（AMD SVMが有効なCPUの場合）が強調表示されます。結果が得られない場合は、BIOSまたはUEFIファームウェアのマニュアルを見て、プロセッサの仮想化を有効にしてください。

   結果は、使用しているCPUによって異なります。UEFIファームウェアで仮想化拡張機能が有効になっているIntel CPUを搭載したコンピューターからの出力例を次に示します。

   ```
   $ grep --color -E "vmx|svm" /proc/cpuinfo
   flags       : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
   pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp
   lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc
   cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg
   fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer
   aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb
   invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vpid
   ept_ad fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm mpx rdseed
   adx smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln
   pts hwp hwp_notify hwp_act_window hwp_epp md_clear flush_l1d
   ```

2. プロセッサが仮想化をサポートしていたら、KVMハイパーバイザーを実行する方法を、ディストリビューションのドキュメントから探します。
   - KVMハイパーバイザーを実行するために必要なパッケージをインストールしてください。

     ディストリビューションによって異なるので、参考URLを示します:

     Ubuntu — https://help.ubuntu.com/lts/serverguide/libvirt.html

     Fedora — https://docs.fedoraproject.org/en-US/quick-docs/getting-started-with-virtualization/

     Arch Linux — https://wiki.archlinux.org/index.php/KVM

   - グラフィカルデスクトップ環境を使用している場合は、KVMを利用するためのグラフィカルフロントエンドである `virt-manager` アプリケーションもインストールしてください。仮想マシンのインストールと管理に役立ちます。

繰り返しますが、ディストリビューションによって異なります。

```
$ sudo apt install virt-manager
```

- 好みのLinuxディストリビューションのISOイメージをダウンロードして、そのドキュメントに従ってそのISOから新しい仮想マシンを作成してください。

  このタスクは `virt-manager` パッケージで簡単に処理できます（GUI環境の場合）。`virt-install` コマンドを使用してコマンドラインから仮想マシンを作成することもできます。両方を試して、仮想マシンがどのように配備されるかを理解してください。

Linux
Professional
Institute

課題 103: GNUとUnixコマンド

103.1 コマンドラインでの作業

LPI目標への参照

LPIC-1 version 5.0, Exam 101, Objective 103.1

総重量

4

主な知識分野

- 単一のシェルコマンドと1行の連続したコマンドを使用して、コマンドラインで基本的な作業を実行する。
- 環境変数の定義、参照、およびエクスポートを含むシェル環境の使用と変更。
- コマンド履歴の使用と編集。
- 定義されたパスの内側と外側のコマンドを呼び出す。

用語とユーティリティ

- `bash`
- `echo`
- `env`
- `export`
- `pwd`
- `set`
- `unset`
- `type`
- `which`
- `man`

- `uname`
- `history`
- `.bash_history`
- Quoting

103.1 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	103 GNUおよびUnixコマンド
Objective:	103.1 コマンドラインでの作業
Lesson:	1 of 2

はじめに

Linux管理やBashシェルを触り始めたばかりの人は、GUIインターフェースの快適さがないと不安を感じてしまうことがよくあります。彼らは、グラフィカルなファイルマネージャで利用できる視覚的な手がかりと、コンテキスト情報に右クリックでアクセスすることに慣れています。そのため、まずは比較的少数のコマンドラインツールをマスターし、これまでにGUIで操作していたのと同じように、すべてのデータを操作できるようになることが重要です。

システム情報の取得

コマンドラインプロンプト上に点滅するカーソルを見つめながら、最初の質問はおそらく“私はどこ？” でしょう。より正確には、“Linuxファイルシステムのどこにいて、新しいファイルを作った場合、それはどこに置かれるのか？” ということでしょう。ここで求めているのは、自分の 現在の作業ディレクトリ であり、`pwd` コマンドがそれを教えてくれます。

```
$ pwd
/home/frank
```

Frankがシステムにログインしていて、ホームディレクトリ `/home/frank/` にいるとしまし

ょう。彼がファイルシステム内の他の場所を指定せずに `touch` コマンドを使用して空のファイルを作成した場合、ファイルは `/home/frank/` ディレクトリに作成されます。`ls` を使用してディレクトリの内容を一覧表示すると、作成した新しいファイルが表示されます。

```
$ touch newfile
$ ls
newfile
```

ファイルシステム内の場所だけではなく、実行しているLinuxシステムに関する情報が必要になることがよくあります。例えば、ディストリビューションの正確なリリース番号や、ロードされているLinuxカーネルのバージョンなどです。`uname` ツールを使用してそれら確認することができますが、`-a`（“all”）オプションを指定することが重要です。

```
$ uname -a
Linux base 4.18.0-18-generic #19~18.04.1-Ubuntu SMP Fri Apr 5 10:22:13 UTC 2019
x86_64 x86_64 x86_64 GNU/Linux
```

この例で `uname` は、FrankのマシンにLinuxカーネルバージョン4.18.0がインストールされており、64ビット（`x86_64`）CPUでUbuntu18.04を実行していることを示しています。

コマンドについて調べる

未知のLinuxのコマンドについて書かれた文書に出くわすことがあります。コマンドが何をするもので、どうすれば効果的に使用できるのかといった、あらゆる種類の役立つ情報を、コマンドライン自体が教えてくれます。おそらく有用な情報のほとんどは、`man`（訳注: manはMANualの短縮形です）システム内のファイルで見つかるでしょう。

Linux開発者は原則として、作成したユーティリティと一緒に `man` ファイルを配布します。`man` ファイルはきちんと構造化された文書で、内容を直感的に理解できるように、標準化されたセクション見出しで内容が掴みやすくなっています。`man` に続けてコマンド名を入力すると、コマンド名、使い方の概要、詳細な説明、歴史やライセンスなどの重要な情報が表示されます。以下に例を示します。（訳注: 日本語環境では、日本語に翻訳されたmanpページがインストールされていれば、日本語版が表示されます。ただし、翻訳版はバージョンが古いことが少なくありません。）

```
$ man uname
UNAME(1)            User Commands           UNAME(1)
NAME
   uname - print system information
SYNOPSIS
   uname [OPTION]...
DESCRIPTION
```

```
   Print certain system information.  With no OPTION, same as -s.
   -a, --all
      print  all information, in the following order, except omit -p
      and -i if unknown:
   -s, --kernel-name
      print the kernel name
   -n, --nodename
      print the network node hostname
   -r, --kernel-release
      print the kernel release
   -v, --kernel-version
      print the kernel version
   -m, --machine
      print the machine hardware name
   -p, --processor
      print the processor type (non-portable)
   -i, --hardware-platform
      print the hardware platform (non-portable)
   -o, --operating-system
      print the operating system
   --help display this help and exit
   --version
      output version information and exit
AUTHOR
   Written by David MacKenzie.
REPORTING BUGS
   GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
   Report uname translation bugs to
   <http://translationproject.org/team/>
COPYRIGHT
   Copyright©2017 Free Software Foundation, Inc. License  GPLv3+: GNU
   GPL version 3 or later <http://gnu.org/licenses/gpl.html>.
   This is free software: you are free to change and redistribute it.
   There is NO WARRANTY, to the extent permitted by law.
SEE ALSO
   arch(1), uname(2)
   Full documentation at: <http://www.gnu.org/software/coreutils/uname>
   or available locally via: info '(coreutils) uname invocation'
GNU coreutils 8.28       January 2018             UNAME(1)
```

`man` には、正確なコマンド名を指定します。探したいコマンドの正確な名前がわからない場合は、`apropos` コマンドを使用して、`man` ページ内の「名前」と「概要」を検索できます。たとえば、`uname` がLinuxカーネルのバージョンを表示することを思い出せない時には、`kernel` という単語を `apropros` に渡して検索します。おそらく多くの行が表示されますが、目的のものも含まれているはずです。

```
$ apropos kernel
systemd-udevd-kernel.socket (8) - Device event managing daemon
uname (2)            - get name and information about current kernel
urandom (4)          - kernel random number source devices
```

コマンドの完全なドキュメントが必要ない場合は、`type` を使用してコマンドの種類をすばやく調べられます。以下の例では `type` を使用して、一度に4つのコマンドを調べています。結果は、`cp`（“copy”）が `/bin/cp` というプログラムであり、`kill`（実行中のプロセスの状態を変更する）が組み込みコマンドであること（Bashの一部であること）を示しています。

```
$ type uname cp kill which
uname is hashed (/bin/uname)
cp is /bin/cp
kill is a shell builtin
which is /usr/bin/which
```

`uname` が “hashed” と表示されていますが、これはコマンド種別ではなく、シェルが次の実行に備えてハッシュテーブルと呼ばれる一種の辞書にコマンドを記録したことを示しています。ハッシュされたコマンドは、呼び出されたときに素早く実行されます。Frankがログインした直後に `type uname` を実行すれば、`cp` や `which` と同様に、`uname` がファイルに収められたプログラムであることが表示されることでしょう。

NOTE ログインし直さなくても、コマンド `hash -d` を実行すれば、ハッシュテーブルがクリーンアップされます。

スクリプトを作成している場合などには、コマンドのパス名のみを知りたいことがよくあります。`type` の実行例で示した `which` コマンドは、コマンドの絶対パスのみを返します。次の例では、`uname` コマンドと `which` コマンドの両方を検索しています。

```
$ which uname which
/bin/uname
/usr/bin/which
```

NOTE “組み込み” コマンドに関する情報を表示したい場合は、（`man` コマンドではなく）`help` コマンドを使用します。

コマンド履歴の利用

あるコマンドの正しい使い方を慎重に調べ、複雑なオプションや引数をうまく組み合わせて実行することは、よくあることです。しかし、数週間後に同じオプションと引数を使用して

同じコマンドを実行したい時に、詳細を思い出せない場合はどうしたらよいでしょうか？ 最初から調べ直すのではなく、`history` を使用して、以前に使用したコマンドを再利用ほうが簡単です。

`history` と入力すると、過去に実行したコマンドのリストが返され、末尾には直前に実行したコマンドが表示されます。`grep` コマンドにパイプして、探したい文字列を指定することで、以前実行したコマンドを簡単に検索できます。この例では、テキスト `bash_history` を含むコマンドを検索します。

```
$ history | grep bash_history
1605 sudo find /home -name ".bash_history" | xargs grep sudo
```

この実行例では、シーケンス番号1605と、1つのコマンドが返されています。

なお `.bash_history` は、ユーザーのホームディレクトリ内に存在する隠しファイルの名前です。これは隠しファイル（ファイル名の前のドットが付いている）であるため、 `ls` に `-a` オプションを指定してディレクトリの内容を一覧表示した場合にのみ表示されます。

```
$ ls /home/frank
newfile
$ ls -a /home/frank
.  ..  .bash_history  .bash_logout  .bashrc  .profile  .ssh  newfile
```

`.bash_history` ファイルには何が含まれているでしょうか？ 自分の目で確かめれば、最近のコマンドが何百も表示されるでしょう。ただし、直近に実行した ほとんどの コマンドが欠落していることに気付くかもしれません。コマンドを実行するとすぐに、動的な `history` データベースに追加されますが、セッションを終了するまで `.bash_history` ファイルに書き込まれません。

キーボードの上下の矢印キー（ないしは `Ctrl`+`P` と `Ctrl`+`N` ）を使用すると、`history` の内容を活用して、コマンドラインの実行履歴を高速かつ効率的に確認できます。上矢印キー（`Ctrl`+`P`）を何度か押すと、コマンドラインに最近のコマンドが現われます。もう一度実行したいものに到達したら、Enterキーを押して実行できます。これにより、シェルセッション中にコマンドを何度も呼び出したり、必要に応じてコマンドラインを変更したりすることが簡単になります。

演習

1. `man` コマンドを使って、`apropos` に簡単な使用方法を表示させるオプションを調べてみましょう。

2. `man` コマンドを使って、`grep` コマンドの著作権を調べてみましょう。

発展演習

1. 利用中のシステムのハードウェアアーキテクチャーと、Linuxカーネルのバージョンを表示してみましょう。

2. 動的な `history` データベースと `.bash_history` ファイルの最後の20行をそれぞれ表示して、それらを比べてみましょう。

3. `apropos` コマンドを使って、接続されている物理ブロックデバイスのサイズを調べるコマンドを探し、その `man` ページを参照してメガバイトやギガバイトではなくバイト単位でサイズを表示するコマンドとオプションを見つけてください。

まとめ

このレッスンでは、次のことを学びました。

- 自分がいるファイルシステムの位置や、OSに関する情報を取得する方法。
- コマンドの使用法に関するマニュアルを見つける方法。
- コマンドのファイルシステム上の位置と、種類を確認する方法。
- 以前に実行したコマンドを見つけて再利用する方法。

このレッスンでは、次のコマンドを説明しました:

`pwd`
現在の作業ディレクトリのパスを出力します。

`uname`
システムのハードウェアアーキテクチャ、Linuxカーネルのバージョン、ディストリビューション、およびディストリビューションリリースを出力します。

`man`
コマンドの使用法を文書化したヘルプファイルにアクセスします。

`type`
コマンドのファイルシステム上の位置や種類を出力します。

`which`
コマンドのファイルシステム上の位置を出力します。

`history`
以前に実行したコマンドを表示または操作します。

演習の解答

1. `man` コマンドを使って、`apropos` に簡単な使用方法を表示させるオプションを調べてみましょう。

   `man apropos` を実行し、“Options” セクションの `--usage` までスクロールします。

2. `man` コマンドを使って、`grep` コマンドの著作権を調べてみましょう。

   `man grep` を実行し、ドキュメントの “Copyright” セクションまでスクロールします。このプログラムはフリーソフトウェアファウンデーションが著作権者であり、そのライセンス（GPL）を使用しています。

発展演習の解答

1. 利用中のシステムのハードウェアアーキテクチャーと、Linuxカーネルのバージョンを表示してみましょう。

   `man uname` を実行して“Description”セクションを読むと、必要な結果のみを表示するオプションが分かります。`-v` がカーネルバージョンを、`-i` がハードウェアアーキテクチャを表示します。

   ```
   $ man uname
   $ uname -v
   $ uname -i
   ```

2. 動的な `history` データベースと `.bash_history` ファイルの最後の20行をそれぞれ表示して、それらを比較してください。

   ```
   $ history 20
   $ tail -n 20 .bash_history
   ```

3. `apropos` コマンドを使って、接続されている物理ブロックデバイスのサイズを調べるコマンドを探し、その `man` ページを参照してメガバイトやギガバイトではなくバイト単位でサイズを表示するコマンドとオプションを見つけてください。

   ひとつの方法として、`apropos` に `block` を指定して実行し、結果を見ていきます。`lsblk` がブロックデバイスをリストするコマンド（第8章の管理者コマンド）ですから、多分、これが探しているものだと分かります。`man lsblk` を実行して“Description”セクションを見ると、`-b` がデバイスサイズをバイト単位で表示することが分かります。`lsblk -b` を実行して、結果を確認しましょう

   ```
   $ apropos block
   $ man lsblk
   $ lsblk -b
   ```

Linux Professional Institute

103.1 レッスン 2

Certificate:	LPIC-1
Version:	5.0
Topic:	103 GNUおよびUnixコマンド
Objective:	103.1 コマンドラインでの作業
Lesson:	2 of 2

はじめに

オペレーティングシステムには、作業を行うために必要コマンドラインシェルやGUIなどの基本的なツールが備わっています。それらをユーザーが使用する場合には、さまざまな「変数」にショートカットや規定値を格納して、ユーザー毎の「環境」が定義されます。ここでは、それら変数の値を表示、参照、管理する方法を学習します。

環境変数の検索

最初に、環境を定義する変数の値を調べる方法を説明します。1つは、`env`　コマンドを使用する方法です。

```
$ env
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
XDG_RUNTIME_DIR=/run/user/1000
XAUTHORITY=/run/user/1000/gdm/Xauthority
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/l
ocal/games:/snap/bin
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
[...]
```

実際には、この例よりもはるかに多くの出力が得られるでしょう。この中では、シェルや他のプログラムが（フルパス指定なしで）別のプログラムを検索するディレクトリを含む PATH エントリが最も重要です。この指定によって、たとえばホームディレクトリに居ても、/usr/local/bin にあるバイナリプログラムをコマンド名だけで実行できます。

次の方法に進みましょう。echo コマンドは、ユーザーが指示した内容を画面に出力します。このコマンドが何の役に立つのか、と思うかもしれませんが、何かを文字通り echo したい場面は少なくありません。

```
$ echo "Hi. How are you?"
Hi. How are you?
```

echo は文字列を表示するだけではありません。変数であることを示す $ に続けて変数名を入力すると、シェルが変数を展開して、変数の名前ではなくその値を教えてくれます。例えば、現在のPATHにお気に入りのディレクトリが含まれているかどうかわからない場合には、echo を実行すると簡単に確認できます。

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/
games:/snap/bin
```

新しい変数の作成

独自の変数を環境に追加することもできます。最も簡単な方法は、= を使用することです。左側の文字列が新しい変数の名前になり、右側の文字列がその値になります。変数名を echo に与えれば、動作を確認できます。

```
$ myvar=hello
$ echo $myvar
hello
```

NOTE 変数に値を割り当てる時は、等号の両側にスペースを置かないことに注意してください。

作成した変数は正しく機能しましたか？ ターミナルに bash と入力すると、新しいシェルが起動します。この新しいシェルは、それまで使用していたものとまったく同じように見えますが、実際には元のシェル（親）の 子 です。次に、この新しい子シェルの中で、echo を使って、作成した変数を表示してみましょう。今度は、結果に何も表示されません。何が起きているのでしょうか？

```
$ bash
```

```
$ echo $myvar

$
```

`=` で作成した変数は、作成したシェル ～ つまり変数を作成したシェルセッション内でのみ使用可能です。新しいシェルを起動した場合、あるいは `exit` を使用してセッションを閉じた場合は、この変数の値は引き継がれません。ここで `exit` と入力すると、元の親シェルに戻ることができます。これは、`bash` を入力する前に作業していたシェルです。もう一度 `echo $myvar` を実行すると、変数がまだ有効であることを確認できます。`export myvar` と入力すると、後で開く子シェルに変数を引き継ぐことができます。再度 `bash` と入力し、新しいシェルを起動した後に、`echo $myvar` と入力してみてください。今度は変数の値が表示されます。通常の変数が、シェルの「環境」に追加されたので、子シェルにも引き継がれたのです。通常の変数を「シェル変数」、exportされた変数を「環境変数」と呼びます。

```
$ exit
$ export myvar
$ bash
$ echo $myvar
hello
```

何の目的もなく子シェルを作っているときは、このようなことは少しばかばかしいと感じるかもしれません。しかし、本格的なスクリプトを書き始めると、システムで変数がどのように伝搬されるかを理解することが非常に重要になってきます。

環境変数の削除

作成した一時的な変数を削除する方法も知っておきましょう。1つ目の方法は、親シェルを閉じるか、コンピューターを再起動することです。2つ目の方法は、`unset` を使って（親シェルを閉じたり、コンピューターを再起動すること無く）変数を削除することです。変数名に `$` を付けずに `unset` と入力してみてください。変数が削除されます。削除されたことは、`echo` で確認できます。

```
$ unset myvar
$ echo $myvar

$
```

`unset` コマンドがあるということは、それに対応する `set` というコマンドも存在します。`set` を単独で実行すると多くの出力が得られますが、その中には環境変数も含まれています。`PATH` を取り出してみましょう。

```
$ set | grep PATH
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/l
ocal/games:/snap/bin
[...]
```

`set` と `env` の違いは何でしょう？ `set` は、すべての変数や関数を出力します。試しに `mynewvar` という新しい変数を作成して確認してみましょう。

```
$ mynewvar=goodbye
$ echo $mynewvar
goodbye
```

ここで、`grep` を使用して文字列 `mynewvar` を取り出しながら、`env` と `set` を実行してみましょう。`env` では（exportしていないので）何も表示されませんが、`set` ではシェル変数が表示されます。

```
$ env | grep mynewvar

$ set | grep mynewvar
mynewvar=goodbye
```

特殊文字をエスケープするための引用符

続いて、特殊文字について解説します。英数字（a-zおよび0-9）は通常、Bashによって文字通り解釈されます。`myfile` という名前の新しいファイルを作成する場合、`touch` に続けて `myfile` と入力すれば、Bashはその処理方法を認識します。ただし、ファイル名に特殊文字を含めたい場合は、少し工夫する必要があります。

説明のために、`touch` と入力し、その後に `my big file` という名前を続けてみましょう。Bashが解釈する単語の間に2つのスペースがあることが問題になります。技術的には、スペースを “文字” とは呼びませんが、Bashが文字通りに解釈しないという意味では特殊文字に似ています。現在のディレクトリ内容を一覧表示すると、`my big file` という1つのファイルではなく、`my`、`big`、`file` という名前の3つのファイルが作成されます。これは、Bashが、リストで渡した名前を「複数個のファイルを作成したい」と解釈したためです。

```
$ touch my big file
$ ls
my big file
```

同様に、3つのファイルすべてを1つのコマンドで削除（`rm`）すると、スペースが同じように解釈されます。

```
$ rm my big file
```

それでは、正しい方法を試してみましょう。`touch` に続けてファイル名の3つの部分を入力しますが、名前をダブルクオートで囲みます。今度はうまくいきました。ディレクトリの内容を一覧表示すると、意図した名前のファイルが1つ作成されます。

```
$ touch "my big file"
$ ls
'my big file'
```

同じ効果を得る別の方法もあります。たとえば、シングルクオート（一重引用符）はダブルクォート（二重引用符）と同じように機能します。シングルクォートはすべての文字をそのままの“文字”として処理しますが、ダブルクォートは `$`,`,\`、と、場合によっては `!` を“特殊文字”として処理することに注意してください。

```
$ rm 'my big file'
```

特殊文字の前にバックスラッシュを付けると、その文字の特別な意味が“エスケープ”されて、Bashは普通の文字として解釈します。

```
$ touch my\ big\ file
```

演習

1. `export` コマンドを使用して、PATH変数に新しいディレクトリを追加してください（再起動後には無くなっても構いません）。

2. `unset` コマンドを使用して `PATH` 変数を削除してください。その後、`sudo` を使用してコマンド（`sudo cat /etc/shadow` など）を実行してみましょう。どうなりますか？またそれは、どうしてですか？（シェルを終了すれば、元の状態に戻ります。）

発展演習

1. インターネットを検索して、特殊文字の完全なリストを調べてください。

2. 特殊文字で構成された文字列をさまざまな方法でをエスケープして、コマンドを実行してください。エスケープ方法によって動作に違いはありますか？

まとめ

このレッスンでは、次のことを学びました。

- システムの環境変数を確認する方法。
- 独自の変数を作成して他のシェルにエクスポートする方法。
- 変数を削除する方法と、`env` コマンドと `set` コマンドの違い。
- Bashが文字通りに解釈するように特殊文字をエスケープする方法。

このレッスンでは、次のコマンドを説明しました:

`echo`
入力した文字列や変数を出力します。

`env`
環境変数を調べたり、（一時的に）変更したりします。

`export`
シェル変数を環境変数に変換します。

`unset`
シェル変数や関数を削除します。

演習の解答

1. `export` コマンドを使用して、PATH変数に新しいディレクトリを追加してください（再起動後には無くなっても構いません）。

   `export PATH="/home/yourname/myfiles:$PATH"` によって、一時的に新しいディレクトリ（ホームディレクトリにある `myfiles` というディレクトリ）をPATHに追加することができます。`myfiles/` ディレクトリに簡単なスクリプトを作成し、実行権限を付与して、別のディレクトリから実行してみてください。なお、作業ディレクトリはホームディレクトリであることを想定しています。

   ```
   $ touch myfiles/myscript.sh
   $ echo '#!/bin/bash' >> myfiles/myscript.sh
   $ echo 'echo Hello' >> myfiles/myscript.sh
   $ chmod +x myfiles/myscript.sh
   $ myscript.sh
   Hello
   ```

2. `unset` コマンドを使用して `PATH` 変数を削除してください。その後、`sudo` を使用してコマンド（`sudo cat /etc/shadow` など）を実行してみましょう。どうなりますか？またそれは、どうしてですか？（シェルを終了すれば、元の状態に戻ります。）

   `unset PATH` と入力すると、PATH変数が消去されます。そのため、絶対パスを指定せずにプログラムを呼び出そうとしても失敗します。例えば、sudo（それ自体は `/usr/bin/sudo` にあるバイナリプログラム）を使用してコマンドを実行しようとすると失敗します。つまり、`/usr/bin/sudo /bin/cat /etc/shadow` のように絶対パスを指定しない限り実行できません。`PATH` をリセットするには、`export` を使用してPATH変数を再設定するか、いったんシェルを終了します。

発展演習の解答

1. インターネットを検索して、特殊文字の完全なリストを調べてください。

   次の通りです: `& ; | * ? " ' [ ] ( ) $ < > { } # / \ ! ~`

2. 特殊文字で構成された文字列をさまざまな方法でをエスケープして、コマンドを実行してください。エスケープ方法によって動作に違いはありますか？

   文字 `"` を使用してエスケープすると、ドル記号、バッククォート、およびバックスラッシュの特別な意味は無くなりません。一方、`'` 文字を使用してエスケープすると、すべての 文字がそのまま“文字”として解釈されます。

```
$ echo "$mynewvar"
goodbye
$ echo '$mynewvar'
$mynewvar
```

103.2 フィルターを使用してテキストストリームを処理する

LPI目標への参照

LPIC-1 v5, Exam 101, Objective 103.2

総重量

2

主な知識分野

- テキストファイルと出力ストリームをテキストユーティリティフィルタで送信して、GNU　textutilsパッケージにある標準のUNIXコマンドを使用して出力を変更します

用語とユーティリティ

- `bzcat`
- `cat`
- `cut`
- `head`
- `less`
- `md5sum`
- `nl`
- `od`
- `paste`
- `sed`
- `sha256sum`
- `sha512sum`
- `sort`

- `split`
- `tail`
- `tr`
- `uniq`
- `wc`
- `xzcat`
- `zcat`

103.2 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	103 GNUおよびUnixコマンド
Objective:	103.2 テキストストリームをフィルターで処理する
Lesson:	1 of 1

はじめに

すべてのシステム管理者にとって、テキストの処理は重要な仕事の一つです。Unix開発チームのオリジナルメンバーであるDoug McIlroyは、Unix哲学を要約し、（とりわけ重要なこととして）“テキストストリームを扱うプログラムを書きなさい、なぜならそれが普遍的なインターフェイスだからです。” と述べています。LinuxはUnixオペレーティングシステムに触発され、その哲学をしっかりと引き継いでいるため、Linuxディストリビューションには多くのテキスト操作ツールが含まれています。

リダイレクトとパイプの簡単な紹介

（※訳注: 詳細は「103.4 ストリームとパイプ、リダイレクトを使う」で説明します。）

Unix哲学より抜粋;

- 1つのことをうまく実行するプログラムを作成しよう。
- 連携できるようにプログラムを作成しよう。

プログラムを連携させる主な方法の1つは、パイプ と リダイレクト を使用することです。ほとんどすべてのテキスト操作プログラムは、標準入力（stdin）からテキストを取得

し、処理結果を標準出力（stdout）に出力し、さらにエラーメッセージを標準エラー（stderr）に出力します。特に指定しない限り、標準入力はキーボードからの入力です（Enterキーを押すたびにプログラムが読み取ります）。標準出力とエラー出力が、端末画面に表示されます。これらの働きを見ていきましょう。

ターミナルで `cat` と入力し、Enterキーを押します。次に、ランダムなテキストを入力します。

```
$ cat
This is a test
This is a test
Hey!
Hey!
It is repeating everything I type!
It is repeating everything I type!
(I will hit ctrl+c so I will stop this nonsense)
(I will hit ctrl+c so I will stop this nonsense)
^C
```

`cat` コマンド（“concatenate” に由来）の詳細は、manページを参照してください。

NOTE

最小構成でインストールしたLinuxサーバーでは、`info` や `less` など一部のコマンドが使用できないことがあります。そのような場合は、対応するレッスン（102.4および102.5）を参照して、システムに応じた方法でツールをインストールしてください。

例に示したように、`cat` で読み取り先を指定しない場合、入力した内容（標準入力）は、そのままターミナルウィンドウ（標準出力）に出力されます。

では、次を試してみましょう。

```
$ cat > mytextfile
This is a test
I hope cat is storing this to mytextfile as I redirected the output
I will hit ctrl+c now and check this
^C

$ cat mytextfile
This is a test
I hope cat is storing this to mytextfile as I redirected the output
I will hit ctrl+c now and check this
```

`>`（大なり記号）は、`cat` の標準出力を `mytextfile` ファイルに送るように（シェルに）

指示します。続いてこれを試してみましょう。

```
$ cat mytextfile > mynewtextfile
$ cat mynewtextfile
This is a test
I hope cat is storing this to mytextfile as I redirected the output
I will hit ctrl+c now and check this
```

これは、`mytextfile` を `mynewtextfile` にコピーします。`diff` を実行することで、これら2つのファイルの内容が同じであることを確認できます。

```
$ diff mynewtextfile mytextfile
```

実行結果に何も出力されないので、ファイルは同一です。次に、追記リダイレクト（`>>`）を試してみましょう。

```
$ echo 'This is my new line' >> mynewtextfile
$ diff mynewtextfile mytextfile
4d3
< This is my new line
```

ここまでは、リダイレクトを使用してファイルを作成および操作してきました。次に、パイプ（記号 `|`）を使用して、あるプログラムの出力を別のプログラムにリダイレクトしてみましょう。“this” という単語が含まれている行を見つけてみます。

```
$ cat mytextfile | grep this
I hope cat is storing this to mytextfile as I redirected the output
I will hit ctrl+c now and check this

$ cat mytextfile | grep -i this
This is a test
I hope cat is storing this to mytextfile as I redirected the output
I will hit ctrl+c now and check this
```

この例では、`cat` の標準出力を、別のコマンド `grep` にパイプしています。`-i` オプションで大文字、小文字を区別しないように指示すると、出力結果が変化します。

テキストストリームの処理

圧縮ファイルの読み取り

以下のコマンドリストを含む `ftu.txt` というファイルを作成します。（訳注: ファイル名 `ftu` は出題範囲にあるタイトル files, terms and utilities に由来しています。すなわち、このレッスンで出題される可能性が高いコマンド名のリストです。）

```
bzcat
cat
cut
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
sort
split
tail
tr
uniq
wc
xzcat
zcat
```

次に、`grep` コマンドを使用して、文字列 `cat` を含むすべての行を出力します。

```
$ cat ftu.txt | grep cat
bzcat
cat
xzcat
zcat
```

`grep` コマンドでテキストを直接フィルタリングしても、同じ結果が得られます。`cat` コマンドでテキストストリームを `stdout` に送る必要はありません。

```
$ grep cat ftu.txt
bzcat
cat
xzcat
```

```
zcat
```

NOTE Linuxでは、同じタスクを処理する方法がたくさんあることを覚えておいてください。

ファイルを圧縮するコマンドには `gzip`、`bzip2`、`xz` などがありますが、それぞれに圧縮ファイルの内容を表示するコマンドが用意されています。`gzip` には `zcat`、`bzip2` には `bzcat`、`xz` には `xzcat` がそれぞれ対応します。

新しく作成したファイル `ftu.txt` がディレクトリにあることを確認してから、`gzip` 圧縮ファイルを作成します。

```
$ ls ftu*
ftu.txt

$ gzip ftu.txt
$ ls ftu*
ftu.txt.gz
```

次に、`zcat` コマンドを使用して、gzip圧縮された圧縮ファイルの内容を表示してみましょう。

```
$ zcat ftu.txt.gz
bzcat
cat
cut
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
sort
split
tail
tr
uniq
wc
xzcat
```

```
zcat
```

`gzip` は `ftu.txt` を `ftu.txt.gz` に圧縮した後、元のファイルを削除することに注意してください。デフォルトでは、`gzip` コマンドの処理内容は表示されません。`gzip` の処理内容を確認したい場合は、`-v` オプション（“verbose”）で冗長な出力を指定します。

ページャーでファイルを表示する

`cat` コマンドの後にファイルを指定すると、その内容を標準出力に送ることはご存じでしょう。ファイル `/var/log/syslog` には、Linuxシステムで起きている重要な事柄が保存されます。（`sudo` コマンドを使用して特権を得ると）`/var/log/syslog` ファイルを読み取れます。（訳注: /var/log/syslogのパーミッションはディストリビューションによって異なり、多くの場合は一般ユーザーでも読み出せますから、その場合は `sudo` は必要ありません。）

```
$ sudo cat /var/log/syslog
```

…ターミナルウィンドウ内でメッセージが非常に高速にスクロールすることでしょう。
出力プログラムの1つである `less` にパイプすると、結果をページ送りすることができるようになります。`less` を使用すると、メッセージをスクロールしたり、`vi` エディタを使った時のようにテキスト全体を見渡したり検索することができます。（訳注: `less` の操作方法は、`vi` エディタとほぼ同じです。103.8 を参照してください。）

もっとも、`cat` コマンドをページャーにパイプするよりも、ページャーを直接使用する方が実用的です。

```
$ sudo less /var/log/syslog
... (□□□□)
```

テキストファイルの一部を取り出す

ファイルの先頭ないし末尾のみを取り出したい時には、別のコマンドを使います。`head` コマンドは、デフォルトでファイルの先頭10行を読み出し、`tail` コマンドは、デフォルトでファイルの末尾10行を読み出します。試してみましょう。

```
$ sudo head /var/log/syslog
Nov 12 08:04:30 hypatia rsyslogd: [origin software="rsyslogd" swVersion="8.1910.0"
x-pid="811" x-info="https://www.rsyslog.com"] rsyslogd was HUPed
Nov 12 08:04:30 hypatia systemd[1]: logrotate.service: Succeeded.
Nov 12 08:04:30 hypatia systemd[1]: Started Rotate log files.
Nov 12 08:04:30 hypatia vdr: [928] video directory scanner thread started (pid=882,
tid=928, prio=low)
```

```
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'A - ATSC'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'C - DVB-C'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'S - DVB-S'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'T - DVB-T'
Nov 12 08:04:30 hypatia vdr[882]: vdr: no primary device found - using first
device!
Nov 12 08:04:30 hypatia vdr: [929] epg data reader thread started (pid=882,
tid=929, prio=high)
$ sudo tail /var/log/syslog
Nov 13 10:24:45 hypatia kernel: [ 8001.679238] mce: CPU7: Core temperature/speed
normal
Nov 13 10:24:46 hypatia dbus-daemon[2023]: [session uid=1000 pid=2023] Activating
via systemd: service name='org.freedesktop.Tracker1.Miner.Extract' unit='tracker-
extract.service' requested by ':1.73' (uid=1000 pid=2425
comm="/usr/lib/tracker/tracker-miner-fs ")
Nov 13 10:24:46 hypatia systemd[2004]: Starting Tracker metadata extractor...
Nov 13 10:24:47 hypatia dbus-daemon[2023]: [session uid=1000 pid=2023] Successfully
activated service 'org.freedesktop.Tracker1.Miner.Extract'
Nov 13 10:24:47 hypatia systemd[2004]: Started Tracker metadata extractor.
Nov 13 10:24:54 hypatia kernel: [ 8010.462227] mce: CPU0: Core temperature above
threshold, cpu clock throttled (total events = 502907)
Nov 13 10:24:54 hypatia kernel: [ 8010.462228] mce: CPU4: Core temperature above
threshold, cpu clock throttled (total events = 502911)
Nov 13 10:24:54 hypatia kernel: [ 8010.469221] mce: CPU0: Core temperature/speed
normal
Nov 13 10:24:54 hypatia kernel: [ 8010.469222] mce: CPU4: Core temperature/speed
normal
Nov 13 10:25:03 hypatia systemd[2004]: tracker-extract.service: Succeeded.
```

表示される行数をわかりやすくするために、`head` コマンドの出力を `nl` コマンドにパイプすると、コマンドに送られたテキストに行番号が表示されます。

```
$ sudo head /var/log/syslog | nl
1   Nov 12 08:04:30 hypatia rsyslogd: [origin software="rsyslogd"
swVersion="8.1910.0" x-pid="811" x-info="https://www.rsyslog.com"] rsyslogd was
HUPed
2   Nov 12 08:04:30 hypatia systemd[1]: logrotate.service: Succeeded.
3   Nov 12 08:04:30 hypatia systemd[1]: Started Rotate log files.
4   Nov 12 08:04:30 hypatia vdr: [928] video directory scanner thread started
(pid=882, tid=928, prio=low)
5   Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'A - ATSC'
6   Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'C - DVB-C'
7   Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'S - DVB-S'
8   Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'T - DVB-T'
```

```
9   Nov 12 08:04:30 hypatia vdr[882]: vdr: no primary device found - using first
device!
10  Nov 12 08:04:30 hypatia vdr: [929] epg data reader thread started (pid=882,
tid=929, prio=high)
```

また、`tail` コマンドの出力を `wc` コマンドにパイプすることで、行数やワード数などを確認することができます。このコマンドに `-l` オプションを指定すると、行数のみをカウントします。

```
$ sudo tail /var/log/syslog | wc -l
10
```

ファイルの先頭または末尾から、より多く（あるいは少なく）の行を取り出したいときは、`-n` オプションに取り出す行数を指定します。

```
$ sudo tail -n 5 /var/log/syslog
Nov 13 10:37:24 hypatia systemd[2004]: tracker-extract.service: Succeeded.
Nov 13 10:37:42 hypatia dbus-daemon[2023]: [session uid=1000 pid=2023] Activating
via systemd: service name='org.freedesktop.Tracker1.Miner.Extract' unit='tracker-
extract.service' requested by ':1.73' (uid=1000 pid=2425
comm="/usr/lib/tracker/tracker-miner-fs ")
Nov 13 10:37:42 hypatia systemd[2004]: Starting Tracker metadata extractor...
Nov 13 10:37:43 hypatia dbus-daemon[2023]: [session uid=1000 pid=2023] Successfully
activated service 'org.freedesktop.Tracker1.Miner.Extract'
Nov 13 10:37:43 hypatia systemd[2004]: Started Tracker metadata extractor.
$ sudo head -n 12 /var/log/syslog
Nov 12 08:04:30 hypatia rsyslogd: [origin software="rsyslogd" swVersion="8.1910.0"
x-pid="811" x-info="https://www.rsyslog.com"] rsyslogd was HUPed
Nov 12 08:04:30 hypatia systemd[1]: logrotate.service: Succeeded.
Nov 12 08:04:30 hypatia systemd[1]: Started Rotate log files.
Nov 12 08:04:30 hypatia vdr: [928] video directory scanner thread started (pid=882,
tid=928, prio=low)
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'A - ATSC'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'C - DVB-C'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'S - DVB-S'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'T - DVB-T'
Nov 12 08:04:30 hypatia vdr[882]: vdr: no primary device found - using first
device!
Nov 12 08:04:30 hypatia vdr: [929] epg data reader thread started (pid=882,
tid=929, prio=high)
Nov 12 08:04:30 hypatia vdr: [882] no DVB device found
Nov 12 08:04:30 hypatia vdr: [882] initializing plugin: vnsiserver (1.8.0): VDR-
```

```
Network-Streaming-Interface (VNSI) Server
```

ストリームエディタ sedの基本

今度は名前に `cat` が含まれていない単語を見てみましょう。`grep` に `-v` オプションを指定して、`cat` を 含まない 行のみを出力します。

```
$ zcat ftu.txt.gz | grep -v cat
cut
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
sort
split
tail
tr
uniq
wc
```

`grep` でできることのほとんどは、`sed` でも行えます。このコマンドは、テキストをフィルタしたり変換するためのストリームエディタです（`sed` のマニュアルページにそう書かれています）。まず、`gzip` 圧縮ファイルを伸張して、元の `ftu.txt` ファイルを復元します。

```
$ gunzip ftu.txt.gz
$ ls ftu*
ftu.txt
```

次のように、`sed` で文字列 `cat` を含む行のみを表示できます。

```
$ sed -n /cat/p < ftu.txt
bzcat
cat
xzcat
zcat
```

小なり記号 `<` を使用して、ファイル `ftu.txt` の内容を `sed` コマンドに送っています。スラッシュで囲まれた単語（つまり、`cat`）が、検索対象となる単語です。`-n` オプションは `sed` に、（`p` コマンドで指示された場合を除いて）何も出力しないように指示するものです。同じコマンドを `-n` オプションなしで実行するとどうなるか見てみましょう。

```
$ sed /cat/d < ftu.txt
cut
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
sort
split
tail
tr
uniq
wc
```

`-n` オプションを指定しない場合、`sed` は `d` コマンドによって削除を指示された行（すなわち `cat` を含む行）を除き、すべての行を表示します。

`sed` の最も一般的な使用法は、ファイルから指定したテキストを見つけて置き換えることです。例えば、すべての `cat` を `dog` に変更するとしましょう。`sed` でこれを行うには、`s` コマンド（substituteに由来）を指定して、置換対象の `cat` を、2番目のワード `dog` に置換します。

```
$ sed s/cat/dog/ < ftu.txt
bzdog
dog
cut
head
less
md5sum
nl
od
paste
sed
sha256sum
```

```
sha512sum
sort
split
tail
tr
uniq
wc
xzdog
zdog
```

リダイレクト（<）で `ftu.txt` ファイルを `sed` コマンドに渡すのではなく、`sed` コマンドでファイルを直接操作することもできます。元のファイルのバックアップを作成してから、直接操作を試してみましょう。

```
$ sed -i.backup s/cat/dog/ ftu.txt
$ ls ftu*
ftu.txt  ftu.txt.backup
```

`sed` に `-i` オプション（in-placeに由来）を指定すると、指定したファイルを直接操作して置き換えます。例のように、`-i` オプションに追加の引数を指定すると、そこに指定した文字列をサフィックスとしたファイルにバックアップファイルを作成してから、元のファイルを操作して置き換えます。（訳注: -iオプションと引数文字列の間に空白を置いてはいけません。）

データの同一性を保証する

Linuxでファイルを操作することがいかに簡単であるかを示しました。次に、ファイルを他の人に配布する場合に、受け取った相手が元のファイルと同じであることを確認する方法を解説します。LinuxディストリビューションのダウンロードCD/DVDイメージと共に、そのディスクイメージのハッシュ値を含むファイルをサーバーにホストする場合などに、この手法を使います。以下はDebianのダウンロードミラーからのリスト例です。

```
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
[PARENTDIR] Parent Directory                                    -
[SUM]       MD5SUMS                             2019-09-08 17:46 274
[CRT]       MD5SUMS.sign                        2019-09-08 17:52 833
[SUM]       SHA1SUMS                            2019-09-08 17:46 306
[CRT]       SHA1SUMS.sign                       2019-09-08 17:52 833
[SUM]       SHA256SUMS                          2019-09-08 17:46 402
[CRT]       SHA256SUMS.sign                     2019-09-08 17:52 833
[SUM]       SHA512SUMS                          2019-09-08 17:46 658
[CRT]       SHA512SUMS.sign                     2019-09-08 17:52 833
[ISO]       debian-10.1.0-amd64-netinst.iso     2019-09-08 04:37 335M
```

```
[ISO]       debian-10.1.0-amd64-xfce-CD-1.iso    2019-09-08 04:38 641M
[ISO]       debian-edu-10.1.0-amd64-netinst.iso  2019-09-08 04:38 405M
[ISO]       debian-mac-10.1.0-amd64-netinst.iso  2019-09-08 04:38 334M
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
```

上記のリストでは、Debianのインストールイメージファイルに対応する、さまざまなアルゴリズム（MD5、SHA1、SHA256、SHA512）によるハッシュ値を格納したテキストファイルが付属しています。

> **NOTE** ハッシュ値とは、ファイルに対して暗号化ハッシュ関数に基づく計算を行って算出された値です。強度が異なるさまざまなタイプの暗号化ハッシュ関数があり、LPIC-1では `md5sum`、`sha256sum`、`sha512sum` を使いこなせることが求められます。（訳注: ハッシュ値の用途は「チェックサム」に似ています。数学的により複雑な「ハッシュ関数」を使用して計算するので「ハッシュ値」、あるいは「メッセージダイジェスト」、「フィンガープリント」などと呼びます。）

ファイル（たとえば、`debian-10.1.0-amd64-netinst.iso` イメージ）をダウンロードしたら、ダウンロードしたファイルのハッシュ値を計算し、提供されたハッシュ値と比較します。

次の例でポイントを説明しましょう。`sha256sum` コマンドを使用して、`ftu.txt` ファイルのSHA256ハッシュ値を計算します。

```
$ sha256sum ftu.txt
345452304fc26999a715652543c352e5fc7ee0c1b9deac6f57542ec91daf261c  ftu.txt
```

ファイル名の前にある長い文字列が、このテキストファイルのSHA256ハッシュ値です。その値を含むファイルを作成して、`ftu.txt` が変更されていないことを検証しましょう。`sha256sum` コマンドの出力をファイルにリダイレクトして、ハッシュ値のファイルを作成します。

```
$ sha256sum ftu.txt > sha256.txt
```

次に、`ftu.txt` ファイルが変更されていないことを確認するために、`sha256sum` コマンドに `-c` オプションとハッシュ値を含むファイル名を指定します。

```
$ sha256sum -c sha256.txt
ftu.txt: OK
```

このように、ファイルに含まれるハッシュ値と、`ftu.txt` ファイルに対して計算し

たSHA256ハッシュ値が一致していることが示されます。元のファイルが変更された場合（ファイルのダウンロード中に数バイトが失われた場合や、誰かがファイルを改ざんした場合など）は、値のチェックが失敗します。失敗した場合は、ファイルが不良または破損している可能性が高く、内容の同一性が損なわれています。確認するために、`ftu.txt` の末尾にテキストを追加して試してみましょう。

```
$ echo "new entry" >> ftu.txt
```

再度、ファイルの同一性を検証してみます。

```
$ sha256sum -c sha256.txt
ftu.txt: FAILED
sha256sum: WARNING: 1 computed checksum did NOT match
```

計算した値と、ハッシュファイルの内容が異なってことがわかります。つまり、このファイルの内容が変更されたということです。ファイルの重要度に応じて、ファイルのコピーをもう一回ダウンロードしたり、チェックサムの不一致をファイルの送信者に連絡したり、データセンターのセキュリティチームに報告しましょう。

ファイルを詳しく調べる

`od`（8進ダンプ）コマンドを使って、アプリケーションやさまざまなファイルをデバッグすることがよくあります。オプションを指定しない `od` コマンドは、ファイルの内容を8進数でリストします。先程作成した `ftu.txt` ファイルでやってみましょう。

```
$ od ftu.txt
0000000 075142 060543 005164 060543 005164 072543 005164 062550
0000020 062141 066012 071545 005163 062155 071465 066565 067012
0000040 005154 062157 070012 071541 062564 071412 062145 071412
0000060 060550 032462 071466 066565 071412 060550 030465 071462
0000100 066565 071412 071157 005164 070163 064554 005164 060564
0000120 066151 072012 005162 067165 070551 073412 005143 075170
0000140 060543 005164 061572 072141 000012
0000151
```

出力の最初の列は、ファイル先頭からの　オフセット　を8進数で示した数値です。`od`　はデフォルトで情報を8進数で出力しますから、各行はバイト単位のオフセット（8進数）で始まり、ファイル内容を2バイトずつ8進数で示した値が8つ続きます。つまり、1行には16バイト分の内容が表示されます。なお、2バイトの順序はCPUアーキテクチャによって異なっています。

TIP　1 バイト は8ビットです。

`-x` オプションを指定すると、ファイルの内容が16進数で表示されます。（オフセットは8進数のままです）。

```
$ od -x ftu.txt
0000000 7a62 6163 0a74 6163 0a74 7563 0a74 6568
0000020 6461 6c0a 7365 0a73 646d 7335 6d75 6e0a
0000040 0a6c 646f 700a 7361 6574 730a 6465 730a
0000060 6168 3532 7336 6d75 730a 6168 3135 7332
0000100 6d75 730a 726f 0a74 7073 696c 0a74 6174
0000120 6c69 740a 0a72 6e75 7169 770a 0a63 7a78
0000140 6163 0a74 637a 7461 000a
0000151
```

このように、バイトオフセットの後の8列それぞれが、16進数で表示されます。

`od` コマンドは、スクリプトのデバッグなどに便利です。例えば、`od` コマンドは、ファイル内の 改行（newline）などの、通常は見えない文字を見えるようにします。`-c` オプションを指定すると、数値（文字コード）ではなく、（エスケープされた）文字で表示します。

```
$ od -c ftu.txt
0000000   b   z   c   a   t  \n   c   a   t  \n   c   u   t  \n   h   e
0000020   a   d  \n   l   e   s   s  \n   m   d   5   s   u   m  \n   n
0000040   l  \n   o   d  \n   p   a   s   t   e  \n   s   e   d  \n   s
0000060   h   a   2   5   6   s   u   m  \n   s   h   a   5   1   2   s
0000100   u   m  \n   s   o   r   t  \n   s   p   l   i   t  \n   t   a
0000120   i   l  \n   t   r  \n   u   n   i   q  \n   w   c  \n   x   z
0000140   c   a   t  \n   z   c   a   t  \n
0000151
```

ファイル内の改行は、文字 `\n` で示されます。ファイル内のすべての文字を表示するだけで、バイトオフセットを表示する必要がない場合は、次のようにバイトオフセット列を出力から除外できます。

```
$ od -An -c ftu.txt
   b   z   c   a   t  \n   c   a   t  \n   c   u   t  \n   h   e
   a   d  \n   l   e   s   s  \n   m   d   5   s   u   m  \n   n
   l  \n   o   d  \n   p   a   s   t   e  \n   s   e   d  \n   s
   h   a   2   5   6   s   u   m  \n   s   h   a   5   1   2   s
   u   m  \n   s   o   r   t  \n   s   p   l   i   t  \n   t   a
   i   l  \n   t   r  \n   u   n   i   q  \n   w   c  \n   x   z
   c   a   t  \n   z   c   a   t  \n
```

演習

1. 学校にノートパソコンが寄付されたので、そのノートパソコンにLinuxをインストールしたいと思っています。OSが起動せずマニュアルがないため、LinuxをUSBブートしてノートパソコンのCPU情報を確認することにしました。シェルターミナルを起動して、/proc/cpuinfo ファイルの中身を確認しました:

```
processor   : 0
vendor_id   : GenuineIntel
cpu family  : 6
model       : 158

(□□)

processor   : 1
vendor_id   : GenuineIntel
cpu family  : 6
model       : 158

(□□)
```

   - コマンド `grep` と `wc` を使用して、ノートパソコンのプロセッサ数を表示してください。

   - `grep` の代わりに `sed` を使用して、同じ内容を表示してください。

2. コマンド `grep`、`sed`、`head`、`tail` を使用して、`/etc/passwd` ファイルから、指定された内容を表示してください。
   - Bashシェルを利用するユーザをリストしてください。

   - システムには、プログラムを処理ないし管理するためのユーザが存在します。これらのユーザはシェルを使用しません。システムに存在する、そのようなユーザをカウントしてください。

   - `/etc/passwd` ファイルのみを参照して、システムに存在するユーザーとグループの数を調べて下さい。

- `/etc/passwd` ファイルの、最初の行、最後の行、および10行目のみをリストしてください。

3. 以下の `/etc/passwd` ファイルをこの問題の例として使います。コピーして `mypasswd` というファイルに保存して下さい。（訳注: 第5フィールド（GECOS）はメモで、このような内容であるとは限りません。）

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
nvidia-persistenced:x:121:128:NVIDIA Persistence
Daemon,,,:/nonexistent:/sbin/nologin
libvirt-qemu:x:64055:130:Libvirt Qemu,,,:/var/lib/libvirt:/usr/sbin/nologin
libvirt-dnsmasq:x:122:133:Libvirt
Dnsmasq,,,:/var/lib/libvirt/dnsmasq:/usr/sbin/nologin
carol:x:1000:2000:Carol Smith,Finance,,,Main Office:/home/carol:/bin/bash
dave:x:1001:1000:Dave Edwards,Finance,,,Main Office:/home/dave:/bin/ksh
emma:x:1002:1000:Emma Jones,Finance,,,Main Office:/home/emma:/bin/bash
frank:x:1003:1000:Frank Cassidy,Finance,,,Main Office:/home/frank:/bin/bash
grace:x:1004:1000:Grace Kearns,Engineering,,,Main Office:/home/grace:/bin/ksh
henry:x:1005:1000:Henry Adams,Sales,,,Main Office:/home/henry:/bin/bash
john:x:1006:1000:John Chapel,Sales,,,Main Office:/home/john:/bin/bash
```

- `mypasswd` ファイルから、グループ `1000` のユーザーをすべてリストして下さい。適当なフィールドを選択するために、`sed` を使って下さい。

- そのグループのユーザー全員の氏名をリストして下さい。`sed` と `cut` を使います。

発展演習

1. 前の演習の `mypasswd` ファイルをもう一度使用し、Main Officeから1人をくじ引きで選ぶコマンドラインを作成してください。`sed` コマンドを使用してMain Officeの行のみを出力し、そこから `cut` コマンドを使用してユーザーの名前を取り出します。次に、それらの名前をランダムに並べ替えて、リストから一番上の名前だけを出力します。

2. 財務部門（Finance）、技術部門（Engineering）、営業部門（Sales）で働く人は、それぞれ何人ですか？ `uniq` コマンドを使います。

3. 次に、カンマ区切りのCSVファイルを用意して、LibreOffice に簡単にインポートできるようにします。先ほどの例で作った `mypasswd` ファイルから `names.csv` を作成してください。ファイルの内容は、以下のような形式とします。

```
□,□,□□
Carol,Smith,Finance
...
John,Chapel,Sales
```

   ヒント: `sed`、`cut`、`paste` コマンドを使用します。このファイルでは、区切り文字がコンマ（`,`）になることに注意して下さい。

4. 前の演習で作成した `names.csv` というスプレッドシートは重要なファイルですから、改ざんされていないことを確認できるようにします。`md5sum` を使って、このファイルが変更されていないことを確認するにはどうしますか。

5. 古典文学を毎日100行読むことを自分に課し、Herman Melvilleの Mariner and Mystic から読み始めることにしました。この本を100行ずつのセクションに分割する `split` コマンドはどうなりますか？ なお、この本を入手するには、https://www.gutenberg.org で検索してください。

6. `/etc` ディレクトリで `ls -l` を実行して得られたリストを例にします。`ls` コマンドの出力結果から、`cut` コマンドを使用して、ファイル名のみを表示してください。次に、ファイル名とファイルの所有者を表示して下さい。`tr` コマンドで、連続する空白を1つの空白に まとめる ことができますから、その出力を `cut` コマンドで出力します。

7. この演習では、（仮想マシンではなく）実機を使用していることと、USBメモリーを所持していることを前提とします。`tail` コマンドのマニュアルページを参照して、ファイルにテキストが追加されたことを追跡する方法を見つけてください。`/var/log/syslog` ファイルへの出力を `tail` コマンドで監視し、USBメモリーを挿入した時に、製品名、メーカー、容量を表示するコマンドはどうなりますか？

まとめ

Linuxシステム管理において、テキストストリームの処理は非常に重要です。スクリプトを使用してテキストストリームを処理し、日常のタスクを自動化したり、ログファイルから情報を検索することができます。このレッスンで取り上げたコマンドの概要は次のとおりです。

`cat`
: プレーンテキストファイルを読み取る、ないしは、結合します。

`bzcat`
: `bzip2` 形式を使用して圧縮されたファイルを読み取ります。

`xzcat`
: `xz` 形式を使用して圧縮されたファイルを読み取ります。

`zcat`
: `gzip` 形式を使用して圧縮されたファイルを読み取ります。

`less`
: ファイルの内容をページングし、閲覧や検索を行います。

`head`
: デフォルトではファイルの先頭10行を表示します。`-n` オプションを使用すると、表示する行数を増減できます。

`tail`
: デフォルトではファイルの末尾10行を表示します。`-n` オプションを使用すると、表示する行数を増減できます。`-f` オプションはテキストファイルを追跡して、新しいデータが書き込まれた時にそれらをすぐに出力します。

`wc`
: “word count” の略で、文字、単語、行をカウントします。オプションでカウント対象を指定できます。

`sort`
: 行ごとに、アルファベット順、アルファベットの逆順、ランダムな順序に、並べ替えて出力します。

`uniq`
: 同一の行を1行にまとめて、まとめた行数と共に表示します。ファイルはソートされている必要があります。

od

“octal dump” に由来するコマンドで、ファイルの内容を8進数、10進数、16進数などで表示します。

nl

“number line” コマンドは、各行の先頭に行番号を付けてファイルを表示します。

sed

ストリームエディターは、正規表現を使用して一致する文字列を検索したり、ファイルを編集します。

tr

“translate” に由来するコマンドで、文字を置き換えたり、文字の繰り返しを削除ないしまとめます。

cut

区切り文字に基づいて行をフィールドに区切り、テキストファイルのカラム（列）を出力します。

paste

それぞれのファイルをカラムとして、区切り文字で区切りながら結合します。

split

オプションで指定した条件で、大きなファイルを小さなファイルに分割します。

md5sum

ファイルのMD5ハッシュ値を計算します。また、既存のハッシュ値とファイルの同一性を検証することもできます。

sha256sum

ファイルのSHA256ハッシュ値を計算します。また、既存のハッシュ値とファイルの同一性を検証することもできます。

sha512sum

ファイルのSHA512ハッシュ値を計算します。また、既存のハッシュ値とファイルの同一性を検証することもできます。

演習の解答

1. 学校にノートパソコンが寄付されたので、そのノートパソコンにLinuxをインストールしたいと思っています。OSが起動せずマニュアルがないため、LinuxをUSBブートしてノートパソコンのCPU情報を確認することにしました。シェルターミナルを起動して、/proc/cpuinfo ファイルの中身を確認しました:

```
processor   : 0
vendor_id   : GenuineIntel
cpu family  : 6
model       : 158

(□□)

processor   : 1
vendor_id   : GenuineIntel
cpu family  : 6
model       : 158

(□□)
```

   - コマンド grep と wc を使用して、ノートパソコンのプロセッサの数を表示してください。

     2つの方法があります。

```
$ cat /proc/cpuinfo | grep processor | wc -l
$ grep processor /proc/cpuinfo | wc -l
```

     同じことを行う方法がいくつもありますが、どちらを使うのがよいでしょう。いくつかの要因がありますが、最も重要なのはパフォーマンスと読みやすさの2つです。タスクを自動化するためのシェルスクリプトで、コマンドを使用することが多いでしょう。スクリプトが大きく複雑になるほど、パフォーマンスに気を配る必要があります。

   - `grep` の代わりに `sed` で同じことを行ってみましょう。

     `grep` の代わりに `sed` を使います:

```
$ sed -n /processor/p /proc/cpuinfo | wc -l
```

     `sed` に `-n` オプションを指定したので、`processor` に一致した行のみが `p` コマ

ンドで出力されます。そして `grep` コマンドの場合と同様に、`wc -l` で行数 ～ すなわちプロセッサの数をカウントします。

別の例も見てみましょう:

```
$ sed -n /processor/p /proc/cpuinfo | sed -n '$='
```

このコマンドは、`sed` の出力を `wc` コマンドにパイプした前の例と同じ結果を出力します。ここでの違いは、`wc -l` で行数をカウントするのではなく、`sed` を再度呼び出して同じことを行っています。ここでも `-n` オプションを使用して `sed` の出力を抑制して、今度は `'$='` を指示しています。この式は、`$` が最終行への一致を表し、`=` がその行番号を出力することを指示しています。

2. コマンド `grep`、`sed`、`head`、`tail` を使用して、`/etc/passwd` ファイルから，指定された内容を表示してください。
   - Bashシェルを利用するユーザをリストしてください。

     ```
     $ grep ":/bin/bash$" /etc/passwd
     ```

     Bashシェルを利用するユーザ名のみを表示するには、次のようにします。

     ```
     $ grep ":/bin/bash$" /etc/passwd | cut -d: -f1
     ```

     `grep` コマンドの出力を `cut` コマンドにパイプします。`/etc/passwd` ファイルは区切り文字が `:` なので、`cut` コマンドに `-d:` オプションを指定し、ユーザー名は1番目フィールドなので `-f1` オプションを指定します。

   - システムには、特定のプログラムを処理ないし管理するためのユーザが存在します。これらのユーザはシェルを使用しません。システムに存在する、そのようなユーザをカウントしてください。

     最も簡単な方法は、Bashシェルを使用しないアカウントの行を出力する方法です。

     ```
     $ grep -v ":/bin/bash$" /etc/passwd | wc -l
     ```

   - /etc/passwd ファイルのみを参照して、システムに存在するユーザーとグループの数を調べて下さい。

     `/etc/passwd` の各行のフィールドは、最初がユーザ名、2番目は通常 `x` でユーザのパスワードが `/etc/shadow` ファイルで暗号化されていることを示し、3番目はユ

ーザID（UID）、4番目はグループID（GID）です。ここからユーザ数がわかります。

```
$ cut -d: -f3 /etc/passwd | wc -l
```

ほとんどの場合はこれで大丈夫ですが、スーパーユーザーや特殊なユーザーが1つのUID（ユーザーID）を共有していることがあります。念のため、`cut` コマンドの結果を `sort` コマンドにパイプしてから、行数をカウントします。（訳注: `sort` の `-u` オプションは、`uniq` コマンドと同様に同一行をまとめます。）

```
$ cut -d: -f3 /etc/passwd | sort -u | wc -l
```

グループ数も同様です。

```
$ cut -d: -f4 /etc/passwd | sort -u | wc -l
```

- `/etc/passwd` ファイルの、最初の行、最後の行、および10行目のみをリストしてください。

  次のようになります。

  ```
  $ sed -n -e '1'p -e '10'p -e '$'p /etc/passwd
  ```

  `-n` は `p` コマンドで指定されたものだけを表示するオプションです。ここで使われているドル記号（`$`）は、ファイルの最終行への一致を意味します。

3. 以下の `/etc/passwd` ファイルをこの問題の例として使います。コピーして `mypasswd` というファイルに保存して下さい。（訳注: 実際には、第5フィールド（GECOS）はメモであり、必ずしもこのような内容であるとは限りません。）

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
nvidia-persistenced:x:121:128:NVIDIA Persistence
Daemon,,,:/nonexistent:/sbin/nologin
libvirt-qemu:x:64055:130:Libvirt Qemu,,,:/var/lib/libvirt:/usr/sbin/nologin
libvirt-dnsmasq:x:122:133:Libvirt
Dnsmasq,,,:/var/lib/libvirt/dnsmasq:/usr/sbin/nologin
```

```
carol:x:1000:2000:Carol Smith,Finance,,,Main Office:/home/carol:/bin/bash
dave:x:1001:1000:Dave Edwards,Finance,,,Main Office:/home/dave:/bin/ksh
emma:x:1002:1000:Emma Jones,Finance,,,Main Office:/home/emma:/bin/bash
frank:x:1003:1000:Frank Cassidy,Finance,,,Main Office:/home/frank:/bin/bash
grace:x:1004:1000:Grace Kearns,Engineering,,,Main Office:/home/grace:/bin/ksh
henry:x:1005:1000:Henry Adams,Sales,,,Main Office:/home/henry:/bin/bash
john:x:1006:1000:John Chapel,Sales,,,Main Office:/home/john:/bin/bash
```

- `mypasswd` ファイルから、グループ `1000` のユーザーをすべてリストして下さい。適当なフィールドを選択するために、`sed` を使って下さい。

  `/etc/passwd` ファイルの4番目のフィールドがGIDです。こうやりたくなるかもしれません:

  ```
  $ sed -n /1000/p mypasswd
  ```

  残念ながら、この場合、次の行も含まれてしまいます:

  ```
  carol:x:1000:2000:Carol Smith,Finance,,,Main Office:/home/carol:/bin/bash
  ```

  Carol SmithはGID 2000のメンバーであり、UIDが条件に一致したものですから誤りです。このような場合は、例えばGIDの次のフィールドが大文字で始まっていることに着目しましょう。これに気づけば、この問題を解決するために正規表現を利用できます。

  ```
  $ sed -n /:1000:[A-Z]/p mypasswd
  ```

  `[A-Z]` というパターンは、任意の大文字1文字にマッチします。正規表現については、レッスン103.7で詳しく学びます。

- そのグループのユーザー全員の氏名をリストして下さい。sed と cut を使います。

  2問目の最初の設問に対する解答と同様に、`cut` コマンドにパイプします。

  ```
  $ sed -n /:1000:[A-Z]/p mypasswd | cut -d: -f5
  Dave Edwards,Finance,,,Main Office
  Emma Jones,Finance,,,Main Office
  Frank Cassidy,Finance,,,Main Office
  Grace Kearns,Engineering,,,Main Office
  ```

```
Henry Adams,Sales,,,Main Office
John Chapel,Sales,,,Main Office
```

まだ余計な情報が残っています。結果の中のフィールドは `,` で区切られていることに着目しましょう。今度は区切り文字を `,` として、`cut` コマンドにパイプします。

```
$ sed -n /:1000:[A-Z]/p mypasswd | cut -d: -f5 | cut -d, -f1
Dave Edwards
Emma Jones
Frank Cassidy
Grace Kearns
Henry Adams
John Chapel
```

発展演習の解答

1. 前の演習の `mypasswd` ファイルをもう一度使用し、Main Officeから1人をくじ引きで選ぶコマンドラインを作成してください。`sed` コマンドを使用してMain Officeの行のみを出力し、そこから `cut` コマンドを使用してユーザーの名前を取り出します。次に、これらの名前をランダムに並べ替えて、リストから一番上の名前だけを出力します。

   `sort` コマンドの `-R` オプションの動作を調べておきましょう。次のコマンドを数回繰り返してみてください（`sed` が一つの文字列として扱うように、'Main Office' とシングルクォートで囲むことに注意してください。）:

   ```
   $ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d, -f1 | sort -R
   ```

   ランダムに並び換わることが確認できたでしょうか。最後に、リストの先頭を出力します:

   ```
   $ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d, -f1 | sort -R | head
   -1
   ```

2. 財務部門(Finance)、技術部門(Engineering)、営業部門(Sales)で働く人は、それぞれ何人ですか？ `uniq` コマンドを使います。

   今までの演習で学んだことを応用すれば、回答することができます。次のようにやってみてください:

   ```
   $ sed -n /'Main Office'/p mypasswd
   $ sed -n /'Main Office'/p mypasswd | cut -d, -f2
   ```

   ここでは、区切り文字が `:` であることを気にする必要はありません。今回は行を文字 `,` で分割して、その、2番目のフィールドを出力すればよいのです。

   ```
   $ sed -n /'Main Office'/p mypasswd | cut -d, -f2 | uniq -c
         4 Finance
         1 Engineering
         2 Sales
   ```

   `uniq` コマンドは、重複する行を省いて、1行だけを出力します。`-c` オプションは、重複行の出現回数を数えることを指示します。`uniq` は連続する重複行のみを処理しますから、重複行が離れている場合にはまず `sort` コマンドを使用する必要があります。

3. 次に、カンマ区切りのCSVファイルを用意して、LibreOffice に簡単にインポートできるようにします。先ほどの例で作った `mypasswd` ファイルから `names.csv` を作成してください。ファイルの内容は、以下のような形式とします。

```
□,□,□□
Carol,Smith,Finance
...
John,Chapel,Sales
```

ヒント: `sed`、`cut`、`paste` コマンドを使用します。このファイルでは、区切り文字がコンマ（,）になることに注意して下さい。

ここまでの演習で学んだことを踏まえて、まずは `sed` と `cut` コマンドから始ましょう:

```
$ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d" " -f1 > firstname
```

これで、従業員の名前を含む `firstname` ファイルができました。

```
$ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d" " -f2 | cut -d, -f1
> lastname
```

これで、各従業員の姓を含む `lastname` ファイルができました。

最後に、各従業員の所属部署を含むファイルを作成します:

```
$ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d, -f2 > department
```

先に進む前に、コマンドを試して出力を確認しましょう:

```
$ cat firstname lastname department
$ paste firstname lastname department
```

最後の仕上げです:

```
$ paste firstname lastname department | tr '\t' ,
$ paste firstname lastname department | tr '\t' , > names.csv
```

ここでは、`tr` コマンドで、区切り文字をタブを意味する `\t` から `,` に変換しています。`tr` は、ある文字を別の文字に置き換えるときに非常に便利です。`tr` と `paste` のmanページを読んでみてください。例えば、`paste` の `-d` オプションで区切り文字を指定すると、前のコマンドをよりシンプルにすることができます。

```
$ paste -d, firstname lastname department
```

ここでは、コマンドを紹介するために `paste` コマンドを使いましたが、すべてのタスクを1行で簡単に実行することもできます:

```
$ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d, -f1,2 | tr ' ' , >
names.csv
```

4. 前の演習で作成した `names.csv` というスプレッドシートは重要なファイルですから、改ざんされていないことを確認できるようにします。`md5sum` を使って、このファイルの同一性を確認するにはどうしますか。

   `md5sum`、`sha256sum`、`sha512sum` のマニュアルページを見ると、どれにも以下の文言が含まれていることが分かります:

   “compute and check XXX message digest”（メッセージダイジェストの計算と照合を行う）

   ここでの “XXX” は、メッセージダイジェスト を計算するために使用するアルゴリズムです。

   ここでは例として `md5sum` を使用しますが、他のコマンドも試してください。

   ```
   $ md5sum names.csv
   61f0251fcab61d9575b1d0cbf0195e25  names.csv
   ```

   例えば、sftpサービスなどにファイルをアップロードし、別の安全な方法で メッセージダイジェスト（ハッシュ値） を送信します。もしファイルが少しでも違っていたら、メッセージダイジェスト は全く異なるものになるでしょう。確認するために `names.csv` を編集して、JonesをJamesに変更してみます:

   ```
   $ sed -i.backup s/Jones/James/ names.csv
   $ md5sum names.csv
   f44a0d68cb480466099021bf6d6d2e65  names.csv
   ```

ファイルをダウンロードさせるときは、常に対応する メッセージダイジェスト を添えて配布するのがお勧めです。そうすれば、ダウンロードした人が メッセージダイジェスト を計算して、オリジナルと照合することができます。例えば、https://kernel.org には、ダウンロード可能なすべてのファイルのsha256sum値を取得できるページ https://mirrors.edge.kernel.org/pub/linux/kernel/v5.x/sha256sums.asc が用意されています。

5. 古典文学を毎日100行読むことを自分に課し、Herman Melvilleの Mariner and Mystic から読み始めることにしました。この本を100行ずつのセクションに分割する `split` コマンドはどうなりますか？ なお、この本を入手するには、https://www.gutenberg.org で検索してください。

   まず、Gutenbergプロジェクトのサイトから本を入手します。このサイトでは、この本だけでなく、パブリックドメインで入手可能な他の本も入手できます。

   ```
   $ wget https://www.gutenberg.org/files/50461/50461-0.txt
   ```

   `wget` がシステムにインストールされていない場合は、インストールします。あるいは、`curl` コマンドを使用することもできます。ファイルを入手したら、`less` を使用して確認します。

   ```
   $ less 50461-0.txt
   ```

   次に、この本を100行ごとの塊に分割します。

   ```
   $ split -l 100 -d 50461-0.txt melville
   ```

   `50461-0.txt` はこれから分割するファイルの名前です。`melville` は分割後のファイル名のプレフィックスになります。`-l 100` は分割する行数を示し、`-d` オプションはファイル名のサフィックスを（英字ではなく）数値にすることでファイル名に番号を振るように指定します。分割されたファイルのいずれか（最後のファイル以外）で `nl` を使用して、それぞれが100行であることを確認してみましょう。

6. `/etc` ディレクトリで `ls -l` を実行して得られたリストを例にします。`ls` コマンドの出力結果から、`cut` コマンドを使用して、ファイル名のみを表示してください。次に、ファイル名とファイルの所有者を表示して下さい。`tr` コマンドで、連続する空白を1つの空白に まとめる ことができますから、その出力を `cut` コマンドで出力します。

   オプションを指定しない `ls` コマンドは、ファイルの名前だけを表示します。`ls -l`（長いリスト）では、より詳細な情報を表示できます。

```
$ ls -l /etc | tr -s ' ' ,
drwxr-xr-x,3,root,root,4096,out,24,16:58,acpi
-rw-r--r--,1,root,root,3028,dez,17,2018,adduser.conf
-rw-r--r--,1,root,root,10,out,2,17:38,adjtime
drwxr-xr-x,2,root,root,12288,out,31,09:40,alternatives
-rw-r--r--,1,root,root,401,mai,29,2017,anacrontab
-rw-r--r--,1,root,root,433,out,1,2017,apg.conf
drwxr-xr-x,6,root,root,4096,dez,17,2018,apm
drwxr-xr-x,3,root,root,4096,out,24,16:58,apparmor
drwxr-xr-x,9,root,root,4096,nov,6,20:20,apparmor.d
```

`tr` の `-s` オプションは、指定した文字が連続している場合に、それら1つにまとめることを指示します。ここでは空白を指定しているので、連続する空白を1つにまとめます。`tr` コマンドでは、指定したすべての文字の繰り返しが対象になります。この例ではさらに、空白をコンマ `,` に置き換えていますが、題意からは空白のままで構わないので、`,` を省略します。

```
$ ls -l /etc | tr -s ' '
drwxr-xr-x 3 root root 4096 out 24 16:58 acpi
-rw-r--r-- 1 root root 3028 dez 17 2018 adduser.conf
-rw-r--r-- 1 root root 10 out 2 17:38 adjtime
drwxr-xr-x 2 root root 12288 out 31 09:40 alternatives
-rw-r--r-- 1 root root 401 mai 29 2017 anacrontab
-rw-r--r-- 1 root root 433 out 1 2017 apg.conf
drwxr-xr-x 6 root root 4096 dez 17 2018 apm
drwxr-xr-x 3 root root 4096 out 24 16:58 apparmor
```

ファイル名だけを表示するには、9番目のフィールドのみを表示します。

```
$ ls -l /etc | tr -s ' ' | cut -d" " -f9
```

ファイル名とファイルの所有者を表示する場合は、9番目と3番目のフィールドを使います。

```
$ ls -l /etc | tr -s ' ' | cut -d" " -f9,3
```

ちなみに、フォルダ名とその所有者だけが必要な場合は、以下のようにします。

```
$ ls -l /etc | grep ^d | tr -s ' ' | cut -d" " -f9,3
```

7. この演習では、（仮想マシンではなく）実機を使用していることと、USBメモリーを所持していることを前提とします。`tail` コマンドのマニュアルページを参照して、ファイルにテキストが追加されたことを追跡する方法を見つけてください。`/var/log/syslog` ファイルへの出力を `tail` コマンドで監視し、USBメモリーを挿入した時に、製品名、メーカー、容量を表示するコマンドはどうなりますか？

```
$ tail -f /var/log/syslog | grep -E -i 'product\:\|blocks\|manufacturer'
Nov  8 06:01:35 brod-avell kernel: [124954.369361] usb 1-4.3: Product: Cruzer
Blade
Nov  8 06:01:35 brod-avell kernel: [124954.369364] usb 1-4.3: Manufacturer:
SanDisk
Nov  8 06:01:37 brod-avell kernel: [124955.419267] sd 2:0:0:0: [sdc] 61056064
512-byte logical blocks: (31.3 GB/29.1 GiB)
```

これは一例です。表示される結果はUSBメモリーの製造元によって異なります。検索対象となる文字列が大文字か小文字かわからないので、`grep` コマンドに `-i` オプションを使用しています。また、`|` はOR検索を指示していて、つまり `product`、`blocks`、`manufacturer` のいずれかを含む行を検索します。また、`|` は拡張正規表現のパターンなので、`-E` オプションを指定します。

103.3 基本的なファイル管理を実行する

LPI目標への参照

LPIC-1 v5, Exam 101, Objective 103.3

総重量

4

主な知識分野

- ファイルとディレクトリを個別にコピー、移動、削除する。
- 再帰的に複数のファイルとディレクトリをコピーする。
- 再帰的にファイルとディレクトリを削除する。
- シンプルで高度なワイルドカード仕様をコマンドで使用する。
- findを使用して、種類、サイズ、または時間に基づいてファイルを検索して処理する。
- tar、cpio、ddの使い方。

用語とユーティリティ

- `cp`
- `find`
- `mkdir`
- `mv`
- `ls`
- `rm`
- `rmdir`
- `touch`

- `tar`
- `cpio`
- `dd`
- `file`
- `gzip`
- `gunzip`
- `bzip2`
- `bunzip2`
- ファイルグロビング

103.3 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	103 GNUおよびUnixコマンド
Objective:	103.3 基本的なファイル管理
Lesson:	1 of 2

はじめに

Linuxでは、すべてをファイルに抽象化しているので、ファイルの操作方法を理解することがとても重要です。このレッスンでは、ファイルの基本的な操作について説明します。

Linuxユーザは、ファイルシステム内を移動し、ファイルをある場所から別の場所にコピーし、ファイルを削除する、といった操作を日常的に行います。ここでは、ファイル管理に関連するコマンドについても説明します。

ファイルは、データやプログラムを格納する実体で、コンテンツ（内容本体）とメタデータ（ファイルサイズ、所有者、作成日、パーミッションなど）で構成されています。別のファイルを格納するための特別なファイルであるディレクトリを使って、ファイルを整理することができます。

さまざまな種類のファイルがあります:

通常ファイル
: データやプログラムを保存します。

ディレクトリ
: 通常ファイルやディレクトリを格納します。

デバイスファイル
入出力に使用します。

もちろん、これ以外にもファイルの種類がありますが、本レッスンの範囲外です。これらのさまざまなファイルタイプを見分ける方法を後で説明します。

ファイルの操作

`ls` でファイルを一覧表示する

`ls` コマンドは、ファイルシステムを探索する、もっとも重要なコマンドラインツールの1つです。

（オプション無しの）基本的な形式では、`ls` はファイル名とディレクトリ名 のみ を表示します。

```
$ ls
Desktop   Downloads   emp_salary  file1   Music   Public   Videos
Documents emp_name    examples.desktop    file2   Pictures Templates
```

`-l` オプションを指定すると、“ロングリスト” 形式で表示されます。具体的には、ファイルまたはディレクトリのパーミッション、リンク数、所有者、所有グループ、サイズ、更新日時、名前が表示されます。

```
$ ls -l
total 60
drwxr-xr-x  2   frank   frank   4096    Apr 1   2018    Desktop
drwxr-xr-x  2   frank   frank   4096    Apr 1   2018    Documents
drwxr-xr-x  2   frank   frank   4096    Apr 1   2018    Downloads
-rw-r--r--  1   frank   frank     21    Sep 7   12:59   emp_name
-rw-r--r--  1   frank   frank     20    Sep 7   13:03   emp_salary
-rw-r--r--  1   frank   frank   8980    Apr 1   2018    examples.desktop
-rw-r--r--  1   frank   frank     10    Sep 1   2018    file1
-rw-r--r--  1   frank   frank     10    Sep 1   2018    file2
drwxr-xr-x  2   frank   frank   4096    Apr 1   2018    Music
drwxr-xr-x  2   frank   frank   4096    Apr 1   2018    Pictures
drwxr-xr-x  2   frank   frank   4096    Apr 1   2018    Public
drwxr-xr-x  2   frank   frank   4096    Apr 1   2018    Templates
drwxr-xr-x  2   frank   frank   4096    Apr 1   2018    Videos
```

出力される各行の最初の文字は、ファイル種別を示します。

-
: 通常ファイル

d
: ディレクトリ

b **c** **p** **s** **など**
: 特殊ファイル

ファイルサイズはbyte単位で表示されますが、数が大きくなると分かりづらくなります。これを人間が読みやすい形式で表示するには、`-h` オプションを追加します。

```
$ ls -lh
total 60K
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Desktop
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Documents
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Downloads
-rw-r--r--  1   frank   frank     21    Sep 7   12:59   emp_name
-rw-r--r--  1   frank   frank     20    Sep 7   13:03   emp_salary
-rw-r--r--  1   frank   frank   8.8K    Apr 1   2018    examples.desktop
-rw-r--r--  1   frank   frank     10    Sep 1   2018    file1
-rw-r--r--  1   frank   frank     10    Sep 1   2018    file2
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Music
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Pictures
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Public
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Templates
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Videos
```

隠しファイル（`.` で始まるファイル）を含むすべてのファイルをリストするには、`-a` オプションを使用します。

```
$ ls -a
.               .dbus   file1   .profile
..              Desktop file2   Public
.bash_history   .dmrc   .gconf  .sudo_as_admin_successful
```

通常は表示されない `.bash_history` などの設定ファイルが表示されます。

`ls` コマンドの一般的な構文を以下に示します:

```
ls OPTIONS FILE
```

ここでの OPTIONS は前述のいずれかのオプションであり、FILE はリストしたいファイルまたはディレクトリの名前を指定します。（すべてのオプションを表示するには man ls を実行します。）

NOTE FILE を指定しない場合、現在のディレクトリを指定したとみなされます。

ファイルの作成、コピー、移動、削除

touch でファイルを作成する

touch コマンドは、最も簡単に新しい空のテキストファイルを作成します。また、このコマンドで、既存のファイルやディレクトリのタイムスタンプ（作成時刻や更新時刻）を変更することもできます。touch の構文は次のとおりです。

```
touch OPTIONS FILE_NAME(S)
```

オプションを指定しない場合、touch は引数に指定した名前の新しいテキストファイルを作成します。touch は、任意の数のファイルを一気に作成することもできます。

```
$ touch file1 file2 file3
```

上の例では、file1、file2、file3 という名前の、3つの新しい空のファイルが作成されます。

touch オプションのいくつかは、ファイルのタイムスタンプを変更するためののものです。たとえば、-a オプションは最終アクセス日時のみを変更し、-m オプションは更新日時のみを変更します。両方のオプションを一緒に使用すると、アクセス日時と更新日時の両方が、現在の日時に変更されます。

```
$ touch -am file3
```

cp でファイルをコピーする

Linuxでは、ファイルをある場所から別の場所にコピーすることがよくあります。あるディレクトリから別のディレクトリにファイルを複製する場合には、音楽ファイルであれシステムファイルであれ、いつでも cp を使用します。

```
$ cp file1 dir2
```

このコマンドは、file1 を dir2 ディレクトリにコピーすることを指示しています。その結果、dir2 内に file1 が複製されます。このコマンドを正常に実行するには、file1 がユーザの作業ディレクトリ（カレントディレクトリ）に存在する必要があります。対象のファイルが無い時は、No such file or directory（そのようなファイルまたはディレクトリはありません）というメッセージでエラーが報告されます。

```
$ cp dir1/file1 dir2
```

ここでは file1 へのパスが示されていることに注意してください。ファイルのパスを示すには、相対パス ないし 絶対パス を使います。相対パスは作業ディレクトリを起点として参照されますが、絶対パスでは起点ディレクトリを意識する必要はありません。（訳注: 作業ディレクトリは cd コマンドで、表示・変更する事ができます。詳しくは help cd でヘルプを参照してください）。少し掘り下げましょう。

上のコマンドは、dir1 ディレクトリ内の file1 を、dir2 ディレクトリにコピーしています。ユーザの作業ディレクトリが dir1 ではないので、file1 へのパスを明示する必要があります。

```
$ cp /home/frank/Documents/file2 /home/frank/Documents/Backup
```

この3番目の例では、/home/frank/Documents ディレクトリにある file2 を、/home/frank/Documents/Backup ディレクトリにコピーします。ここではファイルを 絶対パス で指定しています。ここまでの2つの例では 相対パス で指定していました。パスを指定する際に、最初の文字が / で始まる場合は絶対パス、それ以外の場合は相対パスです。

cp の一般的な構文は次のとおりです。

```
cp OPTIONS SOURCE DESTINATION
```

SOURCE はコピーする元のファイルであり、DESTINATION はファイルがコピーされる先のディレクトリです。SOURCE と DESTINATION のいずれも、絶対パスまたは相対パスで指定できます。DESTINATION のディレクトリが存在しない場合（ないしは DESTINATION がファイルの場合）は、DESTINATION に指定した名前のファイルが（必要に応じて）作成され、その内容は SOURCES と同じになります。

mv でファイルを移動する

ファイルを移動ないし名前変更するためのコマンドが用意されています。コピー用の cp と似た、mv と言うコマンドです。

移動の操作は、グラフィカルユーザーインターフェイス（GUI）における、ファイルないし

ディレクトリに対するカット＆ペースト操作に相当します。（訳注: コピーはGUIにおけるコピー＆ペーストに相当します）。

ファイルを別の場所に移動する場合は、次のように `mv` を使用します。

```
mv FILENAME DESTINATION_DIRECTORY
```

例を示しましょう。

```
$ mv myfile.txt /home/frank/Documents
```

このコマンドを実行すると、`myfile.txt` が、移動先（DESTINATION）に指定した `/home/frank/Documents` に移動します。

ファイル名を変更するには、`mv` を次のように使用します。

```
$ mv old_file_name new_file_name
```

このコマンドで、ファイル名前が `old_file_name` から `new_file_name` に変更されます。

`mv` はデフォルトでは、移動先に同名のファイルが存在する場合に、確認することなくそれを上書きします（問い合わせはありません）。オプション `-i` を使用すると、システムは上書きの可否を問い合わせます。

```
$ mv -i old_file_name new_file_name
mv: overwrite 'new_file_name'?
```

この例では、`old_file_name` を `new_file_name` に上書きする前に、ユーザの許可を求めています。

逆に、`-f` オプションを使用すると:

```
$ mv -f old_file_name new_file_name
```

許可を求めることなくファイルを強制的に上書きします。

`mv` コマンドでは、ディレクトリを別の場所に移動することもできます。移動元（第1引数）がファイルではなくディレクトリであった場合は、移動元のディレクトリが `DESTINATION_DIRECTORY` のサブディレクトリとなるように、移動元のディレクトリツリー全

体が移動します。また、ディレクトリの名前を変更する事もできます。

`rm` でファイルを削除する

ファイルを削除するためには `rm` を使用します。“remove” という単語の省略形と考えてください。このコマンドでファイルを削除すると、そのファイルは復元できなくなるため、このコマンドの使用には注意が必要です。

```
$ rm file1
```

これは `file1` を削除するコマンドです。

```
$ rm -i file1
rm: remove regular file 'file1'?
```

このコマンドでは、`file1` を削除する前にユーザに確認を求めます。前の `mv` の学習の際に、`-i` オプションによってメッセージが表示されたことを思い出してください。

```
$ rm -f file1
```

このコマンドは、確認を求めることなく、`file1` を強制的に削除します。

また、複数のファイルを同時に削除することもできます。

```
$ rm file1 file2 file3
```

この例では、`file1`、`file2`、および `file3` が一気に削除されます。

`rm` の構文は、次の通りです。

```
rm OPTIONS FILE
```

ディレクトリの作成と削除

`mkdir` でディレクトリを作成する

ファイルとフォルダを整理するためには、ディレクトリを作成する必要があります。ファイルをディレクトリ内に格納することで、論理的にグループ化することができます。ディレク

トリを作成するには、`mkdir` を使用します。

```
mkdir OPTIONS DIRECTORY_NAME
```

ここで使用する、DIRECTORY_NAME　は、作成するディレクトリの名前です。任意の数のディレクトリを一気に作成することもできます。

```
$ mkdir dir1
```

この例では、現在のディレクトリに、`dir1` ディレクトリを作成します。

```
$ mkdir dir1 dir2 dir3
```

上記のコマンド例では、3つのディレクトリ `dir1`、`dir2`、`dir3` を一気に作成します。

サブディレクトリと共に、そのディレクトリを作成したい場合には、`-p`（“parents”）オプションを使用します。

```
$ mkdir -p parents/children
```

このコマンドは、ディレクトリ構造　`parents/children`　を作成します。つまり、ディレクトリ `parents` を作成し、その中にさらにディレクトリ `children` を作成します。

`rmdir` でディレクトリを削除する

`rmdir`　は、ディレクトリが　空の場合に、そのディレクトリを削除します。構文は次のとおりです。

```
rmdir OPTIONS DIRECTORY
```

ここで DIRECTORY には、ひとつないし複数の引数を指定することができます。

```
$ rmdir dir1
```

このコマンドでは `dir1` を削除します。

```
$ rmdir dir1 dir2
```

このコマンドでは、`dir1` と `dir2` を一気に削除します。

サブディレクトリを持つディレクトリを削除することもできます。

```
$ rmdir -p parents/children
```

このコマンドは、ディレクトリ構造 `parents/children` を削除します。ただし、いずれかのディレクトリが空でない場合には、削除されません。

ファイルとディレクトリの再帰的な操作

ディレクトリとその内容をまとめて操作したい時には、再帰的な 処理を行う必要があります。再帰とは、ディレクトリツリー全体を下にたどりながら、同じ操作を繰り返し行うことを意味します。Linuxでは、オプション `-r` ないし `-R`、`--recursive` を使うのが一般的です。

次のシナリオは、再帰的操作をよりよく理解するのに役立ちます。

あるディレクトリ `students` の内容をリストします。このディレクトリには、2つのサブディレクトリ `level 1` と `level 2`、ならびに1つのファイル `frank` が含まれています。`ls` コマンドに再帰的な処理を指定すると、まず `students` の内容～`level 1`、`level 2`、`frank` をリストしますが、（サブディレクトリがあるので）処理が続きます。ディレクトリ `students` の内容をリストした時と同様に、サブディレクトリである `level 1` と `level 2` に入り、それぞれの内容をリストし、サブディレクトリを見つける度に、ディレクトリツリーを下へ下へと進んでいきます。

`ls -R` による再帰リスト

ディレクトリの内容を、そのサブディレクトリやファイルとともにリストするには、`ls -R` を使用します。

```
$ ls -R mydirectory
mydirectory/:
file1   newdirectory

mydirectory/newdirectory:
```

このコマンドでは、ディレクトリ `mydirectory` に含まれるすべてのコンテンツがリストされて、サブディレクトリ `newdirectory` とファイル `file1` を含んでいることがわかりま

す。なお `newdirectory` は空なので、その内容は表示されません。

サブディレクトリを含むディレクトリの内容すべてをリストするには、次の構文を使用します。

```
ls -R DIRECTORY_NAME
```

DIRECTORY_NAME に末尾のスラッシュを追加しても、しなくても結果は変わりません。

```
$ ls -R animal
```

上の例も、次の例も、同じ結果が表示されます。

```
$ ls -R animal/
```

`cp -r` による再帰的コピー

`cp -r`（または `-R` ないし `--recursive` ）を使用すると、対象のディレクトリと、そのディレクトリ内のすべてのサブディレクトリおよびファイルを、まとめてコピーできます。

```
$ tree mydir
mydir
|_file1
|_newdir
    |_file2
    |_insidenew
        |_lastdir


3 directories, 2 files
$ mkdir newcopy
$ cp mydir newcopy
cp: omitting directory 'mydir'
$ cp -r mydir newcopy
$ tree newcopy
newcopy
|_mydir
    |_file1
    |_newdir
        |_file2
        |_insidenew
```

```
            |_lastdir

4 directories, 2 files
```

上記のコマンド実行例では、`-r` を指定せずに `cp` を使用してディレクトリ `mydir` を `newcopy` にコピーしようとすると、メッセージ `cp: omitting directory 'mydir'` が表示されて、コピーできないことがわかります。オプション `-r` を追加すると、指定したディレクトリである `mydir` を含む、すべての内容が `newcopy` にコピーされます。つまり、ディレクトリをコピーするには、`-r` オプションが必須です。

ディレクトリとサブディレクトリをコピーする際の構文は、次のとおりです。

```
cp -r SOURCE DESTINATION
```

`rm -r` による再帰的削除

`rm -r` は、ディレクトリとそのすべての内容（サブディレクトリとファイル）を削除します。

WARNING

`rm` コマンドに `-r` または `-rf` オプションを指定する場合は、十分な注意が必要です。重要なシステムディレクトリに対して再帰的な削除コマンドを実行すると、システムが使用できなくなるおそれがあります。ディレクトリの内容をコンピュータから削除しても安全であると確信できない限り、再帰的な削除コマンドを使用してはいけません。

`-r` を指定せずにディレクトリを削除しようとすると、システムはエラーを表示します。

```
$ rm newcopy/
rm: cannot remove 'newcopy/': Is a directory
$ rm -r newcopy/
```

ディレクトリを削除するには、2番目のコマンドのように `-r` を追加する必要があります。

NOTE

なぜ `rmdir` を使用しないのかを、疑問に思うかもしれません。2つのコマンドには微妙な違いがあります。`rmdir` は、指定のディレクトリが空の場合にのみ削除しますが、`rm -r` は、ディレクトリが空であるかどうかに関係なく（すべてを再帰的に）削除します。

`-i` オプションを指定すると、ファイルを削除する前に確認メッセージを表示します。

```
$ rm -ri mydir/
```

```
rm: remove directory 'mydir/'?
```

このように、システムは `mydir` を削除する前に、確認メッセージを表示します。

ファイルのグロブとワイルドカード

ファイルの　グロブ　（globbing）は、Unix/Linuxシェルが提供する機能であり、ワイルドカード　と呼ばれる特殊文字を使用して複数のファイル名を表現します。ワイルドカードは基本的に、1つ以上の文字の代わりに使用する記号です。たとえば、文字　`A`　で始まるすべてのファイルや、文字列 `.conf` で終わるすべてのファイルを表現できます。

`cp`、`ls`、`rm`　などのコマンドを使用するときにワイルドカードを使用すると、とても便利です。

グロブの例をいくつか示します。

`rm *`
: 現在の作業ディレクトリ内のすべてのファイルを削除します。

`ls l?st`
: 名前が　`l`　で始まり、2文字目に任意の1文字が続き、`st`　で終わる、すべてのファイルをリストします。

`rmdir [a-z]*`
: 名前が英小文字で始まるすべてのディレクトリを削除します。

ワイルドカードの種類

ワイルドカードとして使用する文字が、3つあります。

`*` (アスタリスク)
: 0文字以上の任意の文字列を示します。

`?` (疑問符)
: 任意の1文字を示します。

`[ ]` (括弧で囲まれた文字)
: 角括弧で囲まれた文字のいずれか1文字を示します。数字、文字、その他の特殊文字など、さまざまな種類の文字を使用できます。たとえば、`[0-9]`　はすべての数字のうちの1文字を示します。

アスタリスク

アスタリスク（`*`）は、0文字以上の任意の文字列に一致します。

例えば:

```
$ find /home -name *.png
```

このコマンドは、`photo.png`、`cat.png`、`frank.png` などの `.png` で終わるすべてのファイルを検索します。`find` コマンドについては、次のレッスンで詳しく説明します。

同様に:

```
$ ls lpic-*.txt
```

このコマンドは、`lpic-1.txt` や `lpic-2.txt` のように、文字列 `lpic-` で始まり、その後に任意の数の文字が続き、`.txt` で終わる、すべてのファイルがリストされます。

ワイルドカード文字であるアスタリスクを使用して、ディレクトリのすべての内容を操作（コピー、削除、または移動）することもできます。

```
$ cp -r animal/* forest
```

この例では、`animal` ディレクトリのすべての内容（ファイルとサブディレクトリ）が、`forest` ディレクトリにコピーされます。

ディレクトリのすべての内容をコピーするには、次のようにするのが一般的です。

```
cp -r SOURCE_PATH/* DEST_PATH
```

コピーしたいファイルがあるディレクトリにいる場合は、`SOURCE_PATH` を省略できます。ただし、`*` は隠しファイルには一致しないので、ディレクトリに隠しファイルがある場合、それらはコピーされません。

他のワイルドカードも同様ですが、1つのコマンドの任意の位置で、アスタリスクを繰り返し使用できます。

```
$ rm *ate*
```

ファイル名の途中に、文字列 `ata` を含むファイル（`ata` が先頭、あるいは末尾であっても構いません）が削除されます。

疑問符

疑問符（`?`）は、任意の 1文字 と一致します。

下記のリストを見てみましょう:

```
$ ls
last.txt    lest.txt    list.txt    third.txt   past.txt
```

`l` で始まり、その後に任意の1文字を挟み、その後に `st.txt` が続く名前のファイルをリストするには、疑問符（`?`）ワイルドカードを使用します。

```
$ ls l?st.txt
last.txt    lest.txt    list.txt
```

指定した条件に一致するファイル `last.txt`、`lest.txt`、`list.txt` のみが表示されました。

同様に:

```
$ ls ??st.txt
last.txt    lest.txt    list.txt    past.txt
```

先頭2文字が任意で、その後に文字列 `st.txt` が続くファイルをリストします。

括弧で囲まれた文字

角括弧で囲んだワイルドカードは、角括弧で囲まれた文字のいずれかに一致した1文字を示します。

```
$ ls l[aef]st.txt
last.txt    lest.txt
```

このコマンドは、`l` で始まり、2文字目が `aef` 内の いずれかの 文字、さらに `st.txt` で終わるファイルをリストしました。

角括弧には、範囲を指定することができます。

```
$ ls l[a-z]st.txt
last.txt    lest.txt    list.txt
```

今度は、`l` で始まり、2文字目が `a` から `z` の範囲（英小文字）1文字、さらに `st.txt` で終わるファイルをリストしました。

角括弧内に、複数の範囲を指定することもできます。

```
$ ls
student-1A.txt  student-2A.txt  student-3.txt
$ ls student-[0-9][A-Z].txt
student-1A.text student-2A.txt
```

登録済み学生のリストが表示されました。登録番号が次の基準を満たす学生のみをリストしたことになります:

- `student-` で始まり、
- 数字1文字と大文字1文字が続き、
- `.txt` で終わる

ワイルドカードの組み合わせ

次のように、ワイルドカードを組み合わせることができます。

```
$ ls
last.txt    lest.txt    list.txt    third.txt   past.txt
$ ls [plf]?st*
last.txt    lest.txt    list.txt    past.txt
```

ワイルドカードの最初の要素（`[plf]`）には、文字 `p`、`l`、`f` のいずれか1文字が一致します。2番目の要素（`?`）には、任意の1文字が一致します。3番目の要素（`*`）は、任意の文字列を表します。

```
$ ls
file1.txt file.txt file23.txt fom23.txt
$ ls f*[0-9].txt
file1.txt file23.txt fom23.txt
```

このコマンドは、文字 `f` で始まり、その後に任意の文字列が続き、少なくとも1桁の数字

があり、`.txt` で終わるファイルを表示します。`file.txt` はこの条件に一致しないため、表示されなかったということです。（訳注: `.txt` の前に数字が続く場合、最後の1文字だけが `[0-9]` に一致し、それに先立つ数字は `*` に含まれます。そのため「少なくとも1桁の数字」になります。）

演習

1. 以下のリストについて回答して下さい。

```
$ ls -lh
total 60K
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Desktop
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Documents
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Downloads
-rw-r--r--  1   frank   frank     21    Sep 7   12:59   emp_name
-rw-r--r--  1   frank   frank     20    Sep 7   13:03   emp_salary
-rw-r--r--  1   frank   frank   8.8K    Apr 1   2018    examples.desktop
-rw-r--r--  1   frank   frank     10    Sep 1   2018    file1
-rw-r--r--  1   frank   frank     10    Sep 1   2018    file2
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Music
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Pictures
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Public
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Templates
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Videos
```

   - 行の先頭に、文字 `d` があるものは何を表しますか？

   - ファイルサイズの数値が、読みやすい形式（単位付き）で表示されているのはなぜですか？

   - `ls` にオプションを指定しない場合、出力はどうなりますか？

2. 以下のコマンドについて回答してください。

```
$ cp /home/frank/emp_name /home/frank/backup
```

   - コマンドが正常に実行された場合、ファイル `emp_name` はどうなりますか？

   - `emp_name` がディレクトリの場合、`cp` コマンドに追加すべきオプションは何ですか？

- `cp` を `mv` に変更した場合、どのような結果になりますか？

3. 以下のリストについて回答して下さい。

```
$ ls
file1.txt file2.txt file3.txt file4.txt
```

このディレクトリすべての内容を削除するワイルドカードは何ですか？

4. 前問のファイルリストがある場合、次のコマンドで表示されるファイルはどれですか？

```
$ ls file*.txt
```

5. 同様に、すべてのファイルをリストするコマンドを完成させるために、角括弧内に埋め込む適切な数字と文字は何ですか？

```
$ ls file[].txt
```

発展演習

1. ホームディレクトリに、`dog` と `cat` というファイルを作成してください。
2. ホームディレクトリに、`animal` というディレクトリを作成し、`dog` と `cat` を `animal` に移動してください。
3. ホームディレクトリにある `Documents` フォルダに移動し、その中にディレクトリ `backup` を作成してください。
4. `animal` ディレクトリと、その内容すべてを `backup` にコピーしてください。
5. `backup` の `animal` の名前を `animal.bkup` に変更してください。
6. `/home/lpi/database` ディレクトリには、`db-1.tar.gz`、`db-2.tar.gz`、`db-3.tar.gz` などの多くのファイルが含まれています。ここに挙げたファイルのみをリストする1行のコマンドはどうなりますか？

7. 以下のリストについて回答して下さい。

```
$ ls
cne1222223.pdf cne12349.txt cne1234.pdf
```

グロブ文字を1つ使用して、PDFファイルのみを削除するコマンドはどうなりますか？

まとめ

このレッスンでは、`ls`　コマンドを使用してディレクトリ内にあるものを表示する方法、ファイルとフォルダをコピー（`cp`）する方法、それらを移動（`mv`）する方法について説明しました。また、`mkdir`　コマンドを使用して新しいディレクトリを作成する方法や、ファイル（`rm`）とフォルダ（`rmdir`）を削除するコマンドについても説明しました。

このレッスンでは、ファイルのグロブとワイルドカードについても学びました。ファイルのグロブは、ワイルドカードと呼ばれる特殊文字を使用して、複数のファイル名を表すために使用します。基本的なワイルドカードとその意味を次に示します:

`?` (疑問符)
　任意の1文字を示します。

`[ ]` (角括弧)
　角括弧で囲まれた文字のいずれか1文字を示します。

`*` (アスタリスク)
　0文字以上の任意の文字列を示します。

これらのワイルドカードは、1つの引数の中で組み合わせることができます。

演習の解答

1. 以下のリストについて回答して下さい。

```
$ ls -lh
total 60K
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Desktop
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Documents
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Downloads
-rw-r--r--  1   frank   frank     21    Sep 7   12:59   emp_name
-rw-r--r--  1   frank   frank     20    Sep 7   13:03   emp_salary
-rw-r--r--  1   frank   frank   8.8K    Apr 1   2018    examples.desktop
-rw-r--r--  1   frank   frank     10    Sep 1   2018    file1
-rw-r--r--  1   frank   frank     10    Sep 1   2018    file2
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Music
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Pictures
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Public
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Templates
drwxr-xr-x  2   frank   frank   4.0K    Apr 1   2018    Videos
```

   - 行の先頭に文字 `d` があるものは何を表しますか？

     `d` はディレクトリを示す文字です。

   - ファイルサイズの数値が、読みやすい形式（単位付き）で表示されているのはなぜですか？

     オプション `-h` を使用したからです。

   - `ls` にオプションを指定しない場合、出力はどうなりますか？

     ディレクトリ名とファイル名のみが表示されます。

2. 以下のコマンドについて回答してください。

```
$ cp /home/frank/emp_name /home/frank/backup
```

   - コマンドが正常に実行された場合、ファイル `emp_name` はどうなりますか？

     `emp_name` は `backup` にコピーされます。

   - `emp_name` がディレクトリの場合、`cp` コマンドに追加すべきオプションは何ですか？

`-r`

- `cp` を `mv` に変更した場合、どのような結果になりますか？

  `emp_name` は `backup` に移動されて、ユーザ `frank` のホームディレクトリ内には存在しなくなります。

3. 以下のリストについて回答して下さい。

```
$ ls
file1.txt file2.txt file3.txt file4.txt
```

このディレクトリすべての内容を削除するワイルドカードは何ですか？

アスタリスク `*`

4. 前問のファイルリストがある場合、次のコマンドで表示されるファイルはどれですか？

```
$ ls file*.txt
```

アスタリスク文字は任意の数の文字を表すので、すべてのファイルが表示されます。

5. 同様に、すべてのファイルをリストするコマンドを完成させるために、角括弧内に埋め込む適切な数字と文字は何ですか？

```
$ ls file[].txt
```

`file[0-9].txt`。`file[1-4].txt` でもかまいません。

発展演習の解答

1. ホームディレクトリに、`dog` と `cat` というファイルを作成してください。

```
$ touch dog cat
```

2. ホームディレクトリに、`animal` というディレクトリを作成し、`dog` と `cat` を `animal` に移動してください。

```
$ mkdir animal
$ mv dog cat -t animal/
```

`-t` はなくても構いません。

3. ホームディレクトリにある `Documents` フォルダに移動し、その中にディレクトリ `backup` を作成してください。

```
$ cd ~/Documents
$ mkdir backup
```

4. `animal` ディレクトリと、その内容すべてを `backup` にコピーしてください。

```
$ cp -r ../animal ~/Documents/backup
```

5. `backup` の `animal` の名前を `animal.bkup` に変更してください。

```
$ mv backup/animal/ backup/animal.bkup
 □□□
$ cd backup
$ mv animail/ animal.bkup
```

6. `/home/lpi/database` ディレクトリには、`db-1.tar.gz`、`db-2.tar.gz`、`db-3.tar.gz` などの多くのファイルが含まれています。ここに挙げたファイルのみをリストする1行のコマンドはどうなりますか？

```
$ ls /home/lpi/database/db-[1-3].tar.gz
```

7. 以下のリストについて回答して下さい。

```
$ ls
cne1222223.pdf cne12349.txt cne1234.pdf
```

グロブ文字を1つ使用して、PDFファイルのみを削除するコマンドはどうなりますか？

```
$ rm *.pdf
```

103.3 レッスン 2

Certificate:	LPIC-1
Version:	5.0
Topic:	103 GNUおよびUnixコマンド
Objective:	103.3 基本的なファイル管理
Lesson:	2 of 2

はじめに

ファイルを見つける方法

システムを使用していると、徐々にファイルの数が増えて消費容量が大きくなっていくため、使いたいファイルを見つけるのが難しくなります。Linuxには、ファイルをすばやく検索して見つけるための `find` コマンドが備わっています。`find` の構文は次の通りです。

```
find STARTING_PATH OPTIONS EXPRESSION
```

`STARTING_PATH`
検索を開始するディレクトリを指定します。

`OPTIONS`
検索の条件や動作を制御します。

`EXPRESSION`
検索条件を定義します。

```
$ find . -name "myfile.txt"
./myfile.txt
```

この例での検索開始パスは、現在のディレクトリです。`-name` は条件式のひとつで、ファイル名を指定するものです。ここでは `myfile.txt` という名前のファイルを探しています。（検索するファイル名に）グロブ文字を使用する場合は、ファイル名全体を必ず引用符で囲みます。

```
$ find /home/frank -name "*.png"
/home/frank/Pictures/logo.png
/home/frank/screenshot.png
```

このコマンドは、`/home/frank/` ディレクトリの下にある `.png` で終わるファイルを検索します。アスタリスク（`*`）の使用法については、前のレッスンを参照してください。

条件を追加して検索を高速化する

`find` コマンドは、種別、サイズ、日時 に基づいてファイルを検索することもできます。条件で絞り込むことにより、より短い時間で目的の結果を得ることができます。

ファイル種別による絞り込みを行うには、以下の条件式を使います。

`-type f`
通常ファイルのみを探す。

`-type d`
ディレクトリのみを探す。

`-type l`
シンボリックリンクのみを探す。

```
$ find . -type d -name "example"
```

このコマンドは、現在のディレクトリの下で、名前が `example` であるディレクトリを探します。

`find` で使用できる条件式には、次のようなものがあります。

`-name`
指定された名前を検索します。

`-iname`
名前を検索しますが、大文字と小文字を区別しません（つまり、`myFile` と `MYFILE` は区別されません）。

`-not`
条件に一致 しない ものに一致します（条件反転）。

`-maxdepth N`
現在のディレクトリから `N` 階層下までを検索します。

更新日時によるファイルの検索

`find` では、ファイルの更新日時に基づいて絞り込むこともできます。

```
$ sudo find / -name "*.conf" -mtime 7
/etc/logrotate.conf
```

このコマンドは、ファイルシステム全体（開始パスがルートディレクトリ `/`）から、文字列 `.conf` で終わり、7日以内に変更されたすべてのファイルを検索します。ここでは、管理者しかアクセスできないディレクトリが含まれているため、`sudo` を使用しています。 `mtime` に渡される引数は、ファイルが最後に変更されてからの 日数 を示します。

サイズによるファイルの検索

`find` では、ファイル サイズ に基づいて絞り込むこともできます。たとえば、`/var` の下から `2G` より大きいファイルを検索してみましょう。

```
$ sudo find /var -size +2G
/var/lib/libvirt/images/debian10.qcow2
/var/lib/libvirt/images/rhel8.qcow2
```

`-size` 条件式は、引数に指定したファイルサイズの条件を満たすものに絞り込みます。引数の例を以下に示します:

`-size 100b`
ちょうど100バイトのファイル

`-size +100k`
100キロバイトを超えるファイル。

`-size -20M`
20メガバイト未満のファイル。

`-size +2G`
2ギガバイトを超えるファイル。

> **NOTE** 空のファイルを見つけるにはこうします: `find . -size 0b` または `find . -empty`。

結果に基づくアクション

`-exec` 条件式は、絞り込まれたファイルごとに、指定したアクション（コマンド）を実行します。

```
$ find . -name "*.conf" -exec chmod 644 '{}' \;
```

このコマンドでは、現在のディレクトリ（`.`）以下の、名前が `.conf` で終わるファイル名に絞り込んで、見つけたファイルごとに `chmod 644` コマンドを実行してパーミッションを変更します。

`'{}' \;` の意味は後で説明しますので、今のところは気にしないでください。

検索結果から `grep` で内容をフィルタリングする

`grep` を使って、キーワードの出現箇所を検索します。

検索されたファイルから、内容をフィルタリングすることを考えてみましょう。

```
$ find . -type f -exec grep "lpi" '{}' \; -print
Alpine/M
helping/M
./.bash_history
```

このコマンドは、現在のディレクトリ（`.`）以下から通常ファイル（`-type f`）を検索して、絞り込まれたファイルごとにコマンド `grep "lpi"` を実行します。`grep` コマンドがステータスコードとして0（成功）を返した場合（すなわち指定した文字列をファイル内で見つけた場合）には、さらにそのファイル名を表示します（`-print`）。

中括弧（`{}`）は、`find` が絞り込んだ結果のファイル名を保持するプレースホルダーです。`{}` には特殊文字が含まれることがあるので、`grep` がファイル名と解釈するように、シングルクォート（`'`）で囲んでいます。

`-exec` 条件式は、セミコロン（;）までをその引数として解釈します。`;` はシェルの特殊記号でもあるので、シェルが解釈してしまわないように、エスケープ（`\;`）する必要があります。

コマンドの最後に `-delete` オプションを指定すると、そこまでの検索条件に一致したすべてのファイルを削除します。このオプションは便利ですが、まず検索条件に一致するファイル名を `-print` で確認してから実行するとよいでしょう。

以下の `find` コマンドは、現在のディレクトリ以下を検索し、ファイル名が `.bak` で終わるすべてのファイルを削除します。

```
$ find . -name "*.bak" -delete
```

ファイルのアーカイブ

`tar` コマンド（アーカイブと圧縮）

“tape archive(r)” の略である `tar` コマンドは、複数のファイルを1つのファイルにまとめた「tarアーカイブ」を作成します。一群のファイルをまとめて移動したり、バックアップしたりするために、アーカイブを利用します。`tar` コマンドは、移動やバックアップを行うために、複数のファイルを1つにまとめるツールだと考えればよいでしょう。

1つの `tar` コマンドで、tarアーカイブを展開したり、アーカイブに含まれているファイルのリストを表示したり、既存のアーカイブにファイルを追加したりできます。

`tar` コマンドの構文は次のとおりです。

```
tar [OPERATION_AND_OPTIONS] [ARCHIVE_NAME] [FILE_NAME(S)]
```

OPERATION

以下のうち、いずれか1つのオプションが必須です。よく使用する操作は次のとおりです。

`--create` (`-c`)

新しいtarアーカイブを作成します。

`--extract` (`-x`)

アーカイブから、すべてないし1つ以上のファイルを取り出します（展開）。

`--list` (`-t`)

アーカイブに含まれているファイルのリストを表示します。

OPTIONS
よく使われるオプションには次のようなものがあります。

--verbose（**-v**）
`tar` コマンドが処理しているファイルの名前を表示します。

--file=archive-name（**-f archive-name**）
アーカイブファイルの名前を指定します。

ARCHIVE_NAME
アーカイブのファイル名。

FILE_NAME(S)
アーカイブに入れる、あるいは、アーカイブから取り出すファイル名のリストです。スペースで区切って複数を指定できます。展開時に指定しないと、アーカイブ全体が抽出されます。

アーカイブの作成

現在のディレクトリに `stuff` という名前のディレクトリがあり、これを `archive.tar` という名前のファイルに保存するには、次のコマンドを実行します。

```
$ tar -cvf archive.tar stuff
stuff/
stuff/service.conf
```

このコマンドにおけるオプションの意味は次のとおりです。

-c
アーカイブを作成します。

-v
“verbose”（冗長）モード。アーカイブの処理中に、画面上に進行状況を表示します。`-v` オプションを持つコマンドが、他にもたくさんあります。

-f
アーカイブのファイル名を指定します。

1つのファイルや、1つのディレクトリをアーカイブするには、次のコマンドを使用します。

```
tar -cvf NAME-OF-ARCHIVE.tar /PATH/TO/DIRECTORY-OR-FILE
```

NOTE `tar` は再帰的に機能するため、ディレクトリを指定するとそのディレクトリの下すべてがアーカイブ対象となります。

複数のディレクトリをまとめてアーカイブするには、`/PATH/TO/DIRECTORY-OR-FILE` の部分に、すべてのディレクトリを空白で区切って指定します。

```
$ tar -cvf archive.tar stuff1 stuff2
```

このコマンドは、`stuff1` と `stuff2` を、1つのアーカイブファイル `archive.tar` にまとめます。

アーカイブの抽出

同じ `tar` コマンドで、アーカイブからファイルを取り出すことができます。

```
$ tar -xvf archive.tar
stuff/
stuff/service.conf
```

このコマンドは、`archive.tar` の内容を現在のディレクトリに展開します。

`tar` コマンドによる展開は、アーカイブ作成時の `-c` オプションの代わりに、`-x` オプションを使用するだけです。

アーカイブの内容を展開するディレクトリを指定するには、`-C` オプションを使用します。

```
$ tar -xvf archive.tar -C /tmp
```

このコマンドは、`archive.tar` の内容を `/tmp` ディレクトリに展開します。

```
$ ls /tmp
stuff
```

`tar` で圧縮する

多くのLinuxディスリビューションに含まれている GNU `tar` コマンドは、`.tar` アーカイブを作成しながら、それを `gzip` や `bzip2` 形式などで圧縮することができます。

```
$ tar -czvf name-of-archive.tar.gz stuff
```

このコマンドは、gzip 圧縮形式（-z）を使用して、tar アーカイブを圧縮したファイルを作成します。

gzip 圧縮によって .tar.gz や .tgz ファイルを作成することが一般的ですが、tar は bzip2 圧縮もサポートしています。つまり、.tar.bz2、.tar.bz、あるいは .tbz ファイルと呼ばれる bzip2 形式の圧縮ファイルを作成することもできます。

gzip 圧縮の時には -z を指定しましたが、bzip2 圧縮の時には、-j を指定します。

```
$ tar -cjvf name-of-archive.tar.bz stuff
```

ファイルを展開する場合は、-c ではなく -x を指定します。ここで、x は “extract” を意味します。

```
$ tar -xzvf archive.tar.gz
```

gzip は処理が高速ですが圧縮率がやや低くく、圧縮後のファイルが bzip2 と比較するとやや大きくなります。bzip2 は処理が低速ですが圧縮率がやや高く、gzip と比較するとファイルがやや小さくなります。とはいえ、gzip も bzip2 も圧縮するという点では実質的に同じであり、圧縮率の違い以外はどちらも同じように扱うことができます。

訳注: より圧縮率の高い xz 圧縮も使用できます。tar で xz 圧縮を使う場合には、-J オプションを指定します。

なお、圧縮だけを行うコマンドがあります。gzip 圧縮には gzip コマンドを、bzip2 圧縮には bzip2 コマンドを使用します。たとえば、ファイルを gzip 圧縮するには、次のようにします。

```
gzip FILE-TO-COMPRESS
```

gzip は、対象ファイルの名前の末尾に .gz を追加した圧縮ファイルを作成します。圧縮ファイルの作成後に、元のファイルは削除されます。

bzip2 コマンド（ならびに xz コマンド）の操作も同様です。

ファイルを伸張するには、ファイルの圧縮に使用した圧縮形式に応じて、gunzip、bunzip2、unxz のいずれかを使用します。

cpio コマンド

cpio は、“copy in, copy out” の略です。このコマンドは、*.cpio 形式や *.tar 形式などのアーカイブファイルを処理します。

`cpio` は次のような処理を行います。

- ファイルをアーカイブにコピーする。
- アーカイブからファイルを抽出する。

`cpio` コマンドは、まず標準入力（主に `ls` から出力）から、アーカイブに入れるファイルのリストを取得します。

すなわち、`cpio` アーカイブを作成するには、次のようにします:

```
$ ls | cpio -o > ~/archive.cpio
```

`-o` オプションは、取得したリストに掲載されたファイルを、アーカイブ形式のファイルに出力することを指示します。この例では、作成される出力ファイルは（リダイレクトで指定している）`~/archive.cpio` です。ここでの `ls` コマンドは、現在のディレクトリの内容をリストします。（訳注: シェルはlsコマンドの実行に先立って、リダイレクトの出力先となるファイルを作成します。そのため、作業ディレクトリの内容をアーカイブする場合には、出力先の `archive.cpio` を別のディレクトリに置く必要があります。）

アーカイブを展開するには：

```
$ cpio -id < archive.cpio
```

`-i` オプションは、アーカイブからファイルを抽出することを指定します。`-d` オプションは、必要に応じて展開先のフォルダを作成します。文字 `<` は標準入力からのリダイレクトを示します。この例では、展開する入力ファイルは `archive.cpio` です。

dd コマンド

`dd` は、ある場所から別の場所にデータを（変換しながら）コピーします。`dd` のコマンドライン構文は他の多くのコマンドとは異なり、GNUツールで標準的な `-option value` や `--option=value` 形式ではなく、オプションを `option=value` という構文で指定します。

```
$ dd if=oldfile of=newfile
```

このコマンドは、`oldfile` の内容を `newfile` にコピーします。ここで、`if=` は入力ファイル（つまりコピー元）を、`of=` は出力ファイル（つまりコピー先）を示します。

NOTE

デフォルトでは `dd` コマンドは、コマンドの処理が終了するまで画面に何も出力しません。`status=progress` オプションを指定すると、エラー出力にコマンドの進捗状況が送られます。

> 例： `dd status=progress if=oldfile of=newfile`。

`dd` コマンドを使うと、データの大文字/小文字を変更したり、`/dev/sdb` などのブロックデバイスに直接書き込んだりすることもできます。

```
$ dd if=oldfile of=newfile conv=ucase
```

このコマンドでは、`oldfile` のすべての内容が `newfile` にコピーされ、すべての英小文字が大文字に変換されます。

次のコマンドは、`/dev/sda` にあるハードディスク全体を、`backup.dd` という名前のファイルにバックアップします。

訳注: `dd` コマンドはメインフレームとのデータ交換に由来するコマンドです。現在では、この例のように ブロックデバイス を丸ごとコピーしたり、バックアップするために主に使われています。`bs` オプションは、1度に読み書きするデータブロックのサイズを指定しています。

```
$ dd if=/dev/sda of=backup.dd bs=4096
```

演習

1. 次の実行例について考察して下さい。

```
$ find /home/frank/Documents/ -type d
/home/frank/Documents/
/home/frank/Documents/animal
/home/frank/Documents/animal/domestic
/home/frank/Documents/animal/wild
```

   - 実行したコマンドは、何を出力していますか？

   - 検索を始めるディレクトリはどこですか？

2. バックアップを圧縮したいと考えています。次のコマンドを実行しました。

```
$ tar cvf /home/frank/backup.tar.gz /home/frank/dir1
```

   バックアップを `gzip` 形式で圧縮したいのですが、オプションが不足しています。不足しているオプションは何ですか？

発展演習

1. `/var` にある名前が `.backup` で終わるファイルが容量を消費するので、システム管理者として定期的にチェックする必要があるとします。
   - `find` を使って、これらのファイルを見つけるコマンドはどうなりますか？
   - 分析したところ、それらのファイルのサイズは `100M` から `1000M` でした。前のコマンドにこの条件を追加して、`100M` から `1000M` の `.backup` ファイルを見つけるようにしてください。
   - 最後に、見つけたファイルを削除するように、コマンドに削除操作を追加してください。

2. `/var` ディレクトリに、次の4つのバックアップファイルがあります。

```
db-jan-2018.backup
db-feb-2018.backup
db-march-2018.backup
db-apr-2018.backup
```

   - `tar` を使用して `db-first-quarter-2018.backup.tar` という名前のアーカイブファイルを作成してください。
   - アーカイブを `gzip` 形式で圧縮する `tar` コマンドはどうなりますか？ アーカイブのファイル名は、`.gz` で終わる必要があります。

まとめ

このセクションでは、以下のことを学びました。

- `find` でファイルを見つける方法。
- `find` に条件式を指定して、日時、ファイル種別、サイズに基づく検索条件を追加する方法。
- 見つけたファイルを処理する方法。
- `tar` を使用してファイルをアーカイブ、圧縮、伸張する方法。
- `cpio` でアーカイブを処理する方法。
- `dd` でファイルをコピーする方法。

演習の解答

1. 次の実行例について考えて下さい。

```
$ find /home/frank/Documents/ -type d
/home/frank/Documents/
/home/frank/Documents/animal
/home/frank/Documents/animal/domestic
/home/frank/Documents/animal/wild
```

   - 実行したコマンドは、何を出力していますか？

     ディレクトリ

   - 検索を始めるディレクトリはどこですか？

     `/home/frank/Documents`

2. バックアップを圧縮したいと考えています。次のコマンドを実行しました。

```
$ tar cvf /home/frank/backup.tar.gz /home/frank/dir1
```

   バックアップを `gzip` 形式で圧縮したいのですが、オプションが不足しています。不足しているオプションは何ですか？

   `-z` オプション

発展演習の解答

1. `/var` にある名前が `.backup` で終わるファイルが容量を消費するので、システム管理者として定期的にチェックする必要があるとします。
   - `find` を使って、これらのファイルを見つけるコマンドはどうなりますか？

     ```
     $ find /var -name "*.backup"
     ```

   - 分析したところ、それらのファイルのサイズは `100M` から `1000M` でした。前のコマンドにこの条件を追加して、`100M` から `1000M` の `.backup` ファイルを見つけるようにしてください。

     ```
     $ find /var -name "*.backup" -size +100M -size -1000M
     ```

   - 最後に、見つけたファイルを削除するように、コマンドに削除操作を追加してください。

     ```
     $ find /var -name "*.backup" -size +100M -size -1000M -delete
     ```

2. `/var` ディレクトリには、次の4つのバックアップファイルがあります。

   ```
   db-jan-2018.backup
   db-feb-2018.backup
   db-march-2018.backup
   db-apr-2018.backup
   ```

   - `tar` を使用して `db-first-quarter-2018.backup.tar` という名前のアーカイブファイルを作成してください。

     ```
     $ tar -cvf db-first-quarter-2018.backup.tar db-jan-2018.backup db-feb-
     2018.backup db-march-2018.backup db-apr-2018.backup
     ```

   - アーカイブを `gzip` 形式で圧縮する `tar` コマンドはどうなりますか？ アーカイブのファイル名は、`.gz` で終わる必要があります。

     ```
     $ tar -zcvf db-first-quarter-2018.backup.tar.gz db-jan-2018.backup db-feb-
     2018.backup db-march-2018.backup db-apr-2018.backup
     ```

103.4 ストリーム、パイプ、リダイレクトを使用する

LPI目標への参照

LPIC-1 v5, Exam 101, Objective 103.4

総重量

4

主な知識分野

- 標準入力、標準出力、および標準エラーのリダイレクト。
- あるコマンドの出力を、別のコマンドの入力にパイプでつなぐ。
- あるコマンドの出力を別のコマンドの引数として使用する。
- stdoutとファイルの両方に出力を送る。

用語とユーティリティ

- `tee`
- `xargs`

103.4 レッスン 1

Certificate:	LPIC-1 (101)
Version:	5.0
Topic:	103 GNUおよびUnixコマンド
Objective:	103.4 ストリーム、パイプ、リダイレクトを使う
Lesson:	1 of 2

はじめに

すべてのコンピュータプログラムは、同じ原則に従います。すなわち、何らかのソースデータを読み込んで、それらを加工して結果を出力することです。Linuxシェルの場合は、ローカルファイル、リモートファイル、キーボードなどのデバイスが、データソースとなります。出力は画面に表示されるのが一般的ですが、出力データをローカルファイルシステムに保存したり、リモートデバイスに送信したり、あるいはオーディオスピーカーで再生することなどもあります。

LinuxのようにUnixに触発されたオペレーティングシステムは、多種多様な入出力を提供します。整数とデータチャンネルを動的に関連付ける　ファイル記述子　の概念では、プロセスはその入力/出力データストリームを、整数値で参照することができます。

標準的なLinuxプロセスは、デフォルトで3つの通信チャネル、標準入力　チャネル（stdin）、標準出力　チャネル（stdout）、および　標準エラー　チャネル（stderr）が開かれています。これらのチャネルに割り当てられたファイル記述子（整数）は、stdinが `0`、stdoutが `1` 、stderrが `2` です。通信チャネルには、特別なデバイス `/dev/stdin`、`/dev/stdout`、`/dev/stderr` を介してアクセスすることもできます。

プログラミングの際には、これらの3つの標準通信チャネルを使えば、データを入出力するメディアの種類を気にしなくても、読み取りと書き込みを行うコードを記述できます。たと

えば、プログラムが一群のデータを入力として必要とする場合、標準入力からデータを読み出せば、そのチャンネルに関連付けられている実体（ファイルやデバイス）がすべてのデータを提供します。同様に、プログラムが処理結果を出力するは、それを標準出力に書き込むのが一番単純です。通常のシェルセッションでは、キーボードがstdinに関連付けられ、モニター画面がstdoutとstderrの両方に関連付けられます。

Bashシェルには、プログラムのロード時に、これらの通信チャネルを再割り当てする機能が備わっています。これにより、たとえばstdoutをモニター画面からローカルファイルに切り替えて使用する事ができます。

リダイレクト

シェル環境における、ファイル記述子に対するチャネルの再割り当てを　リダイレクト　と呼びます。リダイレクトは、コマンドライン内の特殊文字で指定します。たとえば、プロセスの標準出力をファイルにリダイレクトするには、コマンドの最後に　大なり記号　`>`　を置き、リダイレクトされた出力を受け取るファイルへのパスを続けます。

```
$ cat /proc/cpuinfo >/tmp/cpu.txt
```

デフォルトでは、stdout　　に送信されるコンテンツのみがリダイレクトされます。詳しく述べると、Bashでは大なり記号の直前に、切り替えるファイル記述子（数値）を指定するのですが、省略された場合には標準出力をリダイレクトします。つまり、`>`　のみを指定することは、`1>` を指定することと同じです (stdout のファイル記述子は `1` です)。

stderrのコンテンツをリダイレクトするには、代わりにリダイレクトとして　`2>`　を指定します。ほとんどのコマンドラインプログラムは、デバッグ情報とエラーメッセージを標準エラーチャネルに送信します。たとえば、存在しないファイルを読み取ろうとしたときに表示されるエラーメッセージをファイルにキャプチャするには、次のようにします。

```
$ cat /proc/cpu_info 2>/tmp/error.txt
$ cat /tmp/error.txt
cat: /proc/cpu_info: No such file or directory
```

stdoutとstderrの両方を同じターゲットにリダイレクトするには、`&>`　ないし　`>&`　を使用します。大なり記号（>）とアンパサンド（&）の間に空白を置いてはいけません。空白を置くと、Bashはアンパサンドを、バックグラウンドでプロセスを実行する指示とみなして、リダイレクトは行われません。

リダイレクト先のターゲットは、書き込み可能なファイルへのパス（`/tmp/cpu.txt`　　など）、ないしは、書き込み可能なファイル記述子であることが必要です。ターゲットにファイル記述子を指定するには、アンパサンドに続けてファイル記述子（数値）を置きます。たとえば、`1>&2`　はstdoutをstderrにリダイレクトします。逆に、stderrをstdoutにリダイレクトするには、`2>&1` を指定します。

あまり便利とは言えませんが、stdoutをファイルにリダイレクトしておき、さらにstderrをstdoutにリダイレクトすれば、短く書くことができます。たとえば、stderrとstdoutの両方を `log.txt` という名前のファイルに書き込むリダイレクトは、`>log.txt 2>&1` と書くことができます。なお、stderrをstdoutにリダイレクトする主な理由は、デバッグメッセージやエラーメッセージを解析するためです。プログラムの標準出力を別のプログラムの標準入力にリダイレクトすること（パイプ）はできますが、標準エラーを別のプログラムの標準入力に直接リダイレクトすることはできません。つまり、別のプログラムのstdinに繋げられるように、stderrに送られるメッセージを、stdoutにリダイレクトする必要があります。

コマンドの出力を破棄したければ、その内容を特別なファイル `/dev/null` にリダイレクトします。たとえば `>log.txt 2>/dev/null` は、stdoutの出力をファイル `log.txt` に保存し、stderrを破棄します。ファイル `/dev/null` はどのユーザーでも書き込み可能ですが、書き込んだ内容はどこにも保存されずに破棄されます。

指定されたターゲットが書き込み可能でなく（パスがディレクトリまたは読み取り専用ファイルを指している場合）、変更できない場合はエラーメッセージが表示されます。ターゲットが書き込み可能であれば、出力リダイレクトは、確認なしで既存のターゲットを上書きします。Bashのオプション `noclobber` を有効とすれば、既存のファイルの上書きを禁止できます。このオプションを現在のセッションで有効にするには、コマンド `set -o noclobber` ないし `set -C` を使用します。

```
$ set -o noclobber
$ cat /proc/cpu_info 2>/tmp/error.txt
-bash: /tmp/error.txt: cannot overwrite existing file
```

現在のセッションの `noclobber` オプションを無効化するには、`set +o noclobber` ないし `set +C` を実行します。`noclobber` オプションを永続化するには、ユーザーのBashプロファイルまたはシステム全体のプロファイルにコマンドを置く必要があります。

リダイレクトされたデータを既存のコンテンツに追記するには、2つの大なり記号 `>>` を使います。この機能は、`noclobber` オプションを有効にしても禁止できません。

```
$ cat /proc/cpu_info 2>>/tmp/error.txt
$ cat /tmp/error.txt
cat: /proc/cpu_info: No such file or directory
cat: /proc/cpu_info: No such file or directory
```

前の例では、ファイル `/tmp/error.txt` に、新しいエラーメッセージが追加されました。ファイルが存在していない場合は、新しいファイルが作成されます。

プロセスの標準入力に割り当てられるデータソースを切り替えることもできます。プロセスのstdinに、ファイルのコンテンツをリダイレクトするためには、小なり記号 `<` を使います。この場合、データは右から左に流れます。小なり記号の左にリダイレクトするファイル

記述子0があるものと見なされ、右側にデータソース（ファイルのパス）を指定します。次に示す `uniq` コマンドは、ほとんどのテキスト処理ユーティリティと同様に、デフォルトでstdinに送られるデータを読み込みます。

```
$ uniq -c </tmp/error.txt
      2 cat: /proc/cpu_info: No such file or directory
```

`uniq` に `-c` オプションを指定すると、テキストにある同一行の出現回数を表示します。リダイレクトされたファイル記述子が省略されているので、このコマンド例は `uniq -c 0</tmp/error.txt` と同等です。入力リダイレクトで `0` 以外のファイル記述子を使用することは、原則としてありません（ファイル記述子 `3` や `4` を読み取る特別なプログラムでのみ意味があります）。技術的には、データ入出力に3以上のファイル記述子を使用するプログラムを作成することもできます（が、一般的でありません）。たとえば、次のCコードは、ファイル記述子 `3` からデータを読み取り、それをファイル記述子 `4` にコピーします。

NOTE プログラムは、そのようなファイル記述子を正しく処理しなくてはなりません。誤った無効な読み書きによって、クラッシュする可能性があります。

```
#include <stdio.h>

int main(int argc, char **argv){
  FILE *fd_3, *fd_4;
  // Open file descriptor 3
  fd_3 = fdopen(3, "r");
  // Open file descriptor 4
  fd_4 = fdopen(4, "w");
  // Read from file descriptor 3
  char buf[32];
  while ( fgets(buf, 32, fd_3) != NULL ){
    // Write to file descriptor 4
    fprintf(fd_4, "%s", buf);
  }
  // Close both file descriptors
  fclose(fd_3);
  fclose(fd_4);
}
```

このプログラムをテストするには、サンプルコードを `fd.c` として保存し、`gcc -o fd fd.c` でコンパイルします。このプログラムを実行するには、ファイル記述子3から読み込み、ファイル記述子4に書き出せるようにする必要があります。例として、ファイル記述子 `3` に以前に作成しファイル `/tmp/error.txt` を割り当てて読み込ませ、ファイル記述子 `4` をstdoutにリダイレクトしましょう。

```
$ ./fd 3</tmp/error.txt 4>&1
cat: /proc/cpu_info: No such file or directory
cat: /proc/cpu_info: No such file or directory
```

プログラマーの観点からは、ファイル記述子を使用することで、オプションの解析やパス名の処理を省くことができます。その際には、たとえば `3<>/tmp/error.txt` のように小なり記号と大なり記号の両方を使用して、読み書き両用のファイル記述子を定義します。（訳注: 2004年以前のバージョンのbashでは利用できません。）

ヒアドキュメントとヒア文字列

入力をリダイレクトする別の方法として、ヒアドキュメント と ヒア文字列 があります。ヒアドキュメントでは、複数行のテキストをリダイレクトされたコンテンツとして入力できます。2つの小なり記号 `<<` が、ヒアドキュメントによるリダイレクトを示します。

```
$ wc -c <<EOF
> How many characters
> in this Here document?
> EOF
43
```

2つの小なり記号 `<<` の後には、終了を示す単語としてここでは `EOF` を指定しています。終了を示す単語のみの行を入力すると、入力が終了します。別の単語を終了の印として使用できます。この例では、`wc -c` コマンドのstdinに、2行のテキストが送信されて、その文字数が表示されます。入力リダイレクトと同様に、ファイル記述子を省略した場合は、stdin（ファイル記述子 `0`）と見なされます。

ヒア文字列は、ヒアドキュメントによく似ていますが、主に1行のみの入力に使用します。

```
$ wc -c <<<"How many characters in this Here string?"
41
```

この例では、`wc -c` のstdinに3つの小なり記号の右側の文字列が送信され、その文字数がカウントされます。文字列に空白が含まれる場合は、引用符で囲みます。そうしないと、最初の単語のみがヒア文字列として使用されて、残りの単語はコマンドの引数として渡されます。

演習

1. `cat` コマンドは、テキストファイルだけではなく、ブロックデバイスの内容などのバイナリデータを別のファイルに送ることもできます。`cat` とリダイレクトを使用して、デバイス `/dev/sdc` の内容を、現在のディレクトリの `sdc.img` ファイルに送るにはどうしますか？

2. `date 1> now.txt` コマンドでリダイレクトされたチャネルの名前は何ですか？

3. リダイレクトを使用してファイルを上書きしようとした時に、`noclobber` オプションが有効になっていることを通知するエラーが表示されました。現在のセッションで `noclobber` オプションを無効にするにはどうしますか？

4. `cat <<.>/dev/stdout` を実行するとどうなりますか？

発展演習

1. `cat /proc/cpu_info` コマンドを実行すると、`/proc/cpu_info` が存在しないというエラーメッセージが表示されました。`cat /proc/cpu_info 2>1` を実行すると、エラーメッセージはどこにリダイレクトされますか？

2. シェルセッションで `noclobber` オプションが有効になっている場合、`/dev/null` にコンテンツを送信して捨てることはできますか？

3. `echo` を使用せずに、変数 `$USER` の内容を、`sha1sum` のstdinにリダイレクトするにはどうしますか？

4. Linuxでは、IDが PID であるプロセスによって開かれたすべてのファイルへのシンボリックリンクが、`/proc/PID/fd/` ディレクトリに置かれます。そのディレクトリを使用し、`nginx` のログファイルがどこにあるかを調べるにはどうしますか？ `nginx` のPIDは `1234` であると仮定します。

5. 単純な算術計算はシェルの組み込みコマンドだけで行えますが、浮動小数点計算には、`bc`（basic calculator）などの特別なプログラムを使います。`bc` では `scale` パラメータをを使用して小数点以下の桁数を指定することができますが、通常は対話モードの標準入力からしか、それらのパラメータや数式を受け付けません。ヒア文字列を使って、`bc` の標準入力に計算式 `scale=6; 1/3` を送るにはどうしますか？

まとめ

このレッスンでは、標準的な入出力チャネルをリダイレクトしてプログラムを実行する方法について説明しました。Linuxプロセスは所定の ファイル記述子（0から2）に標準的なチャンネルが接続されているものとしてデータを読み書きし、シェルがそれらのチャンネルを任意のファイルやデバイスに変更します。このレッスンでは、次を説明しました。

- ファイル記述子とは何か、また、Linuxにおけるその役割。
- プロセスの標準通信チャネル: stdin、stdout、stderr。
- 入力と出力の両方でデータをリダイレクトしてコマンドを実行する方法。
- ヒアドキュメント と ヒア文字列 を使用する入力のリダイレクト方法。

以下のコマンドと手順を取り上げました:

- リダイレクト演算子: `>`、`<`、`>>`、`<<`、`<<<`
- コマンド: `cat`、`set`、`uniq`、`wc`

演習の解答

1. `cat` コマンドは、テキストファイルだけではなく、ブロックデバイスの内容などのバイナリデータを別のファイルに送ることもできます。`cat` とリダイレクトを使用して、デバイス `/dev/sdc` の内容を、現在のディレクトリの `sdc.img` ファイルに送るにはどうしますか？

   ```
   $ cat /dev/sdc > sdc.img
   ```

2. `date 1> now.txt` コマンドでリダイレクトされたチャネルの名前は何ですか？

   標準出力またはstdout

3. リダイレクトを使用してファイルを上書きしようとした時に、`noclobber` オプションが有効になっていることを通知するエラーが表示されました。現在のセッションで `noclobber` オプションを無効にするにはどうしますか？

   `set +C` または `set +o noclobber`

4. コマンド `cat <<.>/dev/stdout` の結果はどうなりますか？

   Bashはヒアドキュメントの入力モードに入り、ピリオドのみを入力すると終了します。入力されたテキストはstdoutにリダイレクトされて、画面に表示されます。

発展演習の解答

1. `cat /proc/cpu_info` コマンドを実行すると、`/proc/cpu_info` が存在しないというエラーメッセージが表示されました。`cat /proc/cpu_info 2>1` を実行すると、エラーメッセージはどこにリダイレクトされますか？

   現在ディレクトリの `1` という名前のファイル。

2. シェルセッションで `noclobber` オプションが有効になっている場合、`/dev/null` にコンテンツを送信して捨てることはできますか？

   できます。`/dev/null` は `noclobber` の影響を受けない特別なファイルです。

3. `echo` を使用せずに、変数 `$USER` の内容を、`sha1sum` のstdinにリダイレクトするにはどうしますか？

   ```
   $ sha1sum <<<$USER
   ```

4. Linuxでは、IDが PID であるプロセスによって開かれたすべてのファイルへのシンボリックリンクが、`/proc/PID/fd/` ディレクトリに置かれます。そのディレクトリを使用し、`nginx` のログファイルがどこにあるかを調べるにはどうしますか？ `nginx` のPIDは `1234` であると仮定します。

   `ls -l /proc/1234/fd` を実行して、ディレクトリ内のシンボリックリンクが指すファイルを調べます。

5. 単純な算術計算はシェルの組み込みコマンドだけで行えますが、浮動小数点計算には、`bc`（basic calculator）などの特別なプログラムを使います。`bc` では `scale` パラメータを使用して小数点以下の桁数を指定することができますが、通常は対話モードの標準入力からしか、それらのパラメータや数式を受け付けません。ヒア文字列を使って、`bc` の標準入力に計算式 `scale=6; 1/3` を送るにはどうしますか？

   ```
   $ bc <<<"scale=6; 1/3"
   ```

103.4 レッスン 2

Certificate:	LPIC-1 (101)
Version:	5.0
Topic:	103 GNUおよびUnixコマンド
Objective:	103.4 ストリーム、パイプ、リダイレクトを使用する
Lesson:	2 of 2

はじめに

Unix哲学の1つに、それぞれのプログラムはその目的に専念し、範囲外の機能を取り入れべきではない、というものがあります（訳注:　日本語版Wikipedia「UNIX哲学」に詳しくまとめられています）。つまり、複数のプログラムを連携させれば複雑な結果を生成できるので、物事を単純に保つことが単純な結果しか生み出せないということは　ない　ことを意味しています。縦棒　|　は、パイプ　とも呼ばれ、あるプログラムの出力を別のプログラムの入力に直接接続するパイプラインを作ります。また、コマンド置換　　を使ってプログラムの出力を変数に格納して、別のコマンドの引数として使用することもできます。

パイプ

リダイレクトとは異なり、パイプではデータがコマンドラインの左から右に流れます。送り先は別のプロセスであり、ファイル名やファイル記述子、ヒアドキュメントではありません。パイプ　|　は、前のコマンドの出力を次のコマンドの入力に接続し、すべてのコマンドを同時に起動することをシェルに指示します。たとえば、次のコマンドは、`cat`　によって標準出力に送られた `/proc/cpuinfo` ファイルの内容を、`wc` のstdinにパイプします。

```
$ cat /proc/cpuinfo | wc
   208    1184    6096
```

`wc` に入力ファイルを指定しない場合、stdinから読み込んだテキストの行数、単語数、文字数をカウントします。1つの複合コマンドに、複数のパイプを使うことができます。次の例では、2つのパイプを使用しています。

```
$ cat /proc/cpuinfo | grep 'model name' | uniq
model name      : Intel(R) Xeon(R) CPU           X5355  @ 2.66GHz
```

`cat /proc/cpuinfo` は `/proc/cpuinfo` の内容を、`grep 'model name'` にパイプします。`grep` コマンドは `model name` という文字列を含む行のみを選択しますが、このマシンには多くのCPUが搭載されているので、複数の行が出力されます。最後のパイプは、`grep 'model name'` の出力を `uniq` に接続します。`uniq` は、同じ行の繰り返しを1行にまとめます。

1つのコマンドライン中に、リダイレクトとパイプを組み合わせることもできます。前の例は、より単純な形式に書き直せます。

```
$ grep 'model name' </proc/cpuinfo | uniq
model name      : Intel(R) Xeon(R) CPU           X5355  @ 2.66GHz
```

`grep` の引数にファイルパスを指定することができるので、実は `grep` に入力リダイレクトを使う必要もありませんが、この例では、リダイレクトとパイプを組み合わせてコマンドを構築する方法を示しています。

パイプとリダイレクトは排他的です。つまり、1つのソースは、1つのターゲットのみにしかマップできません。しかし、`tee` プログラムを使えば、出力をファイルに書きながら、画面にも表示する（stdoutに出力する)ことができます。先行するプログラムがその出力を `tee` のstdinに送信し、`tee` にはデータを格納するためのファイル名を指定します。

```
$ grep 'model name' </proc/cpuinfo | uniq | tee cpu_model.txt
model name      : Intel(R) Xeon(R) CPU           X5355  @ 2.66GHz
$ cat cpu_model.txt
model name      : Intel(R) Xeon(R) CPU           X5355  @ 2.66GHz
```

連鎖の最後である `uniq` の出力が表示されると共に、`cpu_model.txt` ファイルに保存されます。`tee` に指定したファイルを上書きするのではなく追記するには、`tee` に `-a` オプションを指定します。

パイプに送られるのは、プロセスの標準出力のみです。長時間かかるコンパイル処理を行う場合などに、後で調べるために標準出力と標準エラーの両方をファイルに保存したいことがあります。例えば、次のコマンドは、現在のディレクトリに Makefile が無い場合に、エラーのみを出力します。

```
$ make | tee log.txt
make: *** No targets specified and no makefile found.  Stop.
```

`make` のエラーメッセージが画面に表示されますが、`tee` には取り込まれず、空のファイル `log.txt` ができました。パイプがstderrを取り込めるように、リダイレクトする必要があります。

```
$ make 2>&1 | tee log.txt
make: *** No targets specified and no makefile found.  Stop.
$ cat log.txt
make: *** No targets specified and no makefile found.  Stop.
```

この例では、`make` のstderrがstdoutにリダイレクトされたので、パイプを経由して `tee` がそれを取り込んで、画面に表示すると共に `log.txt` に保存します。エラーメッセージを保存しておけば、後で調べることができて便利です。。

コマンド置換

コマンドの出力を取り込む（保存する）別の方法に、コマンド置換 があります。バッククォート内にコマンドを置くと、Bashはそれを（実行したコマンドの）標準出力に置き換えます。次の例は、プログラムのstdoutを、別のプログラムの引数として使用する方法を示しています。

```
$ mkdir `date +%Y-%m-%d`
$ ls
2019-09-05
```

プログラム `date` の出力（YYYY-MM-DD 形式の日付）が、`mkdir` の引数として使用されて、日付を名前とするディレクトリが作成されます。バッククォートの代わりに `$()` を使用しても、同じ結果が得られます。

```
$ rmdir 2019-09-05
$ mkdir $(date +%Y-%m-%d)
$ ls
2019-09-05
```

同じ方法で、コマンドの出力を変数に格納できます。

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

```
GNU/Linux
```

`uname -o` コマンドは、稼働中のオペレーティングシステムの一般的な名前を出力します。コマンドの出力を変数に割り当てることは、スクリプトで非常に役立ち、さまざまな方法でデータを保存および評価することができます。

コマンドの出力内容によっては、コマンド置換が適切ではない場合があります。あるプログラムの出力を、別のプログラムの引数として使用するための仲介者となる `xargs` というコマンドがあります。`xarg` コマンドは、「`xargs` の引数に指定したコマンド」を実行しますが、その引数として「標準入力から受け取ったテキスト」を追加します。次に示す `xargs` の例は、`find` が見つけたファイル（8つ）を引数として、`identify` を実行します。

```
$ find /usr/share/icons -name 'debian*' | xargs identify -format "%f: %wx%h\n"
debian-swirl.svg: 48x48
debian-swirl.png: 22x22
debian-swirl.png: 32x32
debian-swirl.png: 256x256
debian-swirl.png: 48x48
debian-swirl.png: 16x16
debian-swirl.png: 24x24
debian-swirl.svg: 48x48
```

例で実行している `identify` プログラムは、ImageMagick（ほとんどの種類の画像ファイル扱うことができる描画ツール）の一部です。`find` が出力したパスを `xargs` が読み取って、それを引数に指定した `identify` を実行します。`identify` の `-format` オプションは、指定されたファイルの情報を表示します。`-format` は `identify` へのオプションであり、`xargs` のオプションではないことに注意してください。なお、`find` が見つけた画像ファイルは、Debianディストリビューションのロゴファイルです。

`xargs` に `-n 1` オプションを指定すると、1回のコマンド実行につき1つの引数のみを指定します。例に挙げたケースで `xargs -n 1` を指定すると、`find` が見つけた8つのファイルを `identify` に渡すのではなく、1つのファイルごとに `identify` が8つ起動されます。同様に、`-n 2` を指定すれば2つのファイルずつ `identify` が4つ起動されますし、`-n 3` を指定すれば3つのファイルで2つ、2つのファイルで1つの `identify` が起動されます。（訳注: 複数の `identify` が同時に起動されることに注意してください。`-n` に指定する数値は、最大の 引数の数 です）。また、`xargs` に（例に挙げた `find` のように）複数行からなる入力を与える場合は、`xargs` が起動するコマンドに与える引数の 行数 を `-L` オプションで制限することもできます。

NOTE

例のケースでは、`-n 1` や `-L 1` を指定した `xargs` を使用する必要はありません。`find` には、見つけたファイルごとに指定のコマンドを実行する `-exec` アクションが備わっているからです。

`find` が見つけたファイルのパス名に空白などが含まれている場合には、特別な処理が必要

になります。`find` で `-print0` アクションを使用すると、出力の区切りとして（改行ではなく）Null文字を使用します。さらに、`xargs` に `-0` オプションを指定して、区切りがNull文字であることを指示します。これで、空白をや特殊文字を含むパス名を正しく見分けることができます。次の例の出力は省略しています。

```
$ find . -name '*avi' -print0 -o -name '*mp4' -print0 -o -name '*mkv' -print0 |
xargs -0 du | sort -n
```

この例では、見つけたファイルが使用しているディスク容量を `du` で調べ、サイズ順に表示します。（出力は省略しました）。`find` では、検索条件ごとに、`-print0` アクションを指定することに注意してください。（訳注: 括弧で条件式をまとめる方法もあります。）

デフォルトでは、`xargs` は実行するコマンドの最後に、渡された引数を置きます。引数を置く位置を変更するには、`-I` オプションを使用します。

```
$ find . -mindepth 2 -name '*avi' -print0 -o -name '*mp4' -print0 -o -name '*mkv'
-print0 | xargs -0 -I PATH mv PATH ./
```

最後の例では、`find` が見つけたすべてのファイルが、現在のディレクトリに移動されます。`mv` では、ソースパスを先に指定しますから、`xargs` の `-I` オプションで引数に置き換える単語を指定し、`mv` コマンドの適切な位置にその単語を置きます。（訳注: `-I` オプションを指定すると、`-L 1` が指定されたのと同じ動作になります。）また、Null文字を区切り文字とすれば、引数に置き換える単語を引用符で囲む必要がなくなります。

演習

1. スクリプトを自動的に実行する場合は、実行された日時を保存しておくと便利です。`date +%Y-%m-%d` コマンドは、現在の日付を YYYY-MM-DD 形式で表示します。コマンド置換を使用して、このコマンドの出力を、`TODAY` というシェル変数に格納するにはどうしますか？

2. シェル変数 `TODAY` の値を、`echo` コマンドで、`sed s/-/./g` コマンドの標準入力に送るにはどうしますか？

3. `date +%Y-%m-%d` コマンドの出力を、`sed s/-/./g` にヒア文字列として与えるにはどうしますか？

4. `convert image.jpeg -resize 25% small/image.jpeg` コマンドは、`image.jpeg` を縮小したバージョンを作成して、`small` ディレクトリ内に同名のファイルとして保存します。`xargs` を使用して、`filelist.txt` にリストされているすべての画像に対して、このコマンドを実行するにはどうしますか？

発展演習

1. `/dev/sda1` パーティションのイメージを、`dd < /dev/sda1 > sda1.img` コマンドで作成する、簡単なバックアップスクリプトがあります。データの整合性を確認できるように、`sha1sum < sda1.img > sda1.sha1` コマンドで、SHA1ハッシュ値を格納したファイル `sda1.sha1` も作成しています。`tee` コマンドを使用して、この2つのコマンドを1つにまとめるにはどうしますか？

2. `tar` コマンドは、多くのファイルを、ディレクトリ構造を保持したまま1つのアーカイブファイルにまとめます。`-T` オプションを使うと、アーカイブに格納するファイルのパス名を収めたファイルを指定することができます。たとえば、`find /etc -type f | tar -cJ -f /srv/backup/etc.tar.xz -T -` は、`find` が見つけたファイルを収めた圧縮tarファイル `etc.tar.xz` を作成します。`-T -` オプションは、標準入力からパス名のリストを読み取ることを指示しています。リストから読み取られたパス名は、`tar` に対する引数の一つとして扱われるため、パス名に空白や特殊文字が含まれている場合には、予期しない動作を引き起こすことがあります（詳しくは `tar` のマニュアルを参照してください）。`find` と `tar` にどのようなオプションを指定すれば、空白や特殊文字を含むパス名を正しく取り扱えるようにできますか？

3. 新しいリモートシェルセッションを開かなくても、`ssh` では1つのコマンドをリモートマシンで実行することができます: `ssh user@storage "remote command"`。これを使えば、ローカルコマンドの標準出力を、リモートコマンドの標準入力に送ることもできます。`ssh` 経由で、ローカルファイル `etc.tar.gz` を `cat` して、リモートマシン（`op@storage`）の `/srv/backup/etc.tar.gz` に送るにはどうしますか？

まとめ

このレッスンでは、Linuxに備わっている伝統的なプロセス間通信について説明しました。コマンドパイプライン　　は2つのプロセス間に一方向の通信チャネルを作成し、コマンド置換　　はプロセスの出力をシェル変数に格納します。レッスンでは次の手順を説明しました。

- パイプ を使用して、プロセスの出力を別のプロセスの入力に流し込む方法。
- `tee` コマンドと `xargs` コマンドの働き。
- コマンド置換　　でプロセスの出力を取り込み、変数に格納し、あるいは、別のコマンドの引数として使用する方法。

以下のコマンドと手順を取り上げました:

- `|` を使用したコマンドパイプライン。
- バッククォート ないし `$()` によるコマンド置換。
- `tee`、`xargs`、`find` コマンド。

演習の解答

1. スクリプトを自動的に実行する場合は、実行された日時を保存しておくと便利です。`date +%Y-%m-%d` コマンドは、現在の日付を YYYY-MM-DD 形式で表示します。コマンド置換を使用して、このコマンドの出力を、`TODAY` というシェル変数に格納するにはどうしますか？

```
$ TODAY=`date +%Y-%m-%d`
```

または、

```
$ TODAY=$(date +%Y-%m-%d)
```

2. シェル変数 `TODAY` の値を、`echo` コマンドで、`sed s/-/./g` コマンドの標準入力に送るにはどうしますか？

```
$ echo $TODAY | sed s/-/./g
```

3. `date +%Y-%m-%d` コマンドの出力を、`sed s/-/./g` にヒア文字列として与えるにはどうしますか？

```
$ sed s/-/./g <<< `date +%Y-%m-%d`
```

または、

```
$ sed s/-/./g <<< $(date +%Y-%m-%d)
```

4. `convert image.jpeg -resize 25% small/image.jpeg` コマンドは、`image.jpeg` を縮小したバージョンを作成して、`small` ディレクトリ内に同名のファイルとして保存します。`xargs` を使用して、`filelist.txt` にリストされているすべての画像に対して、このコマンドを実行するにはどうしますか？

```
$ xargs -I IMG convert IMG -resize 25% small/IMG < filelist.txt
```

または、

```
$ cat filelist.txt | xargs -I IMG convert IMG -resize 25% small/IMG
```

発展演習の解答

1. `/dev/sda1` パーティションのイメージを、`dd < /dev/sda1 > sda1.img` コマンドで作成する、簡単なバックアップスクリプトがあります。データの整合性を確認できるように、`sha1sum < sda1.img > sda1.sha1` コマンドで、SHA1ハッシュ値を格納したファイル `sda1.sha1` も作成しています。`tee` コマンドを使用して、この2つのコマンドを1つにまとめるにはどうしますか？

   ```
   # dd < /dev/sda1 | tee sda1.img | sha1sum > sda1.sha1
   ```

2. `tar` コマンドは、多くのファイルを、ディレクトリ構造を保持したまま1つのアーカイブファイルにまとめます。`-T` オプションを使うと、アーカイブに格納するファイルのパス名を収めたファイルを指定することができます。たとえば、`find /etc -type f | tar -cJ -f /srv/backup/etc.tar.xz -T -` は、`find` が見つけたファイルを収めた圧縮tarファイル `etc.tar.xz` を作成します。`-T -` オプションは、標準入力からパス名のリストを読み取ることを指示しています。リストから読み取られたパス名は、`tar` に対する引数の一つとして扱われるため、パス名に空白や特殊文字が含まれている場合には、予期しない動作を引き起こすことがあります（詳しくは `tar` のマニュアルを参照してください）。`find` と `tar` にどのようなオプションを指定すれば、空白や特殊文字を含むパス名を正しく取り扱えるようにできますか？

   `-print0` アクションと `--null` オプション:

   ```
   $ find /etc -type f -print0 | tar -cJ -f /srv/backup/etc.tar.xz --null -T -
   ```

3. 新しいリモートシェルセッションを開かなくても、`ssh` では1つのコマンドをリモートマシンで実行することができます: `ssh user@storage "remote command"`。これを使えば、ローカルコマンドの標準出力を、リモートコマンドの標準入力に送ることもできます。`ssh` 経由で、ローカルファイル `etc.tar.gz` を `cat` して、リモートマシン（`op@storage`）の `/srv/backup/etc.tar.gz` に送るにはどうしますか？

   ```
   $ cat etc.tar.gz | ssh user@storage "cat > /srv/backup/etc.tar.gz"
   ```

   または

   ```
   $ ssh user@storage "cat > /srv/backup/etc.tar.gz" < etc.tar.gz
   ```

103.5 プロセスの作成、監視、終了

LPI目標への参照

LPIC-1 v5, Exam 101, Objective 103.5

総重量

4

主な知識分野

- フォアグラウンドとバックグラウンドでジョブを実行する。
- ログアウト後もプログラムの実行を継続するシグナルを送信する。
- アクティブなプロセスを監視する。
- 表示するプロセスの選択とソートする。
- シグナルをプロセスに送る。

用語とユーティリティ

- `&`
- `bg`
- `fg`
- `jobs`
- `kill`
- `nohup`
- `ps`
- `top`
- `free`
- `uptime`

- `pgrep`
- `pkill`
- `killall`
- `watch`
- `screen`
- `tmux`

103.5 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	103 GNUとUnixコマンド
Objective:	103.5 プロセスの作成、監視、強制終了
Lesson:	1 of 2

はじめに

コマンドを実行すると、1つ以上のプロセスが開始されます。熟達したシステム管理者は、プロセスを作成するだけでなく、プロセスを追跡し、必要に応じてさまざまなタイプのシグナルを送信できなければなりません。このレッスンでは、ジョブ制御とプロセス監視について説明します。

ジョブ制御

パイプを使用した場合など、ターミナルから実行する1つのコマンドラインで、複数のプログラム（プロセス）が実行されることがあります。シェルは、それらのプロセスをまとめて、1つの「ジョブ」として管理します。jobs コマンドを実行すると、バックグラウンドで実行中のジョブ（とそのステータス）を確認することができます。

```
$ jobs
```

上記の `jobs` コマンドの例では出力がありませんでした。現在アクティブなジョブが存在しないからです。このレッスンで見る最初のジョブを作成してみましょう。実行が完了するまでに時間がかかるコマンド（パラメータが `60` の `sleep` コマンド）を実行し、そのコマンドの実行中に`Ctrl`+`Z`を押します。

```
$ sleep 60
^Z
[1]+  Stopped                 sleep 60
```

コマンドの実行が停止され（より正確に言うとサスペンドされ）、コマンドプロンプトが再び使えるようになります。もう一度ジョブを探してみましょう。今度は　サスペンドされたジョブを見つけることができます。

```
$ jobs
[1]+  Stopped                 sleep 60
```

出力内容を説明します。

[1]
: この番号はジョブIDです。`fg`、`bg`、`kill`　ユーティリティで、パーセント記号（`%`）の後にこのジョブIDを指定することにより、ジョブのステータスを変更することができます（後で示します）。

+
: プラス記号は、現在のデフォルトのジョブ（最後にサスペンドされたジョブ、または、最後にバックグラウンドへと送られたジョブ）を示します。最後から一つ前のジョブにはマイナス記号（`-`）のフラグが付けられます。それ以前のジョブにはフラグが付けられません。

Stopped
: ジョブステータスの説明。

sleep 60
: ジョブのコマンドライン。

`-l`　オプションを使用すると、jobsコマンドは、ステータスの直前にプロセスID（PID）を追加で表示します。

```
$ jobs -l
[1]+  1114 Stopped                 sleep 60
```

jobsコマンドの他のオプションは次のとおりです。

-n
: ステータスが変化したプロセスのみを一覧表示します。ステータスには、`▯▯▯`、`▯▯`、`▯▯`

ないし `▯▯` などがあります。

`-p`
プロセスIDを一覧表示します。

`-r`
実行中のジョブのみを一覧表示します。

`-s`
停止された（サスペンドされた）ジョブのみを一覧表示します。

NOTE ジョブには ジョブID と プロセスID （PID）の2つのIDがあります。

ジョブ指定

`jobs` コマンドや `fg`、`bg`、`kill` などの他のユーティリティ（次のセクションで説明します）では、ジョブを指定するためにジョブ指示子（`jobspec`）を使います。通常は、`%` の後にジョブIDの数値を指定しますが、以下に述べるような指示方法もあります。

`%n`
ジョブIDが `n` のジョブ

```
$ jobs %1
[1]+  Stopped                 sleep 60
```

`%str`
コマンドラインが `str` で始まるジョブ

```
$ jobs %sl
[1]+  Stopped                 sleep 60
```

`%?str`
コマンドラインに `str` が含まれているジョブ

```
$ jobs %?le
[1]+  Stopped                 sleep 60
```

`%+` または `%%`
現在のジョブ（最後にバックグラウンドで開始されたジョブ、または、フォアグラウンドから一時停止されたジョブ）

```
$ jobs %+
[1]+  Stopped                 sleep 60
```

%-
現在のジョブから一つ前のジョブ

```
$ jobs %-
[1]+  Stopped                 sleep 60
```

上の例では、ジョブが1つしかないため、そのジョブが、現在のジョブ（%+）と一つ前のジョブ（%-）の両方に当てはまります。

ジョブのステータス：サスペンド、フォアグラウンド、バックグラウンド

ジョブがバックグラウンドにあるとき、またはサスペンドされているときには、次の3つの操作を実行できます。

1. fg でフォアグラウンドに移動させます。

   ```
   $ fg %1
   sleep 60
   ```

   fg　は指定されたジョブをフォアグラウンドに移動させ、そのジョブを現在のジョブにします。これで、終了するまで待つか、Ctrl+Zで再度停止させるか、Ctrl+Cで終了させることができるようになります。

2. bg でジョブをバックグラウンドに移動させます。

   ```
   $ bg %1
   [1]+ sleep 60 &
   ```

   バックグラウンドにあるジョブは　fg　でフォアグラウンドに戻すか、強制終了することができます（以下を参照）。ジョブがバックグラウンドに送られたことを意味するアンパサンド（&）に注意してください。実は、アンパサンドをコマンドの後に付ければ、直接バックグラウンドでプロセスを開始することもできます。

   ```
   $ sleep 100 &
   [2] 970
   ```

新しいジョブのジョブID（[2]）とともに、プロセスID（970）も表示されました。これで、2つのジョブがバックグラウンドで実行されている状態になります。

```
$ jobs
[1]-  Running                 sleep 60 &
[2]+  Running                 sleep 100 &
```

しばらくすると、1つ目のジョブが実行を終了します。

```
$ jobs
[1]-  Done                    sleep 60
[2]+  Running                 sleep 100 &
```

3. `kill` コマンドにより `SIGTERM` を送信してジョブを終了します。

```
$ kill %2
```

ジョブが終了したことを確認するには、`jobs` を再度実行します。

```
$ jobs
[2]+  Terminated              sleep 100
```

NOTE ジョブが指定されていない場合、`fg` と `bg` は、現在のデフォルトのジョブに作用します。ただし、`kill` にはジョブ指定が必要です。

切り離されたジョブ：`nohup`

ここまで見てきたジョブは、そのジョブを呼び出したユーザーのセッションに関連付けられていました。セッションが終了すると、セッションに関連付けられたジョブは失われます。ただし、セッションからジョブを切り離して、セッションが閉じられた後でもジョブを実行し続けることも可能です。これは、`nohup`（“no hangup”）コマンドで実現されます。構文は次のとおりです。

```
nohup COMMAND &
```

`&` はプロセスをバックグラウンドに送り、作業中の端末を解放するということを思い出してください。

バックグラウンドのジョブ `ping localhost` を現在のセッションから切り離してみましょ

う。

```
$ nohup ping localhost &
[1] 1251
$ nohup: ignoring input and appending output to 'nohup.out'
^C
```

コマンドを実行すると、ジョブID（[1]）とPID（1251）が表示され、次の行には、ファイル nohup.out に出力が追記される旨のメッセージが表示されます。このnohup.outファイルは、nohupでコマンドを実行したときに stdout と stderr の出力が保存されるデフォルトのファイルです。Ctrl+Cを押してコマンドプロンプトを解放し、exitと入力していったんログアウトしてセッション（ターミナル）を閉じ、（別のターミナルウィンドウを開くなどして）新しいセッション（ターミナル）をスタートしてください。tail -f を使用すると、先ほどのコマンドが実行され続けており、そのコマンドの出力がデフォルトファイル（nohup.out）に書き込まれていることを確認できます。

```
$ exit
logout
$ tail -f /home/carol/nohup.out
64 bytes from localhost (::1): icmp_seq=3 ttl=64 time=0.070 ms
64 bytes from localhost (::1): icmp_seq=4 ttl=64 time=0.068 ms
64 bytes from localhost (::1): icmp_seq=5 ttl=64 time=0.070 ms
^C
```

TIP デフォルトの nohup.out ファイルを使用する代わりに nohup ping localhost > /path/to/your/file & のように、出力ファイルを指定することもできます。

プロセスを強制終了する場合は、PIDを指定する必要があります。

```
# kill 1251
```

プロセス監視

プロセス（タスク）は、実行中のプログラムのインスタンスです。したがって、端末にコマンドを入力するたびに、新しいプロセスが作成されます。

watch コマンドは、プログラムを一定間隔（デフォルトは2秒）ごとに実行し、プログラムの出力が時によって変化していくのを 監視 することができます。たとえば、watch uptime というコマンドを実行すると、プロセスの増加に伴ってロードアベレージ（負荷の平均）がどのように変化するかを監視できます。

```
Every  2.0s: uptime          debian: Tue Aug 20 23:31:27 2019

 23:31:27 up 21 min,  1 user,  load average: 0.00, 0.00, 0.00
```

watchコマンドは割り込みを受けるまで実行されるため、Ctrl+Cで停止させます。停止させるまでは2行の出力が表示されます。1行目は watch からの出力で、コマンドが実行される頻度（Every 2.0s: uptime）、監視しているプログラム（uptime）、ホスト名と日付（debian: Tue Aug 20 23:31:27 2019）です。2行目はuptimeの出力で、現在時刻（23:31:27）、システムが稼働している時間（up 21 min）、アクティブユーザーの数（1 user）、左から過去1、5、15分間についてのロードアベレージ（実行中ないし待機中のプロセス数の平均）（load average: 0.00, 0.00, 0.00）です。

同様に、watch free で、プロセスの増減に伴って変化するメモリ使用量を確認できます。

```
Every  2.0s: free            debian: Tue Aug 20 23:43:37 2019

 23:43:37 up 24 min,  1 user,  load average: 0.00, 0.00, 0.00
              total        used        free      shared  buff/cache   available
Mem:       16274868      493984    14729396       35064     1051488    15462040
Swap:      16777212           0    16777212
```

watch の更新間隔を変更するには、次のように -n または --interval オプションと秒数を指定します。

```
$ watch -n 5 free
```

これで、free コマンドが5秒ごとに実行されます。

uptime、free、watch のオプションの詳細については、それぞれの項目のマニュアルページを参照してください。

NOTE uptime と free によって得られる情報は、より包括的なツールである後述の top や ps にも含まれています。

プロセスへのシグナルの送信： kill

すべてのプロセスには、一意のプロセス識別子すなわちPIDがあります。プロセスのPIDを調べる方法の一つは、pgrep コマンドに続けてプロセスの名前を入力することです。

```
$ pgrep sleep
```

```
1201
```

NOTE プロセス識別子（PID）は、`pidof` コマンドでも調べることができます。（例： `pidof sleep`）

`pgrep` と同様に、`pkill` コマンドは、プロセスの名前に基づいて、プロセスを強制終了します。

```
$ pkill sleep
[1]+  Terminated              sleep 60
```

同じ名前のプロセスの複数のインスタンスを強制終了するには、`killall` コマンドを使用します。

```
$ sleep 60 &
[1] 1246
$ sleep 70 &
[2] 1247
$ killall sleep
[1]-  Terminated              sleep 60
[2]+  Terminated              sleep 70
```

`pkill` と `killall` はどちらも `kill` とほとんど同じように機能します。これらのコマンドは、指定されたプロセスにシグナルを送信します。シグナルを指定しない場合、デフォルトの `SIGTERM` が送信されます。ただし、`kill` の引数は、ジョブIDかプロセスIDのいずれかで、`pkill` や `killall` のようにプロセスの名前を引数に取ることはできません。

シグナルは次のいずれかで指定できます。

- 名前:

  ```
  $ kill -SIGHUP 1247
  ```

- シグナル番号:

  ```
  $ kill -1 1247
  ```

- -sオプションとシグナル名ないしシグナル番号:

```
$ kill -s SIGHUP 1247
```

`kill` を `pkill` または `killall` と同じように機能させる（そして最初にPIDを見つけるためにコマンドを実行する手間を省く）には、コマンド置換を使用します。

```
$ kill -1 $(pgrep sleep)
```

すでにご存知のとおり、コマンド置換の別の構文は `kill -1 `pgrep sleep`` です。

TIP

`kill` のすべてのシグナルとその番号の完全なリストは、ターミナルに `kill -l` と入力すれば見ることができます。 `-SIGKILL`（`-9` または `-s KILL`）を使えば、他のシグナルでは失敗して終了させられない反抗的なプロセスを強制終了できます。

`top` と `ps`

プロセス監視には、`top` と `ps` の2つが欠かせないツールです。前者（top）はリアルタイムの状態を動的に出力するのに対し、後者（ps）はその時点の状態を静的に出力します。いずれにせよ、どちらもシステムのすべてのプロセスを包括的に把握するための優れたユーティリティです。

`top` を対話的に使う

`top` を呼び出すには、単に `top` と入力します。

```
$ top

top - 11:10:29 up  2:21,  1 user,  load average: 0,11, 0,20, 0,14
Tasks:  73 total,   1 running,  72 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0,0 us,  0,3 sy,  0,0 ni, 99,7 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem :  1020332 total,   909492 free,    38796 used,    72044 buff/cache
KiB Swap:  1046524 total,  1046524 free,        0 used.   873264 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
   436 carol     20   0   42696   3624   3060 R  0,7  0,4   0:00.30 top
     4 root      20   0       0      0      0 S  0,3  0,0   0:00.12 kworker/0:0
   399 root      20   0   95204   6748   5780 S  0,3  0,7   0:00.22 sshd
     1 root      20   0   56872   6596   5208 S  0,0  0,6   0:01.29 systemd
     2 root      20   0       0      0      0 S  0,0  0,0   0:00.00 kthreadd
     3 root      20   0       0      0      0 S  0,0  0,0   0:00.02 ksoftirqd/0
     5 root       0 -20       0      0      0 S  0,0  0,0   0:00.00 kworker/0:0H
     6 root      20   0       0      0      0 S  0,0  0,0   0:00.00 kworker/u2:0
```

```
    7 root      20   0       0      0      0 S  0,0  0,0   0:00.08 rcu_sched
    8 root      20   0       0      0      0 S  0,0  0,0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S  0,0  0,0   0:00.00 migration/0
   10 root       0 -20       0      0      0 S  0,0  0,0   0:00.00 lru-add-drain
   (...)
```

`top` は対話的に利用できます。デフォルトでは、出力は各プロセスで使用されているCPU時間（CPU使用率）の降順でソートされます。この表示順序は、`top` 内から次のキーを押すことで変更できます。

`M`
: メモリ の使用量でソートします。

`N`
: プロセスID でソートします。

`T`
: 実行 時間 でソートします。

`P`
: CPU使用率 でソートします。

TIP 降順と昇順を切り替えるには、`R` を押します。

`top` にコマンドを与えるキーは次のとおりです。

`?` または `h`
: ヘルプ。

`k`
: プロセスを強制終了します。強制終了するプロセスの `PID` と、送信するシグナル（デフォルトでは `SIGTERM 15`）の入力を要求されます。

`r`
: プロセスの優先度を変更します（`renice`）。`nice` 値の入力を要求されます。可能な値の範囲は-20～19ですが、負の値や現在の値よりも低い値に設定できるのはスーパーユーザー（`root`）のみです。

`u`
: 指定したユーザーのプロセスを一覧表示します（デフォルトでは、すべてのユーザーのプロセスが表示されます）。

`c`
: プログラムの絶対パスを表示し、ユーザー空間のプロセスとカーネル空間のプロセスを区別します（角括弧内のプロセスはカーネル空間のプロセス）。

`V`
: プロセスの親子関係をツリー状に表示します。

`t` と `m`
: それぞれ、CPU使用率とメモリ使用量をグラフ様に表示します。表示方法は4パターンあり、1)個別の値をテキスト表示、2)代表値とバーグラフ、3)代表値と棒グラフ、4)非表示、の順に切り替わります。

`W`
: 設定を `~/.toprc` に保存します。

> **TIP**
> 類似のコマンドとして、洗練されたユーザーインターフェイスの `htop` と、より網羅的な情報を出力する `atop` があります。いずれも、インストールされていない場合はパッケージマネージャからインストールできます。

`top` の出力の説明

`top` の出力は、サマリエリア と タスクエリア の2つの領域に分けられます。

`top` のサマリエリア

上部の5行が サマリエリア で、次の情報が表示されます。

- `top - 11:10:29 up 2:21, 1 user, load average: 0,11, 0,20, 0,14`
  - 現在時刻（24時間形式）: `11:20:29`
  - 稼働時間（システムが起動してからの時間）: `up 2:21`
  - ログイン中のユーザー数と過去1、5、15分間のCPUのロードアベレージ: `load average: 0,11, 0,20, 0,14`
- `Tasks: 73 total, 1 running, 72 sleeping, 0 stopped, 0 zombie`（プロセスに関する情報）
  - アクティブなプロセスの総数: `73 total`
  - 実行中のプロセス数: `1 running`
  - スリープ状態（入出力待ちなどで待機中）のプロセス数: `72 sleeping`
  - ジョブ制御により停止中のプロセス数: `0 stopped`
  - ゾンビ（実行を完了したが、親プロセスによりプロセステーブルから削除されるの

を待機しているプロセス）: `0 zombie`

- `%Cpu(s): 0,0 us, 0,3 sy, 0,0 ni, 99,7 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st`（使用されているCPU時間の割合）
  - ユーザープロセス: `0,0 us`
  - システム/カーネルプロセス: `0,3 sy`
  - nice 値が設定されたユーザープロセス: `0,0 ni`
  - 待機—アイドルCPU時間: `99,7 id`
  - 入出力を待機しているプロセス: `0,0 wa`
  - 周辺機器機からのハードウェア割り込み処理に費やされたCPU時間: `0,0 hi`
  - ソフトウェア割り込み処理に費やされたCPU時間: `0,0 si`
  - 仮想化環境において、現在の環境が利用できなかった（ハイパーバイザなどにstealされた）CPU時間: `0,0 st`
- `KiB Mem : 1020332 total, 909492 free, 38796 used, 72044 buff/cache`（キロバイト単位のメモリ情報）
  - メモリの合計量: `1020332 total`
  - 未使用メモリ: `909492 free`
  - 使用中メモリ: `38796 used`
  - ディスクキャッシュとして使用されているメモリ: `72044 buff/cache`

    `total` は、`free` と `used` と `buff/cache` の3つ値の合計であることに注意してください（この場合は約1 GB）。
- `KiB Swap: 1046524 total, 1046524 free, 0 used. 873264 avail Mem`（キロバイト単位のスワップ情報）
  - スワップ領域の合計量: `1046524 total`
  - 未使用のスワップ領域: `1046524 free`
  - 使用中のスワップ領域: `0 used`
  - スワップを発生させずに割り当てることができるメモリ量: `873264 avail Mem`

`top` のタスクエリア：フィールドと列

サマリエリア　の下には　タスクエリア　があります。これには、実行中のプロセスに関する情報を伝える一連の フィールド が含まれています。

PID
プロセスID。

USER
プロセスを実行したユーザー。

PR
プロセスの優先度。

NI
プロセスのnice値。値が低いほど優先度が高い。

VIRT
プロセスによって使用されているメモリの総量（スワップを含む）。

RES
プロセスによって使用されている物理メモリ。

SHR
他のプロセスとの共有メモリ。

S
プロセスのステータス。`S`（割り込み可能なスリープ - イベントが終了するのを待機しているプロセス）、`R`（実行可能 - 実行中または実行待ちのキュー内にあるプロセス）、`Z`（ゾンビ - 終了したがプロセステーブルから削除されていないプロセス）。

%CPU
プロセスのCPU使用率。

%MEM
プロセスの物理メモリ使用率。つまり、`RES`を利用可能な物理メモリ容量で割った値のパーセント表示。

TIME+
プロセスが消費したCPU時間の合計。

COMMAND
プロセスを開始したコマンド。

プロセスを静的に表示する： `ps`

前述のように、`ps` はプロセス状態のスナップショットを示します。現在のターミナル（tty）から起動されたすべてのプロセスを表示するには、`ps a` と入力します。

```
$ ps a
  PID TTY      STAT   TIME COMMAND
  386 tty1     Ss+    0:00 /sbin/agetty --noclear tty1 linux
  424 tty7     Ssl+   0:00 /usr/lib/xorg/Xorg :0 -seat seat0 (...)
  655 pts/0    Ss     0:00 -bash
 1186 pts/0    R+     0:00 ps a
 (...)
```

`ps` オプションのスタイルと出力の説明

psコマンドは、オプションの指定方法に3つのスタイルがあります。BSDスタイル、UNIXスタイル、GNUスタイルの3つです。指定したプロセスIDに関する情報を出力するときに、それぞれのスタイルでどうなるかを見てみましょう。

BSD
オプションにハイフンをつけません。

```
$ ps p 811
  PID TTY      STAT   TIME COMMAND
  811 pts/0    S      0:00 -su
```

UNIX
オプションにハイフンをつけます。

```
$ ps -p 811
  PID TTY          TIME CMD
  811 pts/0    00:00:00 bash
```

GNU
オプションに2個のハイフンをつけます。

```
$ ps --pid 811
  PID TTY          TIME CMD
  811 pts/0    00:00:00 bash
```

これら3つのpsコマンドは、いずれも、`PID` が `811` であるプロセス（この場合は `bash`）に関する情報を出力します。

同様に、`ps` では、ユーザーを指定してそのユーザーが開始したプロセスを検索することもできます。

- `ps U carol` (BSD)
- `ps -u carol` (UNIX)
- `ps --user carol` (GNU)

`carol` が開始したプロセスを調べてみましょう。

```
$ ps U carol
  PID TTY      STAT   TIME COMMAND
  811 pts/0    S      0:00 -su
  898 pts/0    R+     0:00 ps U carol
```

彼女は2つのプロセスを開始しました。`bash`（`-su`）と `ps`（`ps U carol`）です。`STAT` 列は、プロセスの状態を示します（以下を参照）。

psを最大限に活用するためには、さまざまなオプションを組み合わせます。たくさんのオプションがありますが、まず覚えておくと良いのが、BSDスタイルの `ps aux` です。このコマンドは、`top` コマンドと同様に、（現在のターミナルから実行したものだけではなく）すべてのプロセスを表示します。オプションの意味は次のとおりです。

a
: ターミナル（`tty`）に接続されているプロセスを表示します。

u
: ユーザーが読みやすいフォーマットで表示します。

x
: ターミナルに接続されていないプロセスを表示します。aオプションと併用することで、すべてのプロセスが表示されます。

```
$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1 204504  6780 ?        Ss   14:04   0:00 /sbin/init
root         2  0.0  0.0      0     0 ?        S    14:04   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    14:04   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   14:04   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S    14:04   0:00 [rcu_sched]
```

```
root         8  0.0  0.0      0     0 ?        S    14:04   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    14:04   0:00 [migration/0]
(...)
```

各列を説明しましょう。

USER
プロセスを実行したユーザー。

PID
プロセスID。

%CPU
CPUの使用率。

%MEM
物理メモリの使用率。

VSZ
プロセスが使用している仮想メモリ（KiB単位）。

RSS
プロセスが使用しているスワップされていない物理メモリ（KiB単位）。

TT
プロセスを制御するターミナル（tty）。

STAT
プロセスの状態を表すコード。S、R、Z の3つを、top の項で説明しました。他には、D （中断できないスリープ - 通常は入出力を待機しているプロセス）、T（停止 - 通常は制御シグナルによって停止されたプロセス）があります。さらに、< （優先度が高い - 他のプロセスよりもnice値が低い）、N（優先度が低い - 他のプロセスよりもnice値が高い）、+ （フォアグラウンドプロセスグループ）などの修飾子もあります。すべてを覚える必要はありませんから、必要に応じてマニュアルを参照してください。

STARTED
プロセスが開始された時刻。

TIME
累積CPU時間。

COMMAND
プロセスを開始したコマンド。

演習

1. `oneko` は、猫がマウスカーソルを追いかける面白いプログラムです。自分のデスクトップシステムにまだインストールされていない場合は、ディストリビューションのパッケージマネージャーを使用してインストールしてください。以下の演習ではこれを使ってジョブ制御を学びます。
   - どうやってプログラムを起動しますか？
   - マウスカーソルを動かして、猫がどのように追いかけるかを見てみましょう。次に、プロセスを停止します。どうやってプロセスを停止しますか？　出力内容はどうなりますか？
   - ジョブの数を確認してください。何を入力しますか？　出力内容はどうなりますか？
   - ジョブIDを指定して先ほど確認したジョブをバックグラウンドに送ります。出力内容はどうなりますか？　ジョブがバックグラウンドで実行されていることを確認するにはどうしますか？
   - 最後に、ジョブIDを指定してジョブを終了させます。何を入力しますか？
2. Apache HTTPD Webサーバー（`apache2`）によって生成されたすべてのプロセスのPIDを、2通りのコマンドで調べてください。
3. PIDを使用せずに、すべての `apache2` プロセスを終了させる2種類のコマンドはどうなりますか？
4. `apache2` のすべてのインスタンスを終了する必要があるのですが、それらのPIDを調べるのは面倒です。ワンライナー（1行のコマンド）として、killでSIGTERMを送信するにはどうしますか？

5. `top` を起動し、対話的に以下を実行してください。
   - プロセスをツリー状に表示してください。
   - ユーザー空間とカーネル空間を区別し、プロセスのフルパスを表示してください。
6. `ps` コマンドで、Apache HTTPD Webサーバー の所有者（`www-data`）が起動したすべてのプロセスを表示してください。
   - BSDスタイルを使用する場合。
   - UNIXスタイルを使用する場合。
   - GNUスタイルを使用する場合。

発展演習

1. デーモンを再起動するために、`SIGHUP` を送信することがあります。Apache HTTPD Webサーバー を例に取ってみましょう。`SIGHUP` を親プロセス（`init` ないし `systemd` が起動したプロセス）に送信すると、その子プロセスが強制終了されます。さらに、親プロセスは設定ファイルを再読込して、ログファイルを再度開き、新しい子プロセスを生成します。以下の課題に取り組んでみましょう。
   - Webサーバーを起動します。
   - 親プロセスのPIDを確認してください。
   - 親プロセスに `SIGHUP` を送信してApache HTTPD Webサーバーを再起動します。
   - 親プロセスは再起動していないこと、および新しい子プロセスが生成されていることを確認してください。
2. `ps` はある時点の状態を静的に表示をするコマンドですが、`watch` と組み合わせれば、時間による変化を動的に 追いかける ことができます。 Apache HTTPD webサーバー で新しい接続を監視してみましょう。以下で説明するタスクを実行する前に、https://httpd.apache.org/docs/current/mod/mpm_common.html[Apache MPM Common Directives] の `MaxConnectionsPerChild` ディレクティブの説明を読むことをお勧めします。
   - `apache2` の設定ファイルで、`MaxConnectionsPerChild` ディレクティブの値を `1` にセットします。 - Debian 系では `/etc/apache2/apache2.conf` ; CentOS 系は `/etc/httpd/conf/httpd.conf` が設定ファイルです。設定ファイルの変更を有効にするには、`apache2` を再起動することを忘れないでください。
   - `apache2` 接続を監視するために `watch`、`ps`、`grep` を使うコマンドを入力します。
   - ここで、Webブラウザーを開くか、`lynx` などのコマンドラインブラウザーを使用して、IPアドレスでWebサーバーに接続します。`watch` の出力結果から何がわかりますか？

3. このレッスンで学習したように、`top` はデフォルトでCPU使用率の降順に（最も大きな値が一番上になるように）タスクをソートします。この表示順序は、対話型キーである、`M`（メモリ使用量）、`N`（プロセスID）、`T`（実行時間）、`P`（CPU使用率）で変更できます。`-o` オプションを指定して `top` を起動すると、タスクリストをもっと細かく好みに合わせて並べ替えることもできます（詳細については、topの `man` ページを確認してください）。以下でやってみましょう。
   - `top` を起動して、タスクがメモリ使用量でソートされるようにしてください。
   - メモリ列を強調表示して、正しいコマンドを入力したことを確認してください。
4. `ps` にも、表示する列を指定するための `o` オプションがあります。このオプションを調べて、以下を実行してみてください。
   - ユーザー、メモリ使用率、CPU使用率、フルコマンド だけが表示されるように、`ps` を起動してください。
   - ユーザーとコマンド名だけが表示されるように、`ps` を起動してください。

まとめ

このレッスンでは、ジョブ　と　ジョブ制御　について学びました。覚えておくべき重要な概念と事柄は次のとおりです。

- ジョブにまとめられた一群のプロセスを、バックグラウンドに送ることができます。
- ジョブにはプロセスIDとは別に、ジョブIDが割り当てられます。
- ジョブを制御するには、ジョブ指示子（`jobspec`）が必要です。
- ジョブは、フォアグラウンドに移動させたり、バックグラウンドに送ったり、停止や終了（あるいは 強制終了）したりできます。
- ジョブは、端末やセッションから切り離すことができます。

プロセス と プロセス監視 の概念も説明しました。最も重要な概念は次のとおりです。

- プロセスは実行中のプログラムです。
- プロセスの状態を監視できます。
- さまざまなユーティリティを使用して、プロセスの　プロセスID　を確認し、プロセスを終了させるためのシグナルを送信できます。
- シグナルを指定するには、名前（`-SIGTERM`）、番号（`-15`）、オプション（`-s  SIGTERM`）などを使用します。
- プロセスの状態を監視するには、`top`　と　`ps`　が役立ちます。前者（top）の出力は動的であり、定期的に更新されます。一方、`ps`　はある時点でのプロセス状態を静的に表示します。

このレッスンでは以下のコマンドを使用しました:

`jobs`
　アクティブなジョブとそのステータスを表示します。

`sleep`
　指定された時間だけ処理を遅延します。

`fg`
　ジョブをフォアグラウンドに移動させます。

`bg`
　ジョブをバックグラウンドに移動させます。

`kill`
　ジョブにシグナルを送信します。

`nohup`
セッション/端末からジョブを切り離します。

`exit`
現在のシェルを終了します。

`tail`
ファイルの最新の行（最後の行）を表示します。

`watch`
コマンドを繰り返し実行します（デフォルトでは2秒ごと）。

`uptime`
システムの稼働時間、現在のユーザー数、システムのロードアベレージを表示します。

`free`
メモリ使用量を表示します。

`pgrep`
名前からプロセスIDを検索します。

`pidof`
名前からプロセスIDを検索します。

`pkill`
名前を指定してプロセスにシグナルを送信します。

`killall`
名前を指定してプロセスを強制終了させます。

`top`
プロセス状態を動的に表示します。

`ps`
現在のプロセス状態を取得して表示します。

演習の解答

1. `oneko` は、猫がマウスカーソルを追いかける面白いプログラムです。自分のデスクトップシステムにまだインストールされていない場合は、ディストリビューションのパッケージマネージャーを使用してインストールしてください。以下の演習ではこれを使ってジョブ制御を学びます。
   - どうやってプログラムを起動しますか？

     端末に `oneko` と入力します。

   - マウスカーソルを動かして、猫がどのように追いかけるかを見てみましょう。次に、プロセスを停止します。どうやってプロセスを停止しますか？　出力内容はどうなりますか？

     `Ctrl` + `z` を押します。

     ```
     [1]+  Stopped                 oneko
     ```

   - ジョブの数を確認してください。何を入力しますか？　出力内容はどうなりますか？

     ```
     $ jobs
     [1]+  Stopped                 oneko
     ```

   - ジョブIDを指定して先ほど確認したジョブをバックグラウンドに送ります。出力内容はどうなりますか？　ジョブがバックグラウンドで実行されていることを確認するにはどうしますか？

     ```
     $ bg %1
     [1]+ oneko &
     ```

     猫がまた動き出すことから、ジョブが実行されていることを確認できます。

   - 最後に、ジョブIDを指定してジョブを終了させます。何を入力しますか？

     ```
     $ kill %1
     ```

2. Apache HTTPD Webサーバー（`apache2`）によって生成されたすべてのプロセスのPIDを、2通りのコマンドで調べてください。

```
$ pgrep apache2
```

または

```
$ pidof apache2
```

3. PIDを使用せずに、すべての `apache2` プロセスを終了させる2種類のコマンドはどうなりますか？

```
$ pkill apache2
```

または

```
$ killall apache2
```

4. `apache2` のすべてのインスタンスを終了する必要があるのですが、それらのPIDを調べるのは面倒です。ワンライナー（1行のコマンド）として、killでSIGTERMを送信するにはどうしますか？

```
$ kill $(pgrep apache2)
$ kill `pgrep apache2`
```

または

```
$ kill $(pidof apache2)
$ kill `pidof apache2`
```

NOTE `SIGTERM`（15）はデフォルトのシグナルであるため、`kill` にオプションを渡す必要はありません。

5. `top` を起動し、対話的に以下を実行してください。
   - プロセスをツリー状に表示してください。

     `V` を押します。

   - ユーザー空間とカーネル空間を区別し、プロセスのフルパスを表示してください。

`c` を押します。

6. `ps` コマンドで、Apache HTTPD Webサーバー の所有者（`www-data`）が起動したすべてのプロセスを表示してください。
   - BSDスタイルを使用する場合。

     ```
     $ ps U www-data
     ```

   - UNIXスタイルを使用する場合。

     ```
     $ ps -u www-data
     ```

   - GNUスタイルを使用する場合。

     ```
     $ ps --user www-data
     ```

発展演習の解答

1. デーモンを再起動するために、`SIGHUP` を送信することがあります。Apache HTTPD Webサーバー を例に取ってみましょう。`SIGHUP` を親プロセス（`init` ないし `systemd` が起動したプロセス）に送信すると、その子プロセスが強制終了されます。さらに、親プロセスは設定ファイルを再読込して、ログファイルを再度開き、新しい子プロセスを生成します。以下の課題に取り組んでみましょう。
   - Webサーバーを起動します。

     ```
     $ sudo systemctl start apache2
     ```

   - 親プロセスのPIDを確認してください。

     ```
     $ ps aux | grep apache2
     ```

     親プロセスは、`root` ユーザーによって開始されたプロセスです（所有者がroot、親のPID（PPID）が1（initないしsystemd）になっています）。筆者が実行した環境では、PIDが `1653` のものでした。

   - 親プロセスに `SIGHUP` を送信してApache HTTPD Webサーバーを再起動します。

     ```
     $ kill -SIGHUP 1653
     ```

   - 親プロセスは再起動していないこと、および新しい子プロセスが生成されていることを確認してください。

     ```
     $ ps aux | grep apache2
     ```

     親の `apache2` プロセスといくつかの新しい子プロセスが表示されるはずです。

2. `ps` はある時点の状態を静的に表示をするコマンドですが、`watch` と組み合わせれば、時間による変化を動的に 追いかける ことができます。 Apache HTTPD webサーバー で新しい接続を監視してみましょう。以下で説明するタスクを実行する前に、https://httpd.apache.org/docs/current/mod/mpm_common.html[Apache MPM Common Directives] の `MaxConnectionsPerChild` ディレクティブの説明を読むことをお勧めします。
   - `apache2` の設定ファイルで、`MaxConnectionsPerChild` ディレクティブの値を `1` にセットします。 - Debian 系では `/etc/apache2/apache2.conf` ； CentOS 系は `/etc/httpd/conf/httpd.conf` が設定ファイルです。設定ファイルの変更を有効にす

るには、`apache2` を再起動することを忘れないでください。

設定ファイルに追加する行は `MaxConnectionsPerChild 1` です。`sudo systemctl restart apache2` を実行すると、Webサーバー（`apache2`）を再起動できます。

- `apache2` 接続を監視するために `watch`、`ps`、`grep` を使うコマンドを入力します。

```
$ watch 'ps aux | grep apache2'
```

または

```
$ watch "ps aux | grep apache2"
```

- ここで、Webブラウザーを開くか、`lynx` などのコマンドラインブラウザーを使用して、IPアドレスでWebサーバーに接続します。`watch` の出力結果から何がわかりますか？

`www-data` が所有する子プロセスの1つが消えます。

3. このレッスンで学習したように、`top` はデフォルトでCPU使用率の降順に（最も大きな値が一番上になるように）タスクをソートします。この表示順序は、対話型キーである、`M`（メモリ使用量）、`N`（プロセスID）、`T`（実行時間）、`P`（CPU使用率）で変更できます。`-o` オプションを指定して `top` を起動すると、タスクリストをもっと細かく好みに合わせて並べ替えることもできます（詳細については、topの `man` ページを確認してください）。以下でやってみましょう。
   - `top` を起動して、タスクがメモリ使用量でソートされるようにしてください。

```
$ top -o %MEM
```

   - メモリ列を強調表示して、正しいコマンドを入力したことを確認してください。

     `x` を押します。

4. `ps` にも、表示する列を指定するための `o` オプションがあります。このオプションを調べて、以下を実行してみてください。
   - ユーザー、メモリ使用率、CPU使用率、フルコマンド だけが表示されるように、`ps` を起動してください。

```
$ ps o user,%mem,%cpu,cmd
```

- ユーザーとコマンド名だけが表示されるように、`ps` を起動してください。

```
$ ps o user,comm
```

103.5 レッスン 2

Certificate:	LPIC-1
Version:	5.0
Topic:	103 GNUおよびUnixコマンド
Objective:	103.5 プロセスの作成、監視、強制終了
Lesson:	2 of 2

はじめに

前のレッスンではプロセス監視のためのツールとユーティリティを紹介しました。システム管理者はもう一歩先に進もうとすることもあるでしょう。このレッスンでは、ターミナルマルチプレクサという概念を紹介し、GNU Screen と tmux を取り上げます。今日では、素晴らしいターミナルエミュレータがいろいろありますが、ターミナルマルチプレクサは今なお生産的なシステム管理者のための強力な特徴を備えています。

ターミナルマルチプレクサの特徴

電子工学の分野では、マルチプレクサ（短く mux と表現することもあります）とは複数の入力を一つの出力に結びつける装置のことです。ターミナルマルチプレクサを使うと、必要に応じて複数の入力を切り替えることができます。ターミナルマルチプレクサである `screen` と `tmux` には、似た機能が備わっています。

- 起動するとセッションが開始されます。セッションはウィンドウを持ち、ウィンドウ内でプログラムを実行します。
- ウィンドウはリージョンやペインに分割できます。複数のプログラムを同時に実行するときにこれが役立ちます。
- コマンドプレフィックス や コマンドキー と呼ばれるキーの組み合わせでコマンドを実行できるので、管理がしやすいです。

- セッションはターミナルからデタッチ（切り離し）できます（プログラムがバックグラウンドに送られて実行され続けます）。そうすると、間違ってターミナルが閉じられたり、固まったり、リモート接続が失われたりしても、プログラムが最後まで実行されます。
- ソケット接続。
- コピーモード。
- カスタマイズ可能。

GNU Screen

初期Unix時代（1970年代から80年代）のコンピュータは、1台の中央コンピュータに複数の端末（訳注：英数字を表示するためのブラウン管とキーボードがセットになった装置で、VT100が代表的）が接続しているものでした。ウィンドウやタブなんてありません。1987年にGNU Screenが開発された背景にはこうした事情がありました。1台の物理的な端末で複数の画面をエミュレートしようとしたのです。

ウィンドウ

GNU Screenは、端末に `screen` と入力するだけで起動できます（訳注：起動しない場合はパッケージマネージャからインストールしてください）。次のようなウェルカムメッセージが表示されます。

```
GNU Screen version 4.05.00 (GNU) 10-Dec-16

Copyright (c) 2010 Juergen Weigert, Sadrul Habib Chowdhury
Copyright (c) 2008, 2009 Juergen Weigert, Michael Schroeder, Micah Cowan, Sadrul
Habib Chowdhury
Copyright (c) 1993-2002, 2003, 2005, 2006, 2007 Juergen Weigert, Michael Schroeder
Copyright (c) 1987 Oliver Laumann
(...)
```

スペースキーかエンターキーを押してメッセージを閉じると、コマンドプロンプトが表示されます。

```
$
```

何も起こらなかったように見えるかもしれませんが、実はもう `screen` が起動してセッションとウィンドウを管理しています。コマンドプレフィックスは`Ctrl`+`a`です。`Ctrl`+`a`-`w`と入力すると、ターミナルの最下部にウィンドウ一覧が表示されます。

```
0*$ bash
```

ウィンドウは1つしかありませんね。数字は0番から始まることに注意してください。別のウィンドウを立ち上げるには、`Ctrl`+`a`-`c`と入力してください。新しいプロンプトが表示されます。`ps`を実行してみましょう。

```
$ ps
  PID TTY          TIME CMD
  974 pts/2    00:00:00 bash
  981 pts/2    00:00:00 ps
```

もう一度`Ctrl`+`a`-`w`と入力してみましょう。

```
0-$ bash  1*$ bash
```

今度は2つのウィンドウがありますね（アスタリスクは現在のウィンドウを示しています）。どちらもBashから開始されたウィンドウなので、名前は両方ともbashです。現在のウィンドウでは `ps` を実行したので、現在のウィンドウを `ps` という名前にしてみましょう。`Ctrl`+`a`-`A`と入力し、プロンプトに新しいウィンドウの名前（`ps`）を入力してください。

```
Set window's title to: ps
```

次に、もう一つ新しいウィンドウを作りましょう。今回は最初から `yetanotherwindow` と名前を付けます。`screen` に `-t` オプションを指定します。

```
$ screen -t yetanotherwindow
```

ウィンドウを移動するにはいくつかやり方があります。

- `Ctrl`+`a`-`n`で次の（next）ウィンドウ、`Ctrl`+`a`-`p`で前の（previous）ウィンドウに移動できます。
- `Ctrl`+`a`-`0`-`9`でその番号（数字）のウィンドウに移動できます。
- `Ctrl`+`a`-`"`ですべてのウィンドウのリストを見ることができます。矢印キーの上と下でウィンドウを選んでエンターキーを押すと移動できます。

  ```
  Num Name                                                    Flags
  ```

```
 0 bash                                                          $
 1 ps                                                            $
 2 yetanotherwindow
```

ウィンドウに関しては、以下の事柄が重要です。

- ウィンドウはそれぞれ独立してプログラム（通常はシェル）を実行します。
- ウィンドウが見えない状態であってもプログラムは実行され続けます（後述のように、セッションがデタッチされたときも同様にプログラムが実行され続けます）。

ウィンドウを開始する際に実行したプログラム（bash）が終了するとウィンドウが削除されます。ウィンドウがすべて削除されると `screen` 自体が終了します。`Ctrl`+`a`-`k`と入力しても現在のウィンドウが削除されます。この場合、（シェルがまだ実行されていますから）確認を求められます。

```
Really kill this window [y/n]

Window 0 (bash) killed.
```

リージョン

`screen` では1つのターミナルを複数のリージョンに分けることができます。それぞれのリージョンがウィンドウを持ちます。リージョンは、水平方向（`Ctrl`+`a`-`S`）にも、垂直方向（`Ctrl`+`a`-`|` 縦棒）にも分割できます。

新しいリージョンには `--` とだけ表示されます。

```
 1 ps                                                            --
```

新しいリージョンに移動するには、`Ctrl`+`a`-`Tab`と入力してください。先ほど紹介したやり方で新しいリージョンにウィンドウを追加できます。例えば、`Ctrl`+`a`-`2`と入力すると、 `--` が `2yetanotherwindow` に変化します。

```
$ ps                                                            $
  PID TTY          TIME CMD
 1020 pts/2    00:00:00 bash
 1033 pts/2    00:00:00 ps
$ screen -t yetanotherwindow
```

```
   1 ps                                                                2
yetanotherwindow
```

リージョンに関して重要な事柄は次のとおりです。

- `Ctrl`+`a`-`Tab`でリージョンを移動します。
- `Ctrl`+`a`-`Q`で現在のリージョン以外のすべてのリージョンを終了します。
- `Ctrl`+`a`-`X`で現在のリージョンを終了します。
- リージョンを終了してもそのリージョンに関連付けられたウィンドウは終了しません。

セッション

ここまでにウィンドウとリージョンを説明しましたが、いずれも一つのセッションに属していました。次はセッションを説明しましょう。すべてのセッションを確認するには、`screen -list` か `screen -ls` と入力します。

```
$ screen -list
There is a screen on:
        1037.pts-0.debian       (08/24/19 13:53:35)     (Attached)
1 Socket in /run/screen/S-carol.
```

ここまでのところではセッションが一つしかありませんね。

PID

`1037`

名前

`pts-0.debian`　（ターミナル（この例ではセッション名を指定していないので、使用している 疑似端末 名）とホスト名を示しています）。

ステータス

`Attached`

わかりやすい名前を付けて新しいセッションを開始してみましょう。

```
$ screen -S "second session"
```

端末の表示がクリアされて新しいプロンプトが始まります。セッションをもう一度確認して

みましょう。（訳注：自分がどのセッションに居るのかを調べるには、`echo $STY` コマンドを使います。）

```
$ screen -ls
There are screens on:
        1090.second session     (08/24/19 14:38:35)     (Attached)
        1037.pts-0.debian       (08/24/19 13:53:36)     (Attached)
2 Sockets in /run/screen/S-carol.
```

セッションを終了するには、そのセッションのすべてのウィンドウを終了させるか、`screen -S ▯▯▯▯▯▯PID -X quit` コマンドを実行します。セッションのPIDのかわりにセッション名でも構いません。1つ目のセッションを閉じてみましょう。

```
$ screen -S 1037 -X quit
```

`screen` の外側のターミナルプロンプトに戻されます。それでも先ほど作成した2つ目のセッションは生きています。

```
$ screen -ls
There is a screen on:
    1090.second session (08/24/19 14:38:35) (Detached)
1 Socket in /run/screen/S-carol.
```

親セッションを終了したので、ラベルが `Detached` に変わりました。

セッションのデタッチ

端末からセッションをデタッチしたい場面があります。

- 職場のコンピュータに作業させておき、後で家からリモート接続する場面。
- 他のユーザーとセッションを共有する場面。

`Ctrl`+`a`-`d`でセッションをデタッチします。`screen` の外側のターミナルプロンプトに戻されます。（訳注：デタッチした後にシステムからログアウトしても、セッションは動き続けます。同じ端末、あるいは異なる端末からログインして、既存のセッションにアタッチできます）。

```
[detached from 1090.second session]
$
```

`screen -r ▯▯▯▯▯▯▯` PID コマンドでセッションをアタッチできます。`▯▯▯▯▯▯▯` PID のかわりに `▯▯▯▯▯▯▯` でも構いません。セッションが1つしかない場合は、`screen -r` だけでもアタッチできます。

```
$ screen -r
```

1つ目のセッションを終了して、2つ目のセッションしか残っていないので、このコマンドだけでアタッチできます。

```
$ screen -ls
There is a screen on:
        1090.second session     (08/24/19 14:38:35)     (Attached)
1 Socket in /run/screen/S-carol.
```

セッションのアタッチに関しては、以下のオプションがあります。

`-d -r`
: セッションを（必要があれば）デタッチしてから、指定のセッションに再アタッチします。

`-d -R`
: 必要があればセッションを作成してから、セッションにアタッチします（`-d -r` と同様）。

`-d -RR`
: `-d -R` と同じ。ただし、複数のセッションにアタッチできる場合には、最初のセッションにアタッチします。

`-D -r`
: セッションが稼働中であれば、そこからデタッチとログアウトさせてからアタッチします。

`-D -R`
: セッションが稼働中の場合は `-D -r` と同じ。実行中でない場合は、セッションを作成してユーザーに通知します。

`-D -RR`
: `-D -R` と同じ（既存のセッションにとにかくアタッチしたいときなどに便利です）。

`-d -m`
: デタッチモード で `screen` を起動します。新しいセッションを作成しますがアタッチ

しません。シェルの起動スクリプトなどで使用すると便利です。

`-D -m`

`-d -m` と同じですが、新しいプロセスをフォークしません。セッションが終了したらこのコマンドを呼び出したプロセスも終了します。

他のオプションについては `screen` のマニュアルを参照してください。

コピーアンドペースト：スクロールバックモード

GNU Screenは、コピーモード（スクロールバックモード）を備えています。矢印キーでカーソルを動かして現在のウィンドウのコマンド履歴を遡れます。テキストのコピーもできます。手順は次のとおりです。

1. `Ctrl`+`a`-`[`でコピーモード（スクロールバックモード）に入ります。
2. 矢印キーでコピーしたいテキストの先頭まで移動します。
3. スペースキーでコピーしたいテキストの先頭にマークをつけます。
4. 矢印キーでコピーしたいテキストの末尾に移動します。
5. スペースキーでコピーしたいテキストの末尾にマークをつけます。
6. ペーストしたいウィンドウに移動し、`Ctrl`+`a`-`]`でペーストします。

screenのカスタマイズ

`screen` のシステム全体の設定ファイルは `/etc/screenrc` です。ユーザーレベルの設定ファイルは `~/.screenrc` です。設定ファイルは、4つの大きなセクションに分かれています。

`SCREEN SETTINGS`

一般的な設定です。`defscrollback 1024` のように、ディレクティブ の後に半角スペースを入れて 値 を設定します。

`SCREEN KEYBINDINGS`

キーバインドを再定義できます。ここで再定義すると、いつものように端末が使えなくなるかもしれないので注意してください。`bind` の後に半角スペースを入れてコマンドプレフィックスの後に入力するキーを書き、さらに半角スペースを入れてコマンドを書きます。`bind l kill` といった具合です。このように再定義すると、ウィンドウを削除するには、デフォルトの`Ctrl`+`a`-`k`ではなく、`Ctrl`+`a`-`l`と入力することになります。

`screen` のキーバインド一覧は、`Ctrl`+`a`-`?`と入力するかマニュアルで確認してください。

> **TIP** コマンドプレフィックス自体を変更することもできます。`Ctrl`+`a`か

らCtrl+bに変更するには、escape ^Bbという行を追記してください。

TERMINAL SETTINGS

このセクションでは、端末のウィンドウサイズやバッファなどを設定します。例えば、ノンブロッキングモードを有効にするには、defnonblock 5と設定します。

STARTUP SCREENS

screen の起動時に実行されるコマンドを指定します。screen -t top top と設定すると、topという名前でtopを実行するウィンドウが開きます。

tmux

tmux は2007年にリリースされました。screen とよく似ていますが、いくつか違いがあります。

- クライアントサーバーモデル。サーバーが複数のセッションを持ち、それぞれのセッションが複数のウィンドウを持ち、ウィンドウは複数のペインに分割できます。
- セッション、ウィンドウをメニューで対話的に選べます。
- 同じウィンドウを複数のセッションに紐付けできます。
- vim と Emacs の両方のキーレイアウトを利用できます。（ここではデフォルトのEmacs モードのキー操作を説明します）。
- UTF-8と256色の端末をサポートします。

ウィンドウ

ターミナルで tmux と入力するだけで起動できます。起動するとシェルのプロンプトとウィンドウの下部にステータスバーが表示されます。

```
[0] 0:bash*                                                    "debian" 18:53
27-Aug-19
```

ステータスバーには、ホスト名、日時に加えて、以下の情報が表示されます。

セッション名

[0]

ウィンドウ番号

0:

ウィンドウ名

bash*。デフォルトではウィンドウ内で実行されているプログラムの名前です。screen

とは違い、tmux は現在実行しているプログラムを自動的にウィンドウ名に反映します。アスタリスクは現在見えているウィンドウです。

セッション名とウィンドウ名を指定して tmux を起動することもできます。

```
$ tmux new -s "LPI" -n "Window zero"
```

指定したセッション名とウィンドウ名に応じてステータスバーの表示が変わります。

```
[LPI] 0:Window zero*                                          "debian" 19:01
27-Aug-19
```

tmux のコマンドプレフィックスはCtrl+bです。Ctrl+b-cと入力すると新しいウィンドウが作成されます。新しいプロンプトが表示され、ステータスバーのウィンドウ名の部分が変わります。

```
[LPI] 0:Window zero- 1:bash*                                  "debian" 19:02
27-Aug-19
```

シェルはbashなので、新しいウィンドウにはbashという名前がデフォルトで付けられます。top を起動して、ウィンドウ名が top に変わることを確かめてみます。

```
[LPI] 0:Window zero- 1:top*                                   "debian" 19:03
27-Aug-19
```

Ctrl+b-,（カンマ）でウィンドウ名を変更することもできます。新しいウィンドウ名の入力を求められるので、名前を入力してエンターキーを押します。

```
(rename-window) Window one
```

Ctrl+b-wを押すとすべてのウィンドウが表示されます。矢印キーの上と下でウィンドウを選択してエンターキーを押すとそのウィンドウに移動できます。

```
(0)  0: Window zero- "debian"
(1)  1: Window one* "debian"
```

screen と同じように、以下のやり方でウィンドウを移動することもできます。

Ctrl+**b**-**n**
次の（next）ウィンドウに移動します。

Ctrl+**b**-**p**
前の（previous）ウィンドウに移動します。

Ctrl+**b**-
その番号（数字）のウィンドウに移動します。

Ctrl+b-&でウィンドウを削除できます。確認を求められます。

```
kill-window Window one? (y/n)
```

他には次のようなコマンドがあります。

Ctrl+**b**-**f**
名前でウィンドウを検索します。

Ctrl+**b**-**.**
ウィンドウ番号を変更します。

その他のコマンドについてはマニュアルか、Ctrl+b-?で表示されるヘルプを参照してください。

ペイン

tmuxでもscreenと同じようにウィンドウを分割できます。ウィンドウを分割した部分を、screenではリージョンと呼びましたが、tmuxではペインと呼びます。ペインはリージョンとは異なりウィンドウと連動している疑似端末です。ペインを終了するとその中で実行されているプログラムも終了します。

Ctrl+b-"で水平方向にウィンドウを分割します。

```
Tasks:  93 total,   1 running,  92 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  4050960 total,  3730920 free,   114880 used,   205160 buff/cache
KiB Swap:  4192252 total,  4192252 free,        0 used.  3716004 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 1340 carol     20   0   44876   3400   2800 R  0.3  0.1   0:00.24 top
    1 root      20   0  139088   6988   5264 S  0.0  0.2   0:00.50 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
```

```
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.04 ksoftirqd/0
    4 root      20   0       0      0      0 S  0.0  0.0   0:01.62 kworker/0:0
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S  0.0  0.0   0:00.06 rcu_sched
    8 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 migration/0
   10 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 lru-add-drain
   11 root      rt   0       0      0      0 S  0.0  0.0   0:00.01 watchdog/0
   12 root      20   0       0      0      0 S  0.0  0.0   0:00.00 cpuhp/0
$
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯
$









[LPI] 0:Window zero- 1:Window one*                                      "debian"
19:05 27-Aug-19
```

Ctrl⌧+⌧b-%で垂直方向にウィンドウを分割します。

```
                                                                                ▯$
    1 root      20   0  139088   6988   5264 S  0.0  0.2   0:00.50 systemd      ▯
                                                                                ▯
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd     ▯
                                                                                ▯
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.04 ksoftirqd/0  ▯
    4 root      20   0       0      0      0 S  0.0  0.0   0:01.62 kworker/0:0  ▯
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H▯
    7 root      20   0       0      0      0 S  0.0  0.0   0:00.06 rcu_sched    ▯
    8 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_bh       ▯
    9 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 migration/0  ▯
                                                                                ▯
   10 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 lru-add-drai▯
n                                                                               ▯
```

```
   11 root      rt   0       0      0      0 S  0.0  0.0   0:00.01 watchdog/0  ▯
                                                                              ▯
   12 root      20   0       0      0      0 S  0.0  0.0   0:00.00 cpuhp/0     ▯
                                                                              ▯
$                                                                             ▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯
$









[LPI] 0:Window zero- 1:Window one*                                      "debian"
19:05 27-Aug-19
```

Ctrl⊠+⊠b-xで現在のペインを（疑似端末やそこで実行されているプログラムとともに）破棄します。ステータスバーで確認を求められます。

```
kill-pane 1? (y/n)
```

よく使うペインの操作コマンドを以下に示します。

Ctrl⊠+⊠**b**-**↑**,**↓**,**←**,**→**
　ペインを移動します。

Ctrl⊠+⊠**b**-**;**
　直前にアクティブだったペインに移動します。

Ctrl⊠+⊠**b**-**Ctrl**⊠+⊠▯▯▯▯
　ペインのサイズを1行分変更します。

`Ctrl`+`b`-`Alt`+`↑↓←→`
ペインのサイズを5行分変更します。

`Ctrl`+`b`-`{`
現在のペインと１つ前のペインを入れ替えます。

`Ctrl`+`b`-`}`
現在のペインと１つ後のペインを入れ替えます。

`Ctrl`+`b`-`z`
パネルのズームインとズームアウトを切り替えます。

`Ctrl`+`b`-`t`
ペイン内に時計を表示します（`q`で非表示にできます）。

`Ctrl`+`b`-`!`
ペインをウィンドウにします。

その他のコマンドについてはマニュアルか、`Ctrl`+`b`-`?` で表示されるヘルプを参照してください。

セッション

`Ctrl`+`b`-`s`で`tmux`のセッションのリストを表示できます。

```
(0) + LPI: 2 windows (attached)
```

`tmux ls`コマンドでもセッションのリストを表示できます。

```
$ tmux ls
LPI: 2 windows (created Tue Aug 27 19:01:49 2019) [158x39] (attached)
```

セッションは１つしかなく（LPI）、そのセッションが2つのウィンドウを持っています。新しいセッションを作ってみましょう。`Ctrl`+`b`の後に `:new` と入力してエンターキーを押してください。新しいセッションに移行します。そのことはステータスバーで示されています。

```
[2] 0:bash*                                                    "debian" 19:15
27-Aug-19
```

デフォルトでは 2 というセッション名になります。Ctrl+b-$でセッション名を変更できます。入力を求められたら、新しいセッション名を入力して、エンターキーを押します。

```
(rename-session) Second Session
```

Ctrl+b-sでセッションを切り替えられます（矢印キーで選んでエンターキーを押してください）。

```
(0) + LPI: 2 windows
(1) + Second Session: 1 windows (attached)
```

セッションを終了するには、tmux kill-session -t ▯▯▯▯▯▯▯ というコマンドを使います。このコマンドで現在のセッションを終了すると、tmux のセッションから tmux を起動した元のターミナルのセッションに戻されます。

```
$ tmux kill-session -t "Second Session"
[exited]
$
```

セッションのデタッチ

Second Session を終了すると tmux の外側に戻されました。しかし、アクティブなセッションが残っています。以下のコマンドを実行すると、確かにセッションがありますね。

```
$ tmux ls
LPI: 2 windows (created Tue Aug 27 19:01:49 2019) [158x39]
```

このセッションはターミナルからデタッチされています。tmux attach -t ▯▯▯▯▯▯▯ でアタッチできます（attach のかわりに at や a という省略形も使えます）。セッションが１つしかなければ、名前の指定は不要です。

```
$ tmux a
```

セッションに戻りました。Ctrl+b-dでデタッチします。

```
[detached (from session LPI)]
$
```

TIP

tmux では、1つのセッションに複数の端末（クライアント）からアタッチできます（一種の画面共有）。アタッチしたセッションを確実に操作するには、セッションにアタッチする際にまずそのセッションから他の端末をデタッチすると良いでしょう。tmux attach -d -t □□□□□□□ のように -d オプションを指定すると、指定したセッションから他の端末をデタッチした後でアタッチできます。

セッションのアタッチとデタッチに関しては次のコマンドがあります。

Ctrl+**b**-**D**
　デタッチするクライアントを選択します。

Ctrl+**b**-**r**
　クライアントの端末を再描画します。

その他のコマンドについてはマニュアルか、Ctrl+b-? で表示されるヘルプを参照してください。

コピーアンドペースト：スクロールバックモード

tmux にもコピーモードがあり、screen と基本的には同じです（tmux のコマンドプレフィックスはCtrl+b で screen のコマンドプレフィックスCtrl+a とは異なることに留意してください）。Ctrl+space で開始点のマーク、Alt+w で選択したテキストをコピーするという点が、screen とは異なります。（viモードではキー操作が異なります。）

tmuxのカスタマイズ

tmux の設定ファイルは /etc/tmux.conf と ~/.tmux.conf です。tmux の起動時にこれらのファイルがあれば読み込まれます。-f オプションを使うとこれらのファイルとは別の設定ファイルを読み込めます。/usr/share/doc/tmux/example_tmux.conf に設定ファイルの例があります。カスタマイズ性が高いです。例えば以下のような設定ができます。

- コマンドプレフィックスのキーを変更します。

```
# Change the prefix key to C-a
set -g prefix C-a
unbind C-b
bind C-a send-prefix
```

- ウィンドウ番号が10以上のウィンドウに切り替えるためのキーバインドを追加します。

```
# Some extra key bindings to select higher numbered windows
bind F1 selectw -t:10
```

```
bind F2 selectw -t:11
bind F3 selectw -t:12
```

キーバインドのリストは`Ctrl`+`b`-`?`で確認できます（`q` を押すと終了します）。マニュアルにも載っています。

演習

1. 次の特徴は、`screen` か `tmux` のどちらに当てはまりますか。両方に当てはまるものもあります。

特徴	GNU Screen	tmux
デフォルトのコマンドプレフィックスは`Ctrl`+`a`		
クライアントサーバーモデル		
ペインは疑似端末		
リージョンを終了してもウィンドウは終了しない		
セッションがウィンドウを持つ		
セッションをデタッチできる		

2. GNU Screenをインストールして（パッケージ名は `screen` です）、次の課題に取り組んでください。
   - `screen` を開始してください。
   - `top` を起動してください。
   - コマンドプレフィックスで新しいウィンドウを開き、`vi` で `/etc/screenrc` を開いてください。
   - ターミナルの下部にウィンドウのリストを表示させてください。
   - 現在のウィンドウ名を `vi` に変更してください。
   - 残っているもう1つウィンドウの名前を `top` に変更してください。まずウィンドウのリストを表示し、名前を変更するウィンドウを選択します。

- もう一度ターミナルの下部にウィンドウ名を表示して、ウィンドウ名が変更できていることを確かめてください。

- セッションをデタッチして、`ssh` という名前の新しいセッションを作成してください。

- `ssh` という名前のセッションもデタッチして、セッションのリストを表示してください。

- PIDを指定して最初のセッションをアタッチしてください。

- `top` を表示しているウィンドウに戻ってくるはずです。ウィンドウを水平方向に分割して、新しい空白のリージョンに移動してください。

- ウィンドウのリストを表示し、`vi` という名前のウィンドウを選んで、先ほど新しく作った空白のリージョンに表示されるようにしてください。

- 現在のリージョンを垂直方向に分割し、新しく作られた空白のリージョンに移動してください。そして、新しいウィンドウをそのリージョンで作ってください。

- 現在のリージョン以外のリージョンをすべて終了してください（リージョンを終了してもウィンドウは終了しません）。次に、すべてのウィンドウを終了してください。そうするとようやくセッションが終了します。

- 最後にセッションのリストを確認します。PIDを指定して `ssh` という名前のセッションを終了し、セッションがもう残っていないことを確認してください。

3. `tmux` をインストールして（パッケージ名は `tmux` です）、次の課題に取り組んでください。
   - `tmux` を開始してください。
   - `top` を起動してください（数秒でステータスバーのウィンドウの名前が `top` に変わることに注目してください）。
   - コマンドプレフィックスで新しいウィンドウを開き、 `nano` で `~/.tmux.conf` を作成してください。
   - ウィンドウを水平方向に分割し、新しく作成したペインのサイズを少し小さくしてください。
   - 現在のウィンドウの名前を `text editing` に変更し、 `tmux` のセッションのリストを表示してください。
   - 同じコマンドを2回入力して、1回目で `top` を実行しているウィンドウに移動し、2回目で現在のウィンドウに戻ってください。
   - 現在のセッションをデタッチして、セッション名が `ssh` 、ウィンドウ名が `ssh window` の新しいセッションを作成してください。
   - `ssh` という名前のセッションもデタッチして、 `tmux` のセッションのリストを表示してください。

NOTE

ここからはリモートマシンからローカルマシンに `ssh` 接続することが必要になります（仮想マシンを使ったリモート接続でも大丈夫です）。ローカルマシンで `openssh-server` をインストールして起

> 動させ、リモートマシンで `openssh-client` をインストールしてください。

- リモートマシンからローカルマシンに `ssh` 接続してください。接続を確立できたら `tmux` のセッションを確認してください。

- リモートマシンから `ssh` という名前のセッションにアタッチしてください。

- ローカルマシンに戻り、`ssh` という名前のセッションをアタッチします。リモートホストとの接続を終了してからアタッチするコマンドを入力してください。

- すべてのセッションを表示し、最初のセッション（`[0]`）に移動します。移動後に `ssh` という名前のセッションを終了してください。

- 最後に、現在のセッションをデタッチして、そのセッションを終了します。

発展演習

1. `screen` でも `tmux` でもコマンドプレフィックスに続けて`:`（コロン）を入力するとコマンドラインモードになります（`tmux` での簡単な例はすでに見ました）。コマンドラインモードで次の課題に取り組んでください。
   - `screen` でコピーモードに入ってください。
   - `tmux` で現在のウィンドウの名前を変更してください。
   - `screen` でですべてのウィンドウを閉じてセッションを終了してください。
   - `tmux` でペインを分割してください。
   - `tmux` で現在のウィンドウを終了してください。
2. `screen` でコピーモードに入ると、矢印キーや`PgUP`、`PgDown`キーで現在のウィンドウを遡れますが、`vi` 風のフルスクリーンエディタを使うこともできます。次の課題に取り組んでください。
   - `screen` の端末で `supercalifragilisticexpialidocious` と出力してください。
   - カーソルの真上にある5つの連続する文字をコピーしてください。
   - 選択した部分（筆者の環境では `stice` の5文字になりました）をコマンドプロンプトにペーストしてください。
3. `tmux` のセッション（`our_session`）を別のユーザーと共有します。相手のユーザーも読み書きできるパーミッションでソケット（`/tmp/our_socket`）は作ってあります。相手のユーザーが `tmux -S /tmp/our_socket a -t our_session` でセッションをアタッチできる

ようにするには、あと２つしなければならないことがあります。それは何でしょう？

まとめ

このレッスンでは、ターミナルマルチプレクサという一般的な概念と、GNU Screen とtmuxという具体的なツールについて学びました。次の事柄が重要です。

- コマンドプレフィックス（`screen` では`Ctrl`+`a`、`tmux` では`Ctrl`+`b`）
- セッション、ウィンドウ、ウィンドウの分割（リージョンまたはペイン）
- コピーモード
- セッションのデタッチ

このレッスンでは以下のコマンドを使用しました:

`screen`
`screen` のセッションを開始します。

`tmux`
`tmux` のセッションを開始します。

演習の解答

1. 次の特徴は、`screen` か `tmux` のどちらに当てはまりますか。両方に当てはまるものもあります。

特徴	GNU Screen	tmux
デフォルトのコマンドプレフィックスは`Ctrl`+`a`	x	
クライアントサーバーモデル		x
ペインは疑似端末		x
リージョンを終了してもウィンドウは終了しない	x	
セッションがウィンドウを持つ	x	x
セッションをデタッチできる	x	x

2. GNU Screenをインストールして（パッケージ名は `screen` です）、次の課題に取り組んでください。
   - `screen` を開始してください。

     ```
     screen
     ```

   - `top` を起動してください。

     ```
     top
     ```

   - コマンドプレフィックスで新しいウィンドウを開き、`vi` で `/etc/screenrc` を開いてください。

     `Ctrl`+`a`-`c`

     ```
     sudo vi /etc/screenrc
     ```

   - ターミナルの下部にウィンドウのリストを表示させてください。

     `Ctrl`+`a`-`w`

   - 現在のウィンドウ名を `vi` に変更してください。

     `Ctrl`+`a`-`A`を押してから、 `vi` と入力してエンターキーを押します。

- 残っているもう1つウィンドウの名前を `top` に変更してください。まずウィンドウのリストを表示し、名前を変更するウィンドウを選択します。

  `Ctrl`+`a`-`"`を押し、矢印キーで `0 bash` と表示されているものを選んでエンターキーを押します。それから`Ctrl`+`a`-`A`を押し、 `top` と入力してエンターキーを押します。

- もう一度ターミナルの下部にウィンドウ名を表示して、ウィンドウ名が変更できていることを確かめてください。

  `Ctrl`+`a`-`w`

- セッションをデタッチして、`ssh` という名前の新しいセッションを作成してください。

  `Ctrl`+`a`-`d`

  `screen -S "ssh"`

- `ssh` という名前のセッションもデタッチして、セッションのリストを表示してください。

  `Ctrl`+`a`-`d`

  `screen -list` または `screen -ls`

- PIDを指定して最初のセッションをアタッチしてください。

  `screen -r` PID

- `top` を表示しているウィンドウに戻ってくるはずです。ウィンドウを水平方向に分割して、新しい空白のリージョンに移動してください。

  `Ctrl`+`a`-`S`

  `Ctrl`+`a`-`Tab`

- ウィンドウのリストを表示し、`vi` という名前のウィンドウを選んで、先ほど新しく作った空白のリージョンに表示されるようにしてください。

  `Ctrl`+`a`-`"`ですべてのウィンドウを表示し、 `vi` を選んでエンターキーを押します。

- 現在のリージョンを垂直方向に分割し、新しく作られた空白のリージョンに移動してください。そして、新しいウィンドウをそのリージョンで作ってください。

`Ctrl`+`a`-`|`

`Ctrl`+`a`-`Tab`

`Ctrl`+`a`-`c`

- 現在のリージョン以外のリージョンをすべて終了してください（リージョンを終了してもウィンドウは終了しません）。次に、すべてのウィンドウを終了してください。そうするとようやくセッションが終了します。

  `Ctrl`+`a`-`Q`

  `exit`（Bashを終了）

  `Shift`+`:`と押した後に `quit` と入力しエンターキーを押す（`vi` を終了）

  `exit` （Bashを終了）

  `q` （`top` を終了）

  `exit` （Bashを終了）。

- 最後にセッションのリストを確認します。PIDを指定して `ssh` という名前のセッションを終了し、セッションがもう残っていないことを確認してください。

  `screen -list` または `screen -ls`

  `screen -S ▯▯▯▯▯▯▯ PID -X quit`

  `screen -list` または `screen -ls`

3. `tmux` をインストールして（パッケージ名は `tmux` です）、次の課題に取り組んでください。
   - `tmux` を開始してください。

     `tmux`

   - `top` を起動してください（数秒でステータスバーのウィンドウの名前が `top` に変わることに注目してください）。

     `top`

   - コマンドプレフィックスで新しいウィンドウを開き、 `nano` で `~/.tmux.conf` を作成してください。

`Ctrl`+`b`-`c`

```
nano ~/.tmux.conf
```

- ウィンドウを水平方向に分割し、新しく作成したペインのサイズを少し小さくしてください。

  `Ctrl`+`b`-`"`

  `Ctrl`+`b`-`Ctrl`+`↓`

- 現在のウィンドウの名前を `text editing` に変更し、 `tmux` のセッションのリストを表示してください。

  `Ctrl`+`b`-`,`を押してから新しい名前（`text editing`）を入力してエンターキーを押します

  `Ctrl`+`b`-`s` または `tmux ls`

- 同じコマンドを2回入力して、1回目で `top` を実行しているウィンドウに移動し、2回目で現在のウィンドウに戻ってください。

  `Ctrl`+`b`-`n` または `Ctrl`+`b`-`p`

- 現在のセッションをデタッチして、セッション名が `ssh` 、ウィンドウ名が `ssh window` の新しいセッションを作成してください。

  `Ctrl`+`b`-`d`

  ```
  tmux new -s "ssh" -n "ssh window"
  ```

- `ssh` という名前のセッションもデタッチして、 `tmux` のセッションのリストを表示してください。

  `Ctrl`+`b`-`d`

  ```
  tmux ls
  ```

> **NOTE**
>
> ここからはリモートマシンからローカルマシンに `ssh` 接続することが必要になります（仮想マシンを使ったリモート接続でも大丈夫です）。ローカルマシンで `openssh-server` をインストールして起動させ、リモートマシンで `openssh-client` をインストールしてください。

- リモートマシンからローカルマシンに `ssh` 接続してください。接続を確立できたら `tmux` のセッションを確認してください。

リモートホストで `ssh ユーザー名@ローカルマシンのIPアドレス` を実行します。ローカルマシンに接続したら `tmux ls` を実行します。

- リモートマシンで `ssh` という名前のセッションをアタッチしてください。

  `tmux a -t ssh`（`a` は `at` または `attach` でも可）

- ローカルマシンに戻り、`ssh` という名前のセッションにアタッチします。リモートホストとの接続を終了してからアタッチするコマンドを入力してください。

  `tmux a -d -t ssh`（`a` は `at` または `attach` でも可）

- すべてのセッションを表示し、最初のセッション（`[0]`）に移動します。移動後に `ssh` という名前のセッションを終了してください。

  `Ctrl`+`b`-`s`を押し、矢印キーでセッション `0` を選んでエンターキーを押します。それから `tmux kill-session -t ssh` を実行します。

- 最後に、現在のセッションをデタッチして、そのセッションを終了します。

  `Ctrl`+`b`-`d`

  `tmux kill-session -t 0`

発展演習の解答

1. `screen` でも `tmux` でもコマンドプレフィックスに続けて`:`（コロン）を入力するとコマンドラインモードになります（`tmux` での簡単な例はすでに見ました）。コマンドラインモードで次の課題に取り組んでください。
   - `screen` でコピーモードに入ってください。

     `Ctrl`+`a`-`:`を押してから `copy` と入力します。
   - `tmux` で現在のウィンドウの名前を変更してください。

     `Ctrl`+`b`-`:`を押してから `rename-window` と入力します。
   - `screen` ですべてのウィンドウを閉じてセッションを終了してください。

     `Ctrl`+`a`-`:`を押してから `quit` と入力します。
   - `tmux` でペインを分割してください。

     `Ctrl`+`b`-`:`を押してから `split-window` と入力します。
   - `tmux` で現在のウィンドウを終了してください。

     `Ctrl`+`b`-`:`を押してから `kill-window` と入力します。
2. `screen` でコピーモードに入ると、矢印キーや`PgUP`、`PgDown`キーで現在のウィンドウを遡れますが、`vi` 風のフルスクリーンエディタを使うこともできます。次の課題に取り組んでください。
   - `screen` の端末で `supercalifragilisticexpialidocious` と出力してください。

     ```
     echo supercalifragilisticexpialidocious
     ```
   - カーソルの真上にある5つの連続する文字をコピーしてください。

     `Ctrl`+`a`-`[`と押すか、`Ctrl`+`a`-`:`を押した後に`copy`と入力して、コピーモードに入ります。 次に、`k` を押して上の行に移動し、スペースキーを押して選択する部分の先頭にマークをつけます。最後に、 `l` を4回押して4文字前に進み、スペースキーをもう一度押して選択する部分の末尾にマークをつけます。
   - 選択した部分（筆者の環境では `stice` の5文字になりました）をコマンドプロンプトにペーストしてください。

     `Ctrl`+`a`-`]`

3. tmux のセッション（our_session）を別のユーザーと共有します。相手のユーザーも読み書きできるパーミッションでソケット（/tmp/our_socket）を作ってあります。相手のユーザーが `tmux -S /tmp/our_socket a -t our_session` でセッションをアタッチできるようにするには、あと２つしなければならないことがあります。それは何でしょう？

   両方のユーザーが共通のグループに所属している必要があります（ここでは `multiplexer` というグループにします）。次に、ソケットの所有グループをそのグループに変更する必要があります（`chgrp multiplexer /tmp/our_socket` を実行します）。

103.6 プロセス実行の優先順位を変更する

LPI目標への参照

LPIC-1 v5, Exam 101, Objective 103.6

総重量

2

主な知識分野

- 作成されたジョブのデフォルト優先順位を知る。
- デフォルトよりも高いまたは低い優先順位でプログラムを実行する。
- 実行中のプロセスの優先度を変更する。

用語とユーティリティ

- `nice`
- `ps`
- `renice`
- `top`

103.6 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	103 GNUおよびUnixコマンド
Objective:	103.6 プロセス実行の優先順位を変更する
Lesson:	1 of 1

はじめに

複数のプロセスを同時に実行できるOSを、マルチタスクOSないしマルチプロセッシングOSと呼びます。厳密には、複数のプロセスを同時に実行するには複数の処理装置（訳注：CPUなどの命令実行用のプロセッサ）が必要ですが、実行するプロセスを高速に切り替えることで、擬似的に同時実行をしているように見せることができます。処理装置よりもはるかに多くのプロセスを実行したいのが普通ですから、SMP（対称型マルチプロセッシング）など複数の処理装置を備えたシステムであっても、プロセスを高速に切り替える仕組みが採られています。

CPUが一度に処理できるのは一つのプロセスだけです。そして、プロセスの処理内容の大部分は、CPUを直接利用する演算などではなく、OSの機能を呼び出すシステムコールで行われます。システムコールでは主に、メモリの割り当て、ファイルシステムの読み書き、テキストの画面表示などのユーザーとの対話、ネットワーク転送など、デバイスとのやり取り（入出力）が行われます。現在のシステムでは、デバイスとのやり取りにはCPUの速度に比べて桁違いの長い時間がかかりますから、デバイスの応答を待っている間に別のプロセスにCPUを使用させると効率的です。そのため、マルチプロセッシングOSでは、システムコールが行われる度に実行するプロセスを選択しなおします（プロセススイッチと呼びます）。その際には、複数のキューを動的に使い分けることで、プロセスを効果的に管理しています。

しかしながら、システムコールの時にだけプロセス切り替え（プロセススイッチ）を行うの

では十分に効率的とは言えません。プロセスの中には、システムコールを行わずに長時間CPUを使い続けるものもあるからです。そのため現代的なOSでは、（タイマーによって）実行中のプロセスに割り込んで強制的にCPUの利用権を取り上げて、別のプロセスに切り替える仕組み（プリエンプション）を備えています。Linuxは、現代的なプリエンプティブ・マルチタスクOSの1つ、ということです。

Linuxのスケジューラー

Linuxはプリエンプティブ・マルチプロセッシングOSですから、プロセスのキューを管理するスケジューラーがあります。より正確に言うと、スケジューラーはプロセスから分岐した個々の　スレッド　も管理しますが、ここではプロセスもスレッドも同じだと考えても構いません。プロセスにはスケジューリングを調整する2つのパラメータがあります。ポリシーと 優先度 です。

ポリシーには2種類あります。リアルタイムポリシー　と　通常ポリシー　です。リアルタイムポリシーのプロセス同士では、そのまま優先度順にスケジューリングされます。より優先度の高いプロセスが実行可能になると、現在実行中のプロセスはプリエンプトされ、CPUはより優先度の高いプロセスの処理をします。より優先度の高いプロセスがアイドル状態になるかハードウェア（デバイス）の応答待ちになるまで、優先度の低いプロセスは処理されません。

リアルタイムポリシーのプロセスは通常ポリシーのプロセスよりも優先的に処理されます。Linuxは汎用OSであり、リアルタイムポリシーのプロセスはわずかしかありません。システムのプログラムもユーザーのプログラムも、大部分は通常ポリシーのプロセスです。通常ポリシーのプロセスはたいてい同じ優先度ですが、その中での優先順位を　nice値　で定めることができます。通常ポリシーのプロセス同士で優先順位を定めるnice値のことを動的優先度と呼び、全プロセスの優先順位を決める優先度（静的 優先度）と区別します。

Linuxのスケジューラーには様々な設定があり、優先順位を決める複雑な方法も存在しますが、ポリシーと優先度という2つの概念はどの方法にも当てはまります。プロセスのスケジューリングを詳しく調べて調整する際には、通常ポリシーのプロセスの優先度を調整しているのだということを忘れないようにしてください。

優先順位を知る

Linuxでは、リアルタイムポリシーのプロセスに0から99の数値で静的優先度が割り振られ、通常ポリシーのプロセスには100から139の数値で静的優先度が割り振られます。通常ポリシーのプロセスの優先度には40段階あるということです。値が低いほど優先度が高くなります。　`/proc`　ファイルシステム内でプロセスごとに作成される　`sched`　ファイルに静的優先度が書かれています。

```
$ grep ^prio /proc/1/sched
prio                  :       120
```

`prio` で始まる行に静的優先度が書かれています（上の例で見ているPIDが1のプロセスは、カーネルがシステムの初期化の際に開始する最初のプロセスである init ないし systemd です）。通常ポリシーのプロセスの優先度は標準で120です。この値を100から139の間で増減できます。`ps -Al` か `ps -el` のコマンドで全ての実行中のプロセスの優先度を確かめられます。

```
$ ps -el
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 -  9292 -      ?        00:00:00 systemd
4 S     0    19     1  0  80   0 -  8817 -      ?        00:00:00 systemd-journal
4 S   104    61     1  0  80   0 - 64097 -      ?        00:00:00 rsyslogd
4 S     0    63     1  0  80   0 -  7244 -      ?        00:00:00 cron
1 S     0   126     1  0  80   0 -  4031 -      ?        00:00:00 dhclient
4 S     0   154     1  0  80   0 -  3937 -      pts/0    00:00:00 agetty
4 S     0   155     1  0  80   0 -  3937 -      pts/1    00:00:00 agetty
4 S     0   156     1  0  80   0 -  3937 -      pts/2    00:00:00 agetty
4 S     0   157     1  0  80   0 -  3937 -      pts/3    00:00:00 agetty
4 S     0   158     1  0  80   0 -  3937 -      console  00:00:00 agetty
4 S     0   160     1  0  80   0 - 16377 -      ?        00:00:00 sshd
4 S     0   280     0  0  80   0 -  5301 -      ?        00:00:00 bash
0 R     0   392   280  0  80   0 -  7221 -      ?        00:00:00 ps
```

`PRI` 列から静的優先度がわかります。先ほどの例で見た値（120）と違う値が表示されていることに注意してください。歴史的な理由から、`ps` で表示される優先度はデフォルトでは-40から99です。実際の優先度を求めるには40を足します（上の例では80 + 40 = 120になります）。

`top` コマンドを実行すると、Linuxカーネルに管理されているプロセスのリアルタイムの状態を動的に監視できます。`top` の優先度の値の表示は、これまた優先度そのものではありません。リアルタイムポリシーのプロセスと通常ポリシーのプロセスを識別しやすくするために、`top` では100を引いた優先度が表示されます。これにより、リアルタイムポリシーのプロセスは負の値になります（リアルタイムを意味する rt と表示されることもあります）。通常ポリシーのプロセスは0から39になります。

NOTE

いろいろなオプションを指定して `ps` コマンドから詳細な情報を得ることができます。本文中の例と次のコマンドから得られる結果とを比べてみてください。

```
$ ps -e -o user,uid,comm,tty,pid,ppid,pri,pmem,pcpu --sort=-pcpu |
head
```

プロセスのnice値

通常ポリシーのプロセスは、デフォルトではnice値が0（優先度は120）で開始されます。niceなプロセスは実行待ちキューで他のプロセスに優先順位を譲るという発想からnice という名前が付けられました。nice値は-20（niceなプロセスではなく優先度が高い）から19（niceなプロセスで優先度が低い）までの範囲です。Linuxでは同じプロセスから分岐した各スレッドに異なるnice値を割り当てることもできます。`ps` の `NI` 列に nice値が示されます。

プロセスのnice値を減らす（優先度を上げる）ことができるのはrootユーザーだけです。`nice` コマンドで標準とは異なる優先度（0とは異なるnice値）でプロセスを開始できます。`nice` コマンドをデフォルトで使うとnice値が10になり、`-n` オプションに続けて値を指定するとnice値が指定した値になります。

```
$ nice -n 15 tar czf home_backup.tar.gz /home
```

このようにすると、`tar` コマンドはnice値15で実行されます。`renice` コマンドで実行中のプロセスの優先度（nice値）を変更できます。`-p` オプションに続けてPIDを指定します（nice値を減らすことができるのはrootユーザーだけだということに注意してください）。

```
# renice -10 -p 2164
2164 (process ID) old priority 0, new priority -10
```

`-g` オプションに続けてPGID（プロセスグループID）を指定するとそのPGIDのプロセス全部の優先度（nice値）を一括して変更でき、`-u` オプションに続けてユーザー名を指定するとそのユーザーが開始したプロセス全部の優先度（nice値）を一括して変更できます。`renice +5 -u users` を実行すると、users という名前のユーザーが開始したプロセスのnice値が5増えます。

`renice` コマンド以外でも `top` コマンドでプロセスの優先度（nice値）を変えられます。`top` の出力が表示されている画面で `r` を押してからPIDを入力してエンターキーを押し、nice値を入力してエンターキーを押すとその値に変えられます（訳注：プロセスのnice値を減らす（優先度を上げる）ことができるのはrootユーザーだけです）。

```
top - 11:55:21 up 23:38,  1 user,  load average: 0,10, 0,04, 0,05
Tasks:  20 total,   1 running,  19 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0,5 us,  0,3 sy,  0,0 ni, 99,0 id,  0,0 wa,  0,2 hi,  0,0 si,  0,0 st
KiB Mem :  4035808 total,   774700 free,  1612600 used,  1648508 buff/cache
KiB Swap:  7999828 total,  7738780 free,   261048 used.  2006688 avail Mem
PID to renice [default pid = 1]
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    1 root      20   0   74232   7904   6416 S 0,000 0,196   0:00.12 systemd
   15 root      20   0   67436   6144   5568 S 0,000 0,152   0:00.03 systemd-
```

```
journal
    21 root      20   0   61552   5628   5000 S 0,000 0,139   0:00.01 systemd-logind
    22 message+  20   0   43540   4072   3620 S 0,000 0,101   0:00.03 dbus-daemon
    23 root      20   0   45652   6204   4992 S 0,000 0,154   0:00.06 wickedd-dhcp4
    24 root      20   0   45648   6276   5068 S 0,000 0,156   0:00.06 wickedd-auto4
    25 root      20   0   45648   6272   5060 S 0,000 0,155   0:00.06 wickedd-dhcp6
```

`PID to renice [default pid = 1]` というメッセージに示されているように、PIDを入力せずにエンターキーを押すと、最初の行に表示されているプロセスのPIDがデフォルトで選ばれます。そのプロセスとは別のプロセスを選ぶ場合はPIDを入力してエンターキーを押してください。`Renice PID 1 to value` のようなメッセージが表示されるので、変更後のnice値を入力してエンターキーを押すとその値に変更できます。

演習

1. プリエンプティブ・マルチタスクOSでは、CPUがあるプロセスの処理をしているときにもっと優先度の高いプロセスが実行キューに入るとどうなりますか？

2. 以下の `top` の画面表示を見て問いに答えてください。

```
top - 08:43:14 up 23 days, 12:29,  5 users,  load average: 0,13, 0,18, 0,21
Tasks: 240 total,   2 running, 238 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1,4 us,  0,4 sy,  0,0 ni, 98,1 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   7726,4 total,    590,9 free,   1600,8 used,   5534,7 buff/cache
MiB Swap:  30517,0 total,  30462,5 free,     54,5 used.   5769,4 avail Mem
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    1 root      20   0  171420  10668   7612 S   0,0   0,1   9:59.15 systemd
    2 root      20   0       0      0      0 S   0,0   0,0   0:02.76 kthreadd
    3 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_par_gp
    8 root       0 -20       0      0      0 I   0,0   0,0   0:00.00
mm_percpu_wq
    9 root      20   0       0      0      0 S   0,0   0,0   0:49.06
ksoftirqd/0
   10 root      20   0       0      0      0 I   0,0   0,0  18:24.20 rcu_sched
   11 root      20   0       0      0      0 I   0,0   0,0   0:00.00 rcu_bh
   12 root      rt   0       0      0      0 S   0,0   0,0   0:08.17
migration/0
   14 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/0
   15 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/1
   16 root      rt   0       0      0      0 S   0,0   0,0   0:11.79
migration/1
   17 root      20   0       0      0      0 S   0,0   0,0   0:26.01
ksoftirqd/1
```

   リアルタイムポリシーのプロセスはどれですか。PIDを答えてください。

3. 以下の `ps-el` の画面表示を見て問いに答えてください。

```
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 42855 -      ?        00:09:59 systemd
1 S     0     2     0  0  80   0 -     0 -      ?        00:00:02 kthreadd
1 I     0     3     2  0  60 -20 -     0 -      ?        00:00:00 rcu_gp
```

```
1 S     0     9     2  0  80   0 -     0 -      ?        00:00:49 ksoftirqd/0
1 I     0    10     2  0  80   0 -     0 -      ?        00:18:26 rcu_sched
1 I     0    11     2  0  80   0 -     0 -      ?        00:00:00 rcu_bh
1 S     0    12     2  0 -40   - -     0 -      ?        00:00:08 migration/0
1 S     0    14     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/0
5 S     0    15     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/1
```

優先度が最も高いプロセスはどれですか。PIDを答えてください。

4. `renice` コマンドでプロセスのnice値を変更しようとしたところ、以下のエラーメッセージが表示されました。

```
$ renice -10 21704
renice: failed to set priority for 21704 (process ID): Permission denied
```

このエラーの原因は何でしょう？

発展演習

1. プロセスの優先度を変更する必要を感じるのは、たいてい、あるプロセスがあまりにも多くのCPU時間を占有しているときです。そこで、`ps` コマンドにオプションを指定してすべてのシステムプロセスの詳細表示を見てみましょう。CPU使用率の昇順でソートするにはどのような `--sort` オプションを指定しますか？

2. `schedtool` は、CPUスケジューリングに関するあらゆるパラメータを設定・表示できるコマンドです。（`top` や `ps` の表示から適当にPIDを選んで）指定したプロセスのスケジューリングパラメータを表示してください。また、そのプロセスの優先度を-90（`top` の表示によるもの）に設定してリアルタイムポリシーに変更してください。

まとめ

このレッスンでは、Linuxが管理しているプロセスの間でCPU時間をどのように割り当てているかを説明しました。システムがベストのパフォーマンスを発揮するには、重要なプロセスがそれほど重要でないプロセスよりも優先的に処理されるべきです。以下の順で説明しました。

- マルチプロセッシングOSの基本概念。
- プロセススケジューラーとスケジューリングの実施方法。
- Linuxでの優先度とnice値。
- Linuxでのプロセス優先度の解釈。
- 実行前及び実行中におけるプロセス優先度の変更方法。

演習の解答

1. プリエンプティブ・マルチタスクOSでは、CPUがあるプロセスの処理をしているときにもっと優先度の高いプロセスが実行キューに入るとどうなりますか？

   そのプロセスの処理を中断し、代わりに優先度の高いプロセスの実行処理をします。

2. 以下の `top` の画面表示を見て問いに答えてください。

```
top - 08:43:14 up 23 days, 12:29,  5 users,  load average: 0,13, 0,18, 0,21
Tasks: 240 total,   2 running, 238 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1,4 us,  0,4 sy,  0,0 ni, 98,1 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   7726,4 total,    590,9 free,   1600,8 used,   5534,7 buff/cache
MiB Swap:  30517,0 total,  30462,5 free,     54,5 used.   5769,4 avail Mem
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    1 root      20   0  171420  10668   7612 S   0,0   0,1   9:59.15 systemd
    2 root      20   0       0      0      0 S   0,0   0,0   0:02.76 kthreadd
    3 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_par_gp
    8 root       0 -20       0      0      0 I   0,0   0,0   0:00.00
mm_percpu_wq
    9 root      20   0       0      0      0 S   0,0   0,0   0:49.06
ksoftirqd/0
   10 root      20   0       0      0      0 I   0,0   0,0  18:24.20 rcu_sched
   11 root      20   0       0      0      0 I   0,0   0,0   0:00.00 rcu_bh
   12 root      rt   0       0      0      0 S   0,0   0,0   0:08.17
migration/0
   14 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/0
   15 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/1
   16 root      rt   0       0      0      0 S   0,0   0,0   0:11.79
migration/1
   17 root      20   0       0      0      0 S   0,0   0,0   0:26.01
ksoftirqd/1
```

   リアルタイムポリシーのプロセスはどれですか。PIDを答えてください。

   PID12と16。

3. 以下の `ps-el` の画面表示を見て問いに答えてください。

```
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 42855 -      ?        00:09:59 systemd
1 S     0     2     0  0  80   0 -     0 -      ?        00:00:02 kthreadd
```

```
1 I     0     3     2  0  60 -20 -     0 -      ?        00:00:00 rcu_gp
1 S     0     9     2  0  80   0 -     0 -      ?        00:00:49 ksoftirqd/0
1 I     0    10     2  0  80   0 -     0 -      ?        00:18:26 rcu_sched
1 I     0    11     2  0  80   0 -     0 -      ?        00:00:00 rcu_bh
1 S     0    12     2  0 -40   - -     0 -      ?        00:00:08 migration/0
1 S     0    14     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/0
5 S     0    15     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/1
```

優先度が最も高いプロセスはどれですか。PIDを答えてください。

PID12。

4. `renice` コマンドでプロセスのnice値を変更しようとしたところ、以下のエラーメッセージが表示されました。

```
$ renice -10 21704
renice: failed to set priority for 21704 (process ID): Permission denied
```

このエラーの原因は何でしょう？

nice値を減らすことができるのはrootユーザーだけなのに、一般ユーザーがnice値を減らそうとしたこと。

発展演習の解答

1. プロセスの優先度を変更する必要を感じるのは、たいてい、あるプロセスがあまりにも多くのCPU時間を占有しているときです。そこで、`ps` コマンドにオプションを指定してすべてのシステムプロセスの詳細表示を見てみましょう。CPU使用率の昇順でソートするにはどのような `--sort` オプションを指定しますか？

```
$ ps -el --sort=pcpu
```

2. `schedtool` は、CPUスケジューリングに関するあらゆるパラメータを設定・表示できるコマンドです。（`top` や `ps` の表示から適当にPIDを選んで）指定したプロセスのスケジューリングパラメータを表示してください。また、そのプロセスの優先度を-90（`top` の表示によるもの）に設定してリアルタイムポリシーに変更してください。

```
$ schedtool 1750
```

```
$ schedtool -R -p 89 1750
```

103.7 正規表現を使ってテキストファイルを検索する

LPI目標への参照

LPIC-1 v5, Exam 101, Objective 103.7

総重量

3

主な知識分野

- いくつかの表記要素を含む単純な正規表現を作成する。
- 標準正規表現と拡張正規表現の違いを理解する。
- 特殊文字・文字クラス・数量詞・アンカーの概念を理解する。
- 正規表現ツールを使用して、ファイルシステムまたはファイルコンテンツに対して検索を実行する。
- 正規表現を利用して、テキストの削除、変更、置換ができる。

用語とユーティリティ

- `grep`
- `egrep`
- `fgrep`
- `sed`
- `regex(7)`

103.7 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	103 GNUおよびUnixコマンド
Objective:	103.7 正規表現によるテキストファイルの検索
Lesson:	1 of 2

はじめに

データを加工する作業では文字列検索アルゴリズムを多用します。そこで、どのUnix系のOSにも正規表現（Regular Expression、RE）が備わっています。正規表現は一般的なパターンを表す文字の連なりから構成されます。長いテキストの中で対応する部分を見つけ出したり変更したりするために使います。正規表現は以下のような作業で威力を発揮します。

- 表形式のテキストデータを別のフォーマットに変換するスクリプトを書く。
- ログやジャーナル、文書などから気になる箇所を検索する。
- HTTPサーバー（特に nginx）でリクエストを解析するルールを書く。
- 文書の構造を保持しながら、マークアップ文書のタグを変更する。

正規表現はいくつかの基本要素（アトム）を組み合わせたものです。多くの文字はそのままアトムになりますが、次のように特別な意味や働きを持ったアトムもあります。

`.`（ドット）
　改行を除き、どの文字にもマッチするアトムです。

^ (キャレット)
行頭にマッチするアトムです。

$ (ドル記号)
行末にマッチするアトムです。

例えば、`bc` という正規表現は、そのままその文字を表す `b` と `c` のアトムから構成されていて、`abcd` という文字列にマッチしますが、`a1cd` という文字列にはマッチしません。`.c` という正規表現は、`abcd` と `a1cd` の両方の文字列にマッチします。`.` はどの文字にもマッチするからです。

`^`（キャレット）と `$`（ドル記号）のアトムは、行頭ないし行末にだけマッチさせたいときに使います。これらは アンカー とも呼ばれます。例えば、`abcd` という行に対し、`cd` はマッチしますが、`^cd` はマッチしません。同様に、`abcd` という行に対し、`ab` はマッチしますが、`ab$` はマッチしません。`^` は正規表現の先頭以外ではそのまま `^` を表し、`$` は正規表現の末尾以外ではそのまま `$` を表します。

[]で囲まれた文字クラス

`[]` で囲まれた 文字クラス というアトムもあります。`[]` 内の複数の文字が一つのアトムとみなされます。文字クラスは、基本的に、`[]` 内の複数の文字をそのままマッチの候補リストだと考えます。その候補リストのいずれかの文字にマッチするアトムだということです。例えば、`[1b]` という文字クラスは、`abcd` にも `a1cd` にもマッチします。`[^1b]` のように `^` で始めると、意味が反転されて、`1` と `b` 以外のすべての文字という意味になります。文字クラスでは文字の範囲を指定することもできます。例えば `[0-9]` は0から9の数字にマッチし、`[a-z]` は小文字のアルファベットにマッチします。文字の範囲や比較方法は言語設定（ロケール）によって異なるので、ASCII文字以外を使用する場合には気をつけてください。

文字クラスはクラス名で指定することもできます。次のようなクラス名があります。

`[:alnum:]`
アルファベットと数字

`[:alpha:]`
アルファベット

`[:ascii:]`
ASCII文字

`[:blank:]`
スペースとタブ

`[:cntrl:]`
制御文字

`[:digit:]`
数字（0から9）

`[:graph:]`
スペースを除く印刷可能文字

`[:lower:]`
小文字

`[:print:]`
スペースを含む印刷可能文字

`[:punct:]`
句読記号（!"#$%&'()*+,-./:;<⇒?@[\]^_`{|}~）

`[:space:]`
スペース、フォームフィード（`\f`）、改行（`\n`）、キャリッジリターン（`\r`）、水平タブ（`\t`）、垂直タブ（`\v`）

`[:upper:]`
大文字

`[:xdigit:]`
16進数の数字（0からF）

クラス名は、文字や範囲と併用できますが、範囲の終わりとして使うことはできません。また、クラス名を　`[]`　の外で使うことはできず、単なる文字列として扱われます。つまり、クラス名を使用する場合には、`[[:digit:]]`　のように、二重の角かっこで囲むことになります。

量指定子

アトムの繰返し回数を指定する　量指定子　の概念を使うと、複雑なパータンをシンプルに指定することができます。量指定子は直前のアトム（文字そのもの、ドット、文字クラスのいずれでも）が何回連続して現れるかを指定し、条件に一致する文字列ににマッチします。マッチした部分文字列を　ピース（piece）　と呼ぶことがあります（訳注:　日本では意味のまま マッチ や 一致 と呼ばれることのほうが多いようです）。

POSIXで規格化されている正規表現には、 “基本” 正規表現と “拡張” 正規表現の二種類があります。Linuxのテキスト関連プログラムの多くでは拡張正規表現が使えます（オプシ

ョン指定が必要なものもあります）。表記方法の違いを知っておいたほうがよいですが、拡張正規表現を覚えてしまえば、あえて基本正規表現を使用する機会はほとんどありません。

基本正規表現の量指定子では、`*`（アスタリスク）のみが使えます。直前のアトム（文字そのもの、ドット、文字クラスのいずれでも）の　0回以上　の繰り返しを意味します。0回以上ですから、文字がなくてもマッチすることに気をつけてください。例えば `.*`（ドット・アスタリスク）は、文字や記号が何個あってもいいし全くなくてもいいという意味になります。

拡張正規表現の量指定子では、アスタリスクに加えて `+`（プラス）と `?`（クエスチョンマーク）が使えます。プラスは　1回以上　の、クエスチョンマークは　0回ないし1回　の繰り返しを意味します。

いずれも、量指定子としての特別な意味を打ち消してそのままの文字として扱いたいときは、直前に　\（バックスラッシュ）を置いてエスケープします。また、正規表現の先頭にこれらの文字が置かれた場合には、前にアトムがないのでそのままの文字として扱われます。

繰り返し指定

具体的な繰り返し回数を指定する量指定子も使えます。{i,j}　　（iとjは整数）と指定することで、i文字以上、j文字以下　　の繰り返しを意味します。jを省略したケースを含めて、3つの指定方法があります。

`{i}`

直前のアトムをちょうど　`i`　回繰り返すことを示します。例えば、`[[:blank:]]{2}`　は2個の連続する空白文字（スペースないしタブ）にマッチします。

`{i,}`

直前のアトムを少なくとも　　`i`　　回繰り返すことを示します。例えば、`[[:blank:]]{2,}` は、2個以上の連続する空白文字（スペースないしタブ）にマッチします。

`{i,j}`

直前のアトムを　`i`　回以上　`j`　回以下繰り返すことを示します。（`j`　は　`i`　より大きい）。例えば、`xyz{2,4}` は、`xy` の後に `z` が2〜4個連続する場合にマッチします。

量指定子を含む正規表現では、先頭から検索を始めて、最も長い部分文字列がマッチしたと判定されます。

基本正規表現でも繰り返し指定を使うことができますが、`\{`　や　`\}`　のように　`\`　を付けなければなりません。`{`　や　`}`　だけではそのまま　`{`　や　`}`　と解釈されます。`\{`　の後に数字以外の文字があると、繰り返し指定ではなく、文字 `{` と解釈されます。

選択とグループ化

いくつかの文字列 のどれかとの一致を探したいときには、文字列を | （パイプ）で区切る ▯▯（ブランチと呼ぶこともある）を使います。例えば、he|him は、対象文字列の中に he か him があればマッチします。なお、パイプによる選択は、基本正規表現では使えません（\| で使えるものもあります）。

文字列の前後にさらに別の文字列や正規表現がある場合には、文字列の範囲を明示することが必要なことがあります。( ) で文字列やパターンを囲んでグループ化すると、囲まれた範囲が1つのアトムになります。例えば (AB){3} はABABABにマッチします。また、(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[[:blank:]]\[[:digit:]] と書くと、3文字月名のいずれかの後にスペースないしタブが1つ以上あり、その後に1つ以上の数字が続く文字列（Mar 15など）にマッチします。

( ) によるグループ化には、囲まれた範囲を1つのアトムにすることに加えて、後方参照を可能にするという重要な意味があります。正規表現中の ( ) で囲まれたグループには、左から順に 1,2,3… と番号が振られます。そして、正規表現の後の部分で \1, \2, \3 と番号の前にバックスラッシュを置くと、対応する グループにマッチした文字列 と同じ文字列とのマッチを示します。例えば、([[:digit:]]{3})ABCD\1 は、3桁の数値の後にABCDがあり、その後に一致した3桁の数字が繰り返される文字列（351ABCD351など）にマッチします。

なお、基本正規表現では、単純な (と) ではなく、エスケープした \(と \) で囲まなければなりません。(や) はそのまま (や) を表します。\1, \2, \3 と番号の前にバックスラッシュを置く後方参照については拡張正規表現と同じです。

正規表現を用いた検索

ファイルシステムやテキスト文書を検索する際に正規表現は効力を発揮します。例えば find コマンドに -regex オプションを付けると、正規表現を使ってパスの探索ができます。例を見てみましょう。

```
$ find $HOME -regex '.*/\..*' -size +100M
```

このコマンドは、ファイルサイズが100メガバイト（104857600バイト）以上のファイルを探索しています。ただし、ユーザーのホームディレクトリ以下で .*/\..* とマッチするパス、つまり /. が含まれているパス（/. の前後に文字があってもよい）だけを探索しています。要するに、隠しファイルか隠しディレクトリ以下のファイルだけが探索対象になるということです。.* が最も長い文字列に一致しますから、パス中での /. の位置は問いません。次の例では、-iregex オプションを指定して、正規表現中で大文字と小文字を区別しないことを指定します。

```
$ find /usr/share/fonts -regextype posix-extended -iregex
'.*(dejavu|liberation).*sans.*(italic|oblique).*'
```

```
/usr/share/fonts/dejavu/DejaVuSansCondensed-BoldOblique.ttf
/usr/share/fonts/dejavu/DejaVuSansCondensed-Oblique.ttf
/usr/share/fonts/dejavu/DejaVuSans-BoldOblique.ttf
/usr/share/fonts/dejavu/DejaVuSans-Oblique.ttf
/usr/share/fonts/dejavu/DejaVuSansMono-BoldOblique.ttf
/usr/share/fonts/dejavu/DejaVuSansMono-Oblique.ttf
/usr/share/fonts/liberation/LiberationSans-BoldItalic.ttf
/usr/share/fonts/liberation/LiberationSans-Italic.ttf
```

この例では、`/usr/share/fonts` ディレクトリ以下で指定のフォントファイルだけを見つけるために、拡張 正規表現のパイプによる選択を使っています。`find` コマンドのデフォルトでは拡張正規表現になりませんが、`-regextype posix-extended` か `-regextype egrep` オプションを指定すると、拡張正規表現が使用できます。`find` コマンドのデフォルトの正規表現は、findutils-default というものであり、基本正規表現とほとんど同じです。

コマンドの出力が1画面に収まりきらないときは、`less` コマンドを使うのが普通です。`less` を使えば、ユーザーはファイルの好きな場所に移動できますし、正規表現を使った検索をすることもできます。この機能はとても重要です。というのも、systemdジャーナルを探ったりマニュアルを参照したりするときなどには、`less` がデフォルトのページャになっていることが多いからです。例えば、マニュアルを参照しているときに`/`（スラッシュ）を押すと検索プロンプトが表示されますが、ここで正規表現が役立ちます。あるオプションを探したいとしましょう。マニュアルの中にはオプション自体が何度も登場するのが普通ですから、例えば `-o` オプションを探すのに `-o` とだけ指定するといくつもマッチしてしまいます。オプション自体の説明箇所では、行頭からいくつかの空白があり、その後にオプションが書かれていることが多いので、プロンプトに `^[[:blank:]]*-o` やもっと単純に `^ *-o` と入力すれば、その行に一発で行き着けます。

演習

1. `egrep` コマンドで `info@example.org` のような一般的なメールアドレスにマッチする拡張正規表現を書いてください。

2. `egrep` コマンドで `192.168.15.1` のような標準的なドット区切り十進表記のIPv4アドレスにマッチする拡張正規表現を書いてください。

3. `grep` コマンドで `/etc/services` ファイルのコメント行（`#` で始まる行）以外の中身を表示してください。

4. `domains.txt` ファイルにドメイン名が一行に1つずつ書かれているとします。`egrep` コマンドで `.org` ドメインか `.com` ドメインのものだけを表示してください。

発展演習

1. `find` コマンドで拡張正規表現を使い、カレントディレクトリ以下でサフィックスを含まないファイル（ファイル名が `.txt` や `.c` などで終わらないファイル）を探してください。

2. `less` はシェル環境で長いテキストを表示する際のデフォルトのページャです。`/` を押すと、検索プロンプトに正規表現を入力でき、最初にマッチした部分にジャンプします。ジャンプせずに文書中の現在の場所のままでマッチした部分をハイライトするには、検索プロンプトでどのキーの組み合わせを入力すればよいでしょうか。

3. `less` で出力をフィルタリングして正規表現にマッチした行だけ表示するにはどうすればよいでしょうか。

まとめ

このレッスンではLinuxで一般的な正規表現について学びました。テキスト関連のプログラムでは正規表現のパターンマッチ機能を使えることがほとんどです。このレッスンは以下の順で取り上げました。

- 正規表現とは
- 正規表現の主な構成要素
- 基本正規表現と拡張正規表現の違い
- 正規表現を用いたテキストやファイルの検索

演習の解答

1. `egrep` コマンドで `info@example.org` のような一般的なメールアドレスにマッチする拡張正規表現を書いてください。

```
egrep -i "[a-z][a-z0-9_+\.\-]+@([a-z0-9][a-z0-9\-]*[a-z0-9]*\.)+[a-z]{2,}"
```

（訳注: メールアドレスの規格上の仕様は複雑であり、この正規表現は「正しくない」一部のメールアドレスにもマッチします。）

2. `egrep` コマンドで `192.168.15.1` のような標準的なドット区切り十進表記のIPv4アドレスにマッチする拡張正規表現を書いてください。

```
egrep "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}"
```

3. `grep` コマンドで `/etc/services` ファイルのコメント行（`#` で始まる行）以外の中身を表示してください。

```
grep -v ^# /etc/services
```

4. `domains.txt` ファイルにドメイン名が一行に1つずつ書かれているとします。`egrep` コマンドで `.org` ドメインか `.com` ドメインのものだけを表示してください。

```
egrep ".org$|.com$" domains.txt
```

発展演習の解答

1. `find` コマンドで拡張正規表現を使い、カレントディレクトリ以下でサフィックスを含まないファイル（ファイル名が `.txt` や `.c` などで終わらないファイル）を探してください。

```
find . -type f -regextype egrep -not -regex '.*\.[[:alnum:]]{1,}$'
```

2. `less` はシェル環境で長いテキストを表示する際のデフォルトのページャです。`/` を押すと、検索プロンプトに正規表現を入力でき、最初にマッチした部分にジャンプします。ジャンプせずに文書中の現在の場所のままでマッチした部分をハイライトするには、検索プロンプトでどのキーの組み合わせを入力すればよいでしょうか。

   `/` を押してから正規表現を入力する前に `Ctrl`+`K` を押します。

3. `less` で出力をフィルタリングして正規表現にマッチした行だけ表示するにはどうすればよいでしょうか。

   `&` を押してから正規表現を入力します。

103.7 レッスン 2

Certificate:	LPIC-1
Version:	5.0
Topic:	103 GNUおよびUnixコマンド
Objective:	103.7 正規表現を使用してテキストファイルを検索する
Lesson:	2 of 2

はじめに

コマンドをパイプでつなげることにより、正規表現を使ってフィルターを組み合わせてデータストリームの処理ができます。システム管理だけでなくデータマイニングの領域でも正規表現は重要な技術になります。正規表現を使ってファイルやテキストを処理するのに最適な2つのコマンドがあります。`grep` と `sed` です。`grep` はパターン発見器で、`sed` はストリームエディタです。これらは単独でも有用ですが、他のプログラムと協働する場面で際立ちます。

パターン発見器：grep

`grep` が長いテキストファイルを探索する手助けになることがよくあります。正規表現をテキストファイルの各行に適用するフィルターとして使うのです。例えば、ある単語で始まる行だけを表示するのに使えます。カーネルモジュールの設定ファイルでオプション行だけを表示するならこのようにします。

```
$ grep '^options' /etc/modprobe.d/alsa-base.conf
options snd-pcsp index=-2
options snd-usb-audio index=-2
options bt87x index=-2
```

```
options cx88_alsa index=-2
options snd-atiixp-modem index=-2
options snd-intel8x0m index=-2
options snd-via82xx-modem index=-2
```

パイプ（|）を使えばコマンドの出力を直接 `grep` の入力にリダイレクトできます。以下の例では、文字クラスを使って、`fdisk -l` の出力のうち、`Disk/dev/sda` または `Disk/dev/sdb` で始まる行だけを表示しています。

```
# fdisk -l | grep '^Disk /dev/sd[ab]'
Disk /dev/sda: 320.1 GB, 320072933376 bytes, 625142448 sectors
Disk /dev/sdb: 7998 MB, 7998537728 bytes, 15622144 sectors
```

マッチした行を表示するだけでは事足りないこともあるでしょう。そういうときは、`grep` のオプションを指定して動作の調整ができます。例えば、`-c` または `--count` オプションを指定すると、マッチした行の総数がわかります。

```
# fdisk -l | grep '^Disk /dev/sd[ab]' -c
2
```

オプション指定は正規表現の前でも後でも構いません。`grep` でよく使うオプションは次のとおりです。

`-c` または `--count`
マッチした行を表示するのではなく、マッチした行の総数を表示します。

`-i` または `--ignore-case`
大文字と小文字を区別せずにマッチをします。

`-f FILE` または `--file=FILE`
正規表現が書かれたファイルを指定します。

`-n` または `--line-number`
行番号を表示します。

`-v` または `--invert-match`
マッチしなかった行を表示します。

`-H` または `--with-filename`
行がマッチしたファイル名も表示します。

`-z` または `--null-data`

入出力データストリームをNull文字で区切った部分ごとに処理します。このオプションを指定しないデフォルトでは、 改行 で区切った行ごとに処理します。（ファイル名に改行文字が含まれるかもしれないという理由などにより）`find` コマンドで `-print0` オプションを指定した出力を `grep` で処理する場合には、`-z` または `--null-data` オプションを指定するとうまくいきます。

複数のファイルが与えられるとデフォルトで `-H` オプションが指定されるのですが、一つずつファイルが与えられる場合にはデフォルトで `-H` オプションが指定されません。例えば、次のように、`grep` が `find` によって直接呼び出される場合に問題となります。

```
$ find /usr/share/doc -type f -exec grep -i '3d modeling' "{}" \; | cut -c -100
artistic aspects of 3D modeling. Thus this might be the application you are
This major approach of 3D modeling has not been supported
oce is a C++ 3D modeling library. It can be used to develop CAD/CAM softwares, for
instance [FreeCad
```

この例では、`find` が `/usr/share/doc` 以下のすべてのファイルを一つずつ `grep` に渡し、そのファイル内に `3d modeling` が含まれているかどうかを大文字と小文字を区別せずに検索します。100文字分だけ切り出すために `cut` へパイプでつなげています。しかし、マッチした行がどのファイルの行なのかがわかりません。`grep` に `-H` オプションを指定するとわかるようになります。

```
$ find /usr/share/doc -type f -exec grep -i -H '3d modeling' "{}" \; | cut -c -100
/usr/share/doc/openscad/README.md:artistic aspects of 3D modeling. Thus this might
be the applicatio
/usr/share/doc/opencsg/doc/publications.html:This major approach of 3D modeling has
not been support
```

これでマッチした行がどのファイルの行なのかがわかります。さらに情報を得るために、マッチした前後の行を表示させることもできます。

```
$ find /usr/share/doc -type f -exec grep -i -H -1 '3d modeling' "{}" \; | cut -c
-100
/usr/share/doc/openscad/README.md-application Blender), OpenSCAD focuses on the CAD
aspects rather t
/usr/share/doc/openscad/README.md:artistic aspects of 3D modeling. Thus this might
be the applicatio
/usr/share/doc/openscad/README.md-looking for when you are planning to create 3D
models of machine p
/usr/share/doc/opencsg/doc/publications.html-3D graphics library for Constructive
Solid Geometry (CS
```

```
/usr/share/doc/opencsg/doc/publications.html:This major approach of 3D modeling has
not been support
/usr/share/doc/opencsg/doc/publications.html-by real-time computer graphics until
recently.
```

`-C1` や `--context=1` というオプションを指定すると、マッチした前後の行を1行ずつ含めて結果を表示します。コンテキスト（文脈）を表示するということです。出力ではファイル名の直後に `-`（マイナス）がつけられます（マッチした行はファイル名の直後に `:`（コロン）がつけられます）。上の例のように `-1` という指定でも同じ結果になります。前の行と後の行の表示行数を別々に指定することもできます。

grepには2つの補完的なプログラムがあります。`egrep` と `fgrep` です。`egrep` は `grep -E` と同じで、パターンを指定する正規表現が基本正規表現から拡張正規表現に切り替わります。例えば、`egrep` では、マッチさせたい文字列を `|`（パイプ）で区切る選択が使えます。

```
$ find /usr/share/doc -type f -exec egrep -i -H -1 '3d (modeling|printing)' "{}" \;
| cut -c -100
/usr/share/doc/openscad/README.md-application Blender), OpenSCAD focuses on the CAD
aspects rather t
/usr/share/doc/openscad/README.md:artistic aspects of 3D modeling. Thus this might
be the applicatio
/usr/share/doc/openscad/README.md-looking for when you are planning to create 3D
models of machine p
/usr/share/doc/openscad/RELEASE_NOTES.md-* Support for using 3D-Mouse / Joystick /
Gamepad input dev
/usr/share/doc/openscad/RELEASE_NOTES.md:* 3D Printing support: Purchase from a
print service partne
/usr/share/doc/openscad/RELEASE_NOTES.md-* New export file formats: SVG, 3MF, AMF
/usr/share/doc/opencsg/doc/publications.html-3D graphics library for Constructive
Solid Geometry (CS
/usr/share/doc/opencsg/doc/publications.html:This major approach of 3D modeling has
not been support
/usr/share/doc/opencsg/doc/publications.html-by real-time computer graphics until
recently.
```

この例では、`3D modeling` または `3D printing` のどちらでもマッチします。`-i` オプションを指定しているので大文字と小文字を区別しません。`-o` オプションを指定するとテキストストリームのマッチした部分だけを表示できます。

`fgrep` は `grep -F` と同じで、正規表現の解析をしません。そのままの表現を単純にマッチしたいときに便利です。`$`（ドル記号）や `.`（ドット）などの特殊文字も正規表現での特殊な意味ではなくそのままの記号として解釈されます。

ストリームエディタ：sed

`sed` はテキストベースのデータを非対話的に修正するためのプログラムです。（通常のエディタのように）画面に表示されるテキストを直接思いのままにタイピングするのではなく、事前に定義した指示によって編集を行います。今時風に、テンプレートパーサーだと言ってもよいでしょう。あるテキストを入力として受け取り、事前に決められた場所や正規表現でマッチした部分をカスタムコンテンツに入れ替えるのです。

sedはStream EDitorの略であり、その名前が示すように、パイプを通過するストリーミングテキストの編集にうってつけです。編集指示をスクリプトファイルに保存している場合には `sed -f ▯▯▯▯▯▯▯▯▯▯▯▯`、コマンドラインから直接スクリプトを実行する場合には `sed -e ▯▯▯▯▯▯` とするのが基本です。`-f` も `-e` もなければ、最初の非オプションパラメータがスクリプトファイル名として用いられます。`sed` の引数にファイルパスを渡せばそのファイルの内容が `sed` の入力になります。

`sed` のスクリプト内のコマンドは1文字で表されます。オプションやアドレスの後にスクリプトを書き、それが各行に適用されます。アドレスは行番号、行範囲、正規表現のいずれかです。例えば、`1d` でテキストストリームの最初の行を削除できます。`1` が行番号を指定するアドレスで、`d` が削除を意味するスクリプトです。1から12までの素因数を返す `factor `seq 12`` の出力を利用して `sed` の使い方を見ていきましょう。

```
$ factor `seq 12`
1:
2: 2
3: 3
4: 2 2
5: 5
6: 2 3
7: 7
8: 2 2 2
9: 3 3
10: 2 5
11: 11
12: 2 2 3
```

`1d` で最初の行を削除します。

```
$ factor `seq 12` | sed 1d
2: 2
3: 3
4: 2 2
5: 5
6: 2 3
7: 7
```

```
8: 2 2 2
9: 3 3
10: 2 5
11: 11
12: 2 2 3
```

`,`（カンマ）で行範囲をアドレスに指定できます。

```
$ factor `seq 12` | sed 1,7d
8: 2 2 2
9: 3 3
10: 2 5
11: 11
12: 2 2 3
```

`;`（セミコロン）で区切って続けて書けば複数のスクリプトを実行できます。`;`（セミコロン）がシェルに解釈されないように `"`（ダブルクオート）で囲ってください。

```
$ factor `seq 12` | sed "1,7d;11d"
8: 2 2 2
9: 3 3
10: 2 5
12: 2 2 3
```

上の例では2つの削除スクリプトが実行されます。1行目から7行目までの削除と、11行目の削除です。アドレスは正規表現でも構いません。正規表現にマッチした行にだけスクリプトが適用されます。

```
$ factor `seq 12` | sed "1d;/:.*2.*/d"
3: 3
5: 5
7: 7
9: 3 3
11: 11
```

`:.*2.*` という正規表現は、`:`（コロン）より後のどこかに2があるとマッチするので、上の例は2を素因数に持つ数字の行を削除することになります。`sed` では、`/`（スラッシュ）の間に置かれたものが正規表現だとみなされます。デフォルトでは基本正規表現です。例えば、`sed -e "/^#/d" /etc/services` を実行すると、`#` で始まる行（コメント行）以外の `/etc/services` の内容が表示されます。

削除スクリプトの `d` 以外にも `sed` にはいろいろなスクリプトがあります。行置換スクリプトは次のとおりです。

```
$ factor `seq 12` | sed "1d;/:.*2.*/c REMOVED"
REMOVED
3: 3
REMOVED
5: 5
REMOVED
7: 7
REMOVED
9: 3 3
REMOVED
11: 11
REMOVED
```

スクリプト `c REMOVED` は、対象行を `REMOVED` というテキストに置換します。上の例では、正規表現 `:.*2.*` にマッチする各行に対して行置換スクリプト `c REMOVED` が実行されます。スクリプト `a TEXT` はマッチした行の次の行に `TEXT` というテキストを追加します。スクリプト `r FILE` も同様にテキストを追加するのですが、`FILE` で指定されたファイルの内容をコピーします。スクリプト `w` は `r` の反対のことをします。指定されたファイルに対してマッチした行を書き込みます。

`sed` で断トツによく使われるスクリプトは `s/FIND/REPLACE/` です。正規表現 `FIND` にマッチした部分をテキスト `REPLACE` に置換します。例えば、スクリプト `s/hda/sda/` は、正規表現にマッチした `hda` を `sda` に置換します。その行で最初にマッチした部分だけが置換されます。`s/hda/sda/g` のように `g` フラグを付け加えるとマッチした部分が全部置換されます。

もっと実践的な `sed` の使い方を見てみましょう。あるクリニックが患者に翌日の予約をリマインドするショートメッセージを送るという場面を想定しています。メッセージの送信は商用メッセージサービスが提供するAPIを通じて行うことになるでしょうし、こうしたメッセージの送信は患者の予約を管理しているアプリケーションから指定した時間やその他の何らかのイベントによってトリガーするのが一般的でしょうが、ここでは翌日の予約データからメッセージを組み立てる処理だけを扱います。以下のような翌日の予約データを表形式で格納した `appointments.csv` というファイルがあると仮定して（CSVファイルはデータベースからデータをエクスポートする際の標準的なファイル形式です）、`template.txt` というテンプレートファイルを基に `sed` で各患者に送るテキストメッセージを組み立てます。

```
$ cat appointments.csv
"NAME","TIME","PHONE"
"Carol","11am","55557777"
"Dave","2pm","33334444"
```

最初の行は各列のラベルになっています。そのラベルをテンプレートファイルに埋め込みます。

```
$ cat template.txt
Hey <NAME>, don't forget your appointment tomorrow at <TIME>.
```

タグとして使うために小なり記号 `<` と大なり記号 `>` をラベルの前後に入れています。以下のシェルスクリプトは、翌日の予約データを解析し、`template.txt` テンプレートファイルを基にメッセージを組み立て、商用メッセージサービスが提供するAPIへと送る準備をします。

```
#! /bin/bash

TEMPLATE=`cat template.txt`
TAGS=(`sed -ne '1s/^"//;1s/","/\n/g;1s/"$//p' appointments.csv`)
mapfile -t -s 1 ROWS < appointments.csv
for (( r = 0; r < ${#ROWS[*]}; r++ ))
do
  MSG=$TEMPLATE
  VALS=(`sed -e 's/^"//;s/","/\n/g;s/"$//' <<<${ROWS[$r]}`)
  for (( c = 0; c < ${#TAGS[*]}; c++ ))
  do
    MSG=`sed -e "s/<${TAGS[$c]}>/${VALS[$c]}/g" <<<"$MSG"`
  done
  echo curl --data message=\"$MSG\" --data phone=\"${VALS[2]}\"
https://mysmsprovider/api
done
```

実用レベルのスクリプトにするには、認証、エラーチェック、ログの記録などが求められますが、出発点となる基本的な機能は出揃っています。`sed` の最初のスクリプトは、`1s/^"//;1s/","/\n/g;1s/"$//p` に `1` というアドレスがあるので、1行目にだけ作用します。`1s/^"//` と `1s/"$//` で行頭と行末の `"`（ダブルクオート）を取り除き、`1s/","/\n/g` でフィールドの区切りを `","`（ダブルクオート、カンマ、ダブルクオート）から `\n`（改行）に置換しています。ここではカラム名が書かれた1行目だけがほしいので、マッチしなかった行が出力されないように `-n` オプションを指定しています。ただし、マッチした行を出力するために、`p` フラグが最後に必要となります。このようにして取得されたタグは、Bashの配列として変数 `TAGS` に格納されます。予約行が `mapfile` コマンドでBashの配列として変数 `ROWS` に格納されます。

`for` ループで `ROWS` 変数の各予約行の処理をしています。ヒア文字列として使われている変数 `${ROWS[$r]}` に格納されている予約の各行の行頭と行末の `"`（ダブルクオート）を取り除き、`","`（ダブルクオート、カンマ、ダブルクオート）を `\n`（改行）に置換しています。タグを取得するのに使ったコマンドとほとんど同じです。このようにして分離された値

が配列変数 `VALS` に格納されます。配列の添え字の0、1、2がそれぞれ `NAME`、`TIME`、`PHONE` に対応します。

最後に、入れ子になっている `for` ループで `TAGS` 配列を順番に処理して、テンプレート内のタグを `VALS` の対応する値に置換します。変数 `MSG` は生成されたテンプレートのコピーを保持します。`s/<${TAGS[$c]}>/${VALS[$c]}/g` コマンドで順次置換されて更新されていきます。

こうして `"Hey Carol, don't forget your appointment tomorrow at 11am."` のようなメッセージが生成されることになります。そのメッセージを `curl` コマンドでHTTPリクエストを通じてパラメータとして送り、ショートメールを送信できるというわけです（これは架空の例ですから、`curl` コマンドを実行するのではなく `echo` で表示するようにしています）。

grepとsedを組み合わせて使う

もっと複雑なテキストマイニングでは `grep` コマンドと `sed` コマンドを同時に使うこともあります。システム管理者としてサーバーへのログイン試行を精査したいとしましょう。`/var/log/wtmp` ファイルがすべてのログイン及びログアウトを記録しており、`/var/log/btmp` ファイルが失敗したログイン試行を記録しています。これらのファイルはバイナリ形式で書かれているため、それぞれ `last` コマンドと `lastb` コマンドで読み出します。

`lastb` は失敗したログイン試行のユーザー名とIPアドレスを出力します。

```
# lastb -d -a -n 10 --time-format notime
user     ssh:notty       (00:00)     81.161.63.251
nrostagn ssh:notty       (00:00)     vmd60532.contaboserver.net
pi       ssh:notty       (00:00)     132.red-88-20-39.staticip.rima-tde.net
pi       ssh:notty       (00:00)     132.red-88-20-39.staticip.rima-tde.net
pi       ssh:notty       (00:00)     46.6.11.56
pi       ssh:notty       (00:00)     46.6.11.56
nps      ssh:notty       (00:00)     vmd60532.contaboserver.net
narmadan ssh:notty       (00:00)     vmd60532.contaboserver.net
nominati ssh:notty       (00:00)     vmd60532.contaboserver.net
nominati ssh:notty       (00:00)     vmd60532.contaboserver.net
```

`-d` オプションはIPアドレスの数字をホスト名に変換して表示します。ホスト名は失敗したログイン試行をしたプロバイダーやホスティングサービスについての手がかりになるかもしれません。`-a` オプションはホスト名を最終列に表示するようにします。後でその最終列にフィルターを適用するので、そのための調整です。`--time-format notime` オプションは、ログイン試行の発生した日時を出力しないようにします。多くの失敗したログイン試行があると `lastb` コマンドは時間がかかることがあるので、`-n 10` オプションで出力を10件に限定しています。

すべてのリモートIPアドレスがホスト名と関連づけられているわけではないため、DNSの逆引きができないことがあり、その部分はそのままIPアドレスが表示されます。行末のホスト名にマッチする正規表現を書いてもよいのですが、おそらく行の最後の1文字が数字かアルファベットかを判定する正規表現を書くほうが簡単です。以下の例では、失敗したログイン試行のリストを `grep` コマンドが標準入力から受け取り、ホスト名を取得できなかった行を除去して表示します。

```
# lastb -d -a --time-format notime | grep -v '[0-9]$' | head -n 10
nvidia   ssh:notty       (00:00)     vmd60532.contaboserver.net
n_tonson ssh:notty       (00:00)     vmd60532.contaboserver.net
nrostagn ssh:notty       (00:00)     vmd60532.contaboserver.net
pi       ssh:notty       (00:00)     132.red-88-20-39.staticip.rima-tde.net
pi       ssh:notty       (00:00)     132.red-88-20-39.staticip.rima-tde.net
nps      ssh:notty       (00:00)     vmd60532.contaboserver.net
narmadan ssh:notty       (00:00)     vmd60532.contaboserver.net
nominati ssh:notty       (00:00)     vmd60532.contaboserver.net
nominati ssh:notty       (00:00)     vmd60532.contaboserver.net
nominati ssh:notty       (00:00)     vmd60532.contaboserver.net
```

`grep` コマンドに `-v` オプションを指定すると正規表現にマッチしなかった行だけを表示します。`[0-9]$` という正規表現は行末が数字である行にマッチするので、ホスト名を取得できなかった行を捕捉します。よって、`grep -v '[0-9]$'` は、ホスト名を取得できた行だけを表示します。

この出力に対してさらにフィルターを適用しましょう。各行のドメイン名だけを残し、その他の部分を除去します。`sed` コマンドの置換スクリプトの出番です。ドメイン名をグループ化して後方参照で行全体と置換します。

```
# lastb -d -a --time-format notime | grep -v '[0-9]$' | sed -e 's/.* \(.*\)$/\1/' |
head -n 10
vmd60532.contaboserver.net
vmd60532.contaboserver.net
vmd60532.contaboserver.net
132.red-88-20-39.staticip.rima-tde.net
132.red-88-20-39.staticip.rima-tde.net
vmd60532.contaboserver.net
vmd60532.contaboserver.net
vmd60532.contaboserver.net
vmd60532.contaboserver.net
vmd60532.contaboserver.net
```

`.* \(.*\)$` の `\(` と `\)` （エスケープされた丸かっこ）が行のドメイン名の部分、すなわち最後のスペースから行末までの部分をグループ化して記憶します。この例では、その記

憶された部分を `\1` で後方参照し、行全体と置換しています。

同じリモートホストが何度もログインしようとしているので、同じドメイン名が繰り返されています。繰り返しを表示しないようにするためには、（`sort` コマンドで）ソートしてから `uniq` コマンドに渡します。

```
# lastb -d -a --time-format notime | grep -v '[0-9]$' | sed -e 's/.* \(.*\)$/\1/' |
sort | uniq | head -n 10
116-25-254-113-on-nets.com
132.red-88-20-39.staticip.rima-tde.net
145-40-33-205.power-speed.at
tor.laquadrature.net
tor.momx.site
ua-83-226-233-154.bbcust.telenor.se
vmd38161.contaboserver.net
vmd60532.contaboserver.net
vmi488063.contaboserver.net
vmi515749.contaboserver.net
```

これはいろいろなコマンドを組み合わせて望む出力を得るにはどうしたらよいかを示す格好の例です。このホスト名のリストは、ファイアウォールの拒否ルールの設定その他のセキュリティ対策を実施する際に活用できます。

演習

1. `last` コマンドはログインしたユーザーの履歴をIPアドレスも含めて表示します。その `last` の出力を `egrep` コマンドでフィルタリングしてIPv4アドレスだけを表示するようにしてください。IPv4アドレス以外の情報は必要ありません。

2. `-print0` オプションを指定して実行された `find` コマンドの出力を正常にフィルタリングするには `grep` コマンドにどのオプションを指定すればよいでしょうか。

3. `uptime -s` コマンドは `2019-08-05 20:13:22` のように直近のシステム起動日時を表示します。`uptime -s | sed -e 's/(.*) (.*)/\1/'` コマンドの実行結果はどうなるでしょうか。

4. マッチした行を表示するのではなくマッチした行の総数を表示するには、`grep` にどのオプションを指定しますか。

発展演習

1. HTMLファイルの基本構造は、`html`、`head` 、`body` です。例えば次のようなファイルです。

```
<html>
<head>
  <title>News Site</title>
</head>
<body>
  <h1>Headline</h1>
  <p>Information of interest.</p>
</body>
</html>
```

   `body` 要素とその中身だけを表示する `sed` コマンドのアドレスを書いてください。

2. `sed` コマンドでHTML文書からタグをすべて取り除きテキストだけを表示してください。

3. `.ovpn` というサフィックスのファイルは、設定情報とともにクライアントの鍵と証明書の中身を含められるので、VPNクライアントを設定する際によく用いられます。鍵と証明書は元々別ファイルにあるので、その中身を `.ovpn` ファイルにコピーしなければなりません。以下のようなテンプレートを用意してその作業を楽にしましょう。

```
client
dev tun
remote 192.168.1.155 1194
<ca>
ca.crt
</ca>
<cert>
client.crt
</cert>
<key>
client.key
</key>
<tls-auth>
ta.key
</tls-auth>
```

`ca.crt`、`client.crt`、`client.key`、`ta.key`　ファイルがカレントディレクトリにあるとします。`sed`　コマンドで上記のテンプレートファイル（`client.template`）の該当箇所を各ファイルの内容に置換して、`client.ovpn` ファイルを作成してください。

まとめ

このレッスンでは正規表現に関連する2つの重要なLinuxコマンドを説明しました。`grep` と `sed` です。これらのコマンドとさまざまなコマンドを組み合わせて、テキストの抽出や解析を行うシェルスクリプトを書くことができます。このレッスンは以下の順で取り上げました。

- `grep` とその派生コマンドである `egrep` や `fgrep` の使い方
- `sed` のスクリプトでテキストを処理する方法
- `grep` と `sed` を使った正規表現の応用例

演習の解答

1. `last` コマンドはログインしたユーザーの履歴をIPアドレスも含めて表示します。その `last` の出力を `egrep` コマンドでフィルタリングしてIPv4アドレスだけを表示するようにしてください。IPv4アドレス以外の情報は必要ありません。

   ```
   last -i | egrep -o '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}'
   ```

2. `-print0` オプションを指定して実行された `find` コマンドの出力を正常にフィルタリングするには `grep` コマンドにどのオプションを指定すればよいでしょうか。

   `-z` オプションまたは `--null-data` オプションです。例えば `find . -print0 | grep -z` のようにします。

3. `uptime -s` コマンドは `2019-08-05 20:13:22` のように直近のシステム起動日時を表示します。`uptime -s | sed -e 's/(.*) (.*)/\1/'` コマンドの実行結果はどうなるでしょうか。

   エラーが発生します。`sed` はデフォルトでは基本正規表現を使いますから、`(` と `)`（丸かっこ）をエスケープしなければなりません（訳注：`-r` オプションで拡張正規表現を使用することもできます）。

4. マッチした行を表示するのではなくマッチした行の総数を表示するには、`grep` にどのオプションを指定しますか。

   `-c` オプションを指定します。

発展演習の解答

1. HTMLファイルの基本構造は、`html`、`head` 、`body` です。例えば次のようなファイルです。

```
<html>
<head>
  <title>News Site</title>
</head>
<body>
  <h1>Headline</h1>
  <p>Information of interest.</p>
</body>
</html>
```

   `body` 要素とその中身だけを表示する `sed` コマンドのアドレスを書いてください。

   `body` 要素とその中身だけを表示するアドレスは `/<body>/,/<\/body>/` です。`sed -n -e '/<body>/,/<\/body>/p'` のように実行します。`-n` オプションで行の表示をしないようにして、マッチした行を表示する `p` スクリプトを実行しています（訳注：`1,7` で1行目から7行目までの行範囲を指定するのと同じように、`/<body>/,/<\/body>/` で `<body>` の行から `</body>` の行までの範囲を正規表現で指定しています）。

2. `sed` コマンドでHTML文書からタグをすべて取り除きテキストだけを表示してください。

   `sed` で `s/<[^>]*>//g` と書くと、`<>` で囲われた部分を空の文字列に置換します。

3. `.ovpn` というサフィックスのファイルは、設定情報とともにクライアントの鍵と証明書の中身を含められるので、VPNクライアントを設定する際によく用いられます。鍵と証明書は元々別ファイルにあるので、その中身を `.ovpn` ファイルにコピーしなければなりません。以下のようなテンプレートを用意してその作業を楽にしましょう。

```
client
dev tun
remote 192.168.1.155 1194
<ca>
ca.crt
</ca>
<cert>
client.crt
</cert>
<key>
```

```
client.key
</key>
<tls-auth>
ta.key
</tls-auth>
```

ca.crt、client.crt、client.key、ta.key ファイルがカレントディレクトリにあるとします。sed コマンドで上記のテンプレートファイル（client.template）の該当箇所を各ファイルの内容に置換して、client.ovpn ファイルを作成してください。

```
sed -r -e 's/(^[^.]*)\.(crt|key)$/cat \1.\2/e' < client.template > client.ovpn
```

このコマンドは、client.template ファイルを入力として受け取り、行末が .crt または .key の行をその行と同じ名前のファイルの中身に置き換えて、client.ovpn ファイルに出力します。-r オプションを指定して拡張正規表現を使用します。正規表現の最後の e は cat コマンドの実行結果で置換するという意味です。後方参照の \1 と \2 は、マッチしたファイル名とサフィックスにそれぞれ対応します。

103.8 ファイルの基本的な編集

LPI目標への参照

LPIC-1 v5, Exam 101, Objective 103.8

総重量

3

主な知識分野

- viを使用してドキュメントをナビゲートする。
- 基本的なviモードを使用する。
- viにおける、テキストの挿入、編集、削除、コピー、検索。
- emacs, nano vimなどの知識。
- エディタの設定。

用語とユーティリティ

- `vi`
- `/`, `?`
- `h`, `j`, `k`, `l`
- `i`, `o`, `a`
- `d`, `p`, `y`, `dd`, `yy`
- `ZZ`, `:w!`, `:q!`
- 環境変数EDITOR

103.8 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	103 GNUおよびUnixコマンド
Objective:	103.8 基本的なファイル編集
Lesson:	1 of 1

はじめに

ほとんどのLinuxディストリビューションでは、`vi`（visualの略です）があらかじめインストールされており、シェル環境での標準エディタになっています。viは対話的なテキストエディタであり、編集中のファイルの内容を画面に表示して、文書中のどこでも自由に移動して変更できます。`vi` はグラフィカルなデスクトップ環境で動作するビジュアルエディタではなく、シェル環境で用いるターミナルで動作するアプリケーションで、編集作業にキーボードショートカットを使います。

最近では `vi` の代わりに `vim`（vi improvedの略です）が使われることがほとんどです（訳注：`vi` というコマンドで実際には `vim` が呼び出されます）。`vim` はシンタックスハイライト、多段階の「アンドゥ（元に戻す）」と「リドゥ（やり直し）」や、複数ファイルの編集をサポートしています。多機能になっていても `vim` は `vi` と完全に後方互換性があり、ほとんどの作業はどちらでも同じように行えます。

`vi` はパラメータにファイルパスを指定して起動するのが一般的です。`` `□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□`vi +9 /etc/fstab` □□□□□□□□□□□□□□□□□9□□□□□□□□□ `/etc/fstab` □□□□□□□□□□□□□□□□ □□□□□□ ` ``（プラス）だけだとカーソルが最終行にある状態で開きます。

`vi` のインターフェイスはとてもシンプルです。パラメータに指定したファイルの内容がターミナルウィンドウの全面に表示されます。ファイルの内容以外では、カーソルの現在位置などを示すフッター行（最下行）と、画面サイズよりもファイルの内容が短いことを示す `~`

（チルダ）だけが表示されます。`vi` にはいくつかのモードがあり、モードに応じて動作が変わります。入力モードとノーマルモードの2つをよく使います。

入力モード

入力モードでは、読んで字のごとく、キーボードで打ち込んだ文字が画面に入力されて表示されます。テキストエディタならそれが当たり前だと思われるかもしれませんが、実は `vi` がファイルを開いた直後のモードは入力モードではありません。ノーマルモードが `vi` のデフォルトのモードなのです。入力モードに入るにはノーマルモードで入力コマンドを実行しなければならないのです。`Esc` キーを押すと入力モードからノーマルモードに戻ります。

NOTE

他のモードについてもっと知りたければ、`vi` を起動してから次のように打ち込んでみてください。

```
:help vim-modes-intro
```

ノーマルモード

ノーマルモードは、コマンドモードと呼ぶこともありますが、`vi` が起動したときのデフォルトのモードです。このモードでは、キーボードのそれぞれのキーを押すと、（その文字が画面に入力されて表示されるのではなく）画面を移動したり文字を操作したりするコマンドが実行されます。各コマンドがどれかのキーに割り当てられています（同じコマンドが複数の種類のキーに割り当てられていることもあります）。そのうちのいくつかを示します（訳注：日本語は文やワードの概念が英語とは異なるため、文やワードに関係するコマンドを実行する際には注意してください）。

`0`、`$`
: それぞれ行頭あるいは行末に移動します。

`1G`、`G`
: それぞれファイルの先頭あるいは末尾に移動します。

`(`、`)`
: それぞれ文の先頭あるいは末尾に移動します。

`{`、`}`
: それぞれ段落（空行で区切られたかたまり）の先頭あるいは末尾に移動します。

`w`、`W`
: 次のワードに移動します。`W` は句読記号もワードに含めて移動します。

h、j、k、l
それぞれ、左、下、上、右に移動します。

e、E
ワードの末尾に移動します。E は句読記号もワードに含めて移動します。

/、?
/ は下方向に、? は上方向に、それぞれ続けて入力した文字列を検索します。

i、I
それぞれ現在のカーソル位置あるいは行頭で入力モードに入ります。

a、A
それぞれ現在のカーソル位置の後あるいは行末で入力モードに入ります。

o、O
それぞれ現在行の後あるいは前に空行を加えて、そこで入力モードに入ります。

s、S
それぞれ現在のカーソル位置の文字あるいはカーソルがある行全体を消してから入力モードに入ります。

c
対象を指定して変更します。例えば、cw でカーソル位置からワードの終わりまでを、cb でワードの始まりからカーソル位置までを、cc でカーソルがある行全体を、それぞれ変更します。

r
カーソル位置の文字を、続いて入力した1文字で置き換えて、ノーマルモードに戻ります。

x
選択した文字またはカーソル位置の文字を削除します。

v、V
ビジュアルモードに入り、それぞれ現在カーソルがある文字あるいは行から選択を開始します。G、w、h、j、k、l、e などのカーソル移動コマンドで、選択範囲を調整できます。

y、yy
対象を指定してコピー（ヤンク）します。yw でカーソル位置からワードの終わりまでを、y$ でカーソル位置から行末までを、yy でカーソルがある行全体をコピーします

（訳注：いわゆるクリップボードのことを、vi では「ヤンクバッファ」と呼びます。明示的にヤンクした時だけではなく、x コマンドや d コマンドで削除した文字列や行も、自動的にヤンクバッファに格納されます。格納された文字列ないし行によって、現在の行にペーストする「文字」モードと新しい行にペーストする「行」モードがあることに注意してください）。

p、P
それぞれ現在のカーソル位置の後あるいは前に、コピーした（ヤンクバッファの）内容をペーストします。

.（ピリオド）
現在のカーソル位置で、直前に実行したテキスト編集コマンドを繰り返します。挿入モードに入った瞬間からノーマルモードに戻るまでの間の操作も一つのテキスト編集コマンドだとみなされますから、例えば、A; と順に押して行末に ;（セミコロン）を入力してから Esc キーを押してノーマルモードに戻ると、カーソルを移動させて .（ピリオド）を押すだけでその行末に ;（セミコロン）を入れることができるようになります。

u
直前のテキスト編集コマンドのアンドゥ（元に戻す）。複数回繰り返すことができます。

Ctrl-R
直前のアンドゥ（元に戻す）のリドゥ（やり直し）。

ZZ
保存して終了します（訳注：確認せずに終了するので、通常はコロンコマンド :wq を使うのがお勧めです）。

ZQ
保存せずに終了します（訳注：確認せずに終了するので、通常はコロンコマンド :wq! を使うのがお勧めです）。

コマンドの前に数値を入力すると、その回数分だけそのコマンドを繰り返し実行します。例えば、3yy と押すと現在の行から3行分コピーしますし、d5w（5dw）と押すと現在のワードから5ワード分削除するといった具合です。

vi ではひとつのアクションを実行するために、複数のコマンドを連続して実行することが少なくありません。例えば、カーソル位置からワードの終わりまでを選択してコピーするには、vey と入力します。すなわち、v でビジュアルモードに入り、e で単語の末尾まで移動し（ビジュアルモードなので選択されます）、y で選択部分をヤンクしてビジュアルモードを終えます。3ワードを選択したければ、v3ey と入力して、e コマンドを3回繰り返します（訳注：より簡単には（ないしは vi らしくは）、3dw コマンドで3ワードを削除してから u で削除を元に戻します。アンドゥしてもヤンクバッファの内容は変わりません）。

vi ではコピーや削除したテキストをヤンクバッファではなく、a〜z の名前で区別される レジスタ に格納することができます。レジスタを使えば、複数の異なるテキストや行を、一時的に保持することができます。レジスタは "X （Xはa-zのいずれか）で指定します。たとえば "lyy というキーを順番に押すと、レジスタ l に（"l）、現在カーソルがある行を格納（yy）します。同様に、"lp というキーを順番に押すと、レジスタ l の内容をペーストします。

テキスト中の任意の場所にマークをつけて、後のその位置に素早く戻ることができます。m キーを押してから何らかのキー（a-zとA-Zが使えます）を押すと、そのキーと現在のカーソル位置が結びつけられます。`（バッククオート）を押してから先ほど結びつけたキーを押すとマークの位置に、'（シングルクオート）を押してから先ほど結びつけたキーを押すとマークをつけた行の先頭に、それぞれカーソルが戻ります。

マクロとしてキーの組み合わせを記録することもできます。例えば、選択したテキストを "（ダブルクオート）で囲むマクロを記録してみましょう。まず、テキストを選択し、q を押します。次にマクロと関連付けるキーを押します。ここでは d にしましょう。そうするとフッター行に recording @d と表示されます。これでマクロの記録中だとわかります。すでにテキストは選択されているので、記録する最初のコマンドは選択したテキストを切り取る x です。次に i を押してから2個の " を現在の位置に入力します。それからEscでノーマルモードに戻ります。最後に P を押して、先ほど切り取った選択テキストを2個の " の間にペーストします。もう一度 q を押すとマクロの記録が終了します。これで x、i、""、Esc、P という一連のキーからなるコマンドを実行するマクロの記録ができました。ノーマルモードで @d を押すと、何回でもこの一連のコマンドを実行できます。ここではマクロと関連付けるキーを d にしたので、@d でマクロを実行します。

マクロは現在のセッションでのみ有効です。セッションが終了してもマクロを使えるようにするには設定ファイルに保存します。最近のディストリビューションは vi 互換エディタとして vim を使っていることがほとんどなので、ユーザーごとの設定ファイルは ~/.vimrc です。vi ~/.vimrc でその設定ファイルを開き、let @d = 'xi""^[P' という行を書いてください。^[の部分は、^[（キャレット、角かっこ）とキーボードから打ち込むのではなく、Ctrl+V キーを押してからEscキーを押します。~/.vimrc ファイルを保存してから vi を起動すると、@d で '（シングルクオート）内の一連のコマンド（xi""^[P）を実行できるようになります。先ほどのマクロを記録した後にはレジスタ d に一連のコマンドが保存されていますから（vi のノーマルモードで :register を実行するとレジスタを確認できます）、~/.vimrc を開いている vi のノーマルモードで "dp と順に押して '（シングルクオート）内にその一連のコマンドをペーストしてもよいです。

コロンコマンド

ノーマルモードからは別種の vi のコマンドであるコロンコマンドを使うこともできます。その名のとおり、ノーマルモードで:（コロン）キーを押してからコマンドを実行します。コロンコマンドでは、検索、保存、終了ができ、シェルコマンドを実行したり vi の設定を変更したりすることもできます。よく使うコロンコマンドは次のとおりです（最初の:（コロン）はコロンコマンドであることを示しています）。

`:s/REGEX/TEXT/g`

現在の行で正規表現 `REGEX` にマッチした部分をすべて `TEXT` に置換します。アドレスを指定すれば現在の行以外も置換できます。`sed` コマンドと同じ書き方です（訳注：`:1,10s/REGEX/TEXT/g` で1行目から10行目のマッチした部分を、`:%s/REGEX/TEXT/g` でファイル全体のマッチした部分を、それぞれすべて `TEXT` に置換します）。

`:!`

シェルコマンドを実行します。例えば、`:!ls` を実行するとディレクトリの一覧が表示されます。

`:quit` または `:q`

終了します。

`:quit!` または `:q!`

保存せずに終了します。

`:wq`

保存して終了します。

`:exit` または `:x` または `:e`

テキストが変更されていれば保存して終了します。

`vi` でおよそどのようなテキスト編集作業もできますが、シェル環境で使えるエディタは他にもあります（次の節でいくつか紹介します）。

TIP

`vi` を使い始めた人はコマンドキーを覚えるのに苦労すると思います。`vim` が使えるディストリビューションでは `vimtutor` も使えるはずなので、これをやってみることをお勧めします。`vimtutor` コマンドは `vim` を使って一歩ずつ主な編集作業を体験するガイドです。編集用ファイルでコマンドを練習して少しずつ慣れていくように設計されています。

シェル環境で使える他のエディタ

操作が直感的ではないため `vi` に適応するのは難しいと感じる人もいるかもしれません。そういう人向けにGNU `nano` というもっとシンプルなシェル環境で使えるエディタがあります。「アンドゥ（元に戻す）」と「リドゥ（やり直し）」、シンタックスハイライト、対話的な検索と置換、自動インデント、行番号表示、ワード補完、ファイルのロックとバックアップ、多言語対応などの基本的な機能が備わっています。`vi`（のノーマルモード）とは違ってキーを押したら編集中の文書に入力されます。`nano` ではコマンドを`Ctrl`キーか`▯`キーを押して実行します（訳注：`▯`キーを「メタキー」と呼ぶこともあります。`Meta-` と表記することが多いです。なお、ターミナルエミュレーターを使っている場合は、デスクトップ環境や日本語入力などのショートカットとバッティングするために使用できないことがあります。画面下部に表示されているヘルプを参考にしながら、ファンクションキーや`Ctrl`キーを使うコマンドを覚えるとよいでしょう）。

Ctrl-6 または Meta-A
選択を開始します。シフトキーを押しながら矢印キーで選択することもできます。

Meta-6
現在選択している範囲をコピーします。

Ctrl-K
現在選択している範囲を削除してカットバッファに格納します。

Ctrl-U
カットバッファの内容をカーソル位置にペーストします。

Meta-U
直前の操作のアンドゥ（元に戻す）。

Meta-E
直前のアンドゥ（元に戻す）のリドゥ（やり直し）。

Ctrl-
テキストを検索して置換します。

Ctrl-T
スペルチェックを行います。

Ctrl-G
ヘルプ画面を表示します。`q` の入力でヘルプ画面が閉じます。

シェル環境で使えるテキストエディタには、Emacsというものもあります。基本的には `nano` と同じようにキーをタイプすると入力されていくのですが、`vi` のようにキーボードコマンドで文書中を移動することもできます。Emacsは（独自のLisp言語によって）機能を拡張することができるので、テキストエディタというよりもIDE（integrated development environment（統合開発環境））と捉えたほうがよいかもしれません。プログラムをコンパイルして実行したり、メールクライアントやRSSリーダーとして使う拡張モジュールがたくさん公開されていますから、Emacsだけで日常業務ができると言っても過言ではありません。

シェルコマンドがテキストエディタ（デフォルトでは `vi`）を呼び出すこともあります。たとえば、cronjobs を編集するために `crontab -e` を実行したときなどです。Bashでは `VISUAL` や `EDITOR` という名前の環境変数にデフォルトのテキストエディタを設定します。例えば、`export EDITOR=nano` というコマンドを実行すると、現在のシェルセッションのデフォルトテキストエディタが `nano` になります。`~/.bash_profile` に `export EDITOR=nano` と記述すれば、セッションが終了してもこの設定が有効になります。

演習

1. `vi` がよく使われるのは、設定ファイルやソースコードを編集する場面であり、インデントをするとテキストのまとまりがわかりやすくなります。`<<` を押すと左に、`>>` を押すと右に、それぞれ現在の行をインデントできます。ノーマルモードで現在の行を3段階右へインデントしてください。

2. `vi` のノーマルモードで `V` を押すと行全体を選択できます。このとき改行文字も選択されてしまいます。改行文字を含まない行全体を選択してください。

3. `vi` で `~/.bash_profile` ファイルをカーソルが最終行にある状態で開いてください。

4. `vi` のノーマルモードで現在のカーソル位置から次の `.`（ピリオド）までの文字を削除してください。

発展演習

1. `vim` では行に関係なく任意の幅でテキストのブロック（矩形領域）を選択できます。ノーマルモードで`Ctrl`+`V`と押して選択を開始してからカーソルを上下左右に動かします。現在の行の最初の文字から8文字×5行のブロックを削除してください。

2. `vi` のセッションが予期せぬ電源喪失により中断されたため、もう一度その編集中だったファイルを開いたところ、スワップファイル（`vi` によって自動的に作成されたコピー）を回復するかどうかを尋ねるプロンプトが表示されました。スワップファイルを回復せずに破棄する場合はどうすればよいでしょうか。

3. `vim` の現在のセッションでレジスタ `l` に何らかの行がコピーされているとします。レジスタ `l` にコピーされている行を現在の行の前にペーストするマクロをレジスタ `a` に記録してください。

まとめ

このレッスンではLinuxのシェル環境の標準的なテキストエディタである `vi` を説明しました。慣れていない人にとってはとっつきにくくはあるのですが、テキストを編集する際に便利な機能が備わっています。このレッスンは以下の順で取り上げました。

- `vi` の基本的な使い方と便利な機能
- `vi` の改良版である `vim` と、シェル環境で使えるその他のエディタ
- シェル環境でのデフォルトのテキストエディタの設定方法

以下のコマンドと手順を取り上げました:

- `vi` とその改良版である `vim`
- `vi` での基本的なテキスト編集
- `nano` と `emacs`

演習の解答

1. `vi` がよく使われるのは、設定ファイルやソースコードを編集する場面であり、インデントをするとテキストのまとまりがわかりやすくなります。`<<` を押すと左に、`>>` を押すと右に、それぞれ現在の行をインデントできます。ノーマルモードで現在の行を3段階右へインデントしてください。

   `>>>>>>` と `>` を6回入力します。`3>>` を実行すると、現在の行を3段階ではなく、現在の行から3行を1段階右へインデントします。

2. `vi` のノーマルモードで `V` を押すと行全体を選択できます。このとき改行文字も選択されてしまいます。改行文字を含まない行全体を選択してください。

   `0v$h` というキーを順に押します。`0` で行頭に移動し、`v` で選択を開始して、`$` で行末に移動し、`h` で1文字分左に戻るということです。

3. `vi` で `~/.bash_profile` ファイルをカーソルが最終行にある状態で開いてください。

   `vi + ~/.bash_profile` を実行します。

4. `vi` のノーマルモードで現在のカーソル位置から次の `.`（ピリオド）までの文字を削除してください。

   `dt.` というキーを順に押します。`d` で削除を開始し、`t.` で `.`（ピリオド）までを削除します。

発展演習の解答

1. vim では行に関係なく任意の幅でテキストのブロック（矩形領域）を選択できます。ノーマルモードで Ctrl+V と押して選択を開始してからカーソルを上下左右に動かします。現在の行の最初の文字から8文字×5行のブロックを削除してください。

   0（行頭へ）、Ctrl–V（選択開始）、8l5j（カーソル移動）、d（削除）と順に押します。

2. vi のセッションが予期せぬ電源喪失により中断されたため、もう一度その編集中だったファイルを開いたところ、スワップファイル（vi によって自動的に作成されたコピー）を回復するかどうかを尋ねるプロンプトが表示されました。スワップファイルを回復せずに破棄する場合はどうすればよいでしょうか。

   プロンプトが表示されたら d を押します。

3. vim の現在のセッションでレジスタ l に何らかの行がコピーされているとします。レジスタ l にコピーされている行を現在の行の前にペーストするマクロをレジスタ a に記録してください。

   qa"lPq というキーを順に押します。qa でレジスタ a にマクロの記録を開始し、"l でレジスタ l にコピーされているテキストを選び、P で現在の行の前にペーストし、q でマクロの記録を終了します。

Linux
Professional
Institute

課題 104: デバイス、Linuxファイルシステム、ファイルシステム階層標準

104.1 パーティションとファイルシステムを作成する

LPI目標への参照

LPIC-1 v5, Exam 101, Objective 104.1

総重量

2

主な知識分野

- MBRパーティションテーブル、GPTパーティションテーブルを管理する。
- さまざまなmkfsコマンドを使用して、次のようなさまざまなファイルシステムを作成します:
  - ext2/ext3/ext4
  - XFS
  - VFAT
  - exFAT
- マルチデバイスファイルシステム、圧縮、サブボリュームを含むBtrfsの基本的な知識。

用語とユーティリティ

- `fdisk`
- `gdisk`
- `parted`
- `mkfs`
- `mkswap`

104.1 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	104 デバイス、Linuxファイルシステム、ファイルシステム階層標準
Objective:	104.1 パーティションとファイルシステムを作成する
Lesson:	1 of 1

はじめに

どのOSでもディスクを使用するには、まずパーティションに分割します。パーティションとは、1つの物理的なディスクを論理的に分割して、OSからは複数台のディスクのように見せる仕組みです。物理的なディスクをどのように分割したかを示す情報が、物理ディスクの先頭にあるパーティションテーブルと呼ばれる領域に記録されます。パーティションテーブルには、開始セクター、終了セクター、パーティションタイプ、その他の詳細情報が記録されています。

通常、各パーティションは、同じ物理ディスクにある場合でも、OSからは別々の「ディスク」であるように見えます。Windowsでは各パーティションに `C:`（歴史的な理由からこれがメインディスクです）や `D:` のような文字が割り当てられます。Linuxでは各パーティションに `/dev/sda1` や `/dev/sda2` といった名前の `▯▯▯▯▯▯▯▯▯▯▯▯▯` が割り当てられます。

このレッスンでは、`fdisk`、`gdisk`、`parted` という3つのユーティリティを使ってパーティションを作成・削除・復元・リサイズすること、パーティションにファイルシステムを作成すること、仮想メモリとして利用される スワップパーティション ないし スワップファイル を作成することを学びます。

NOTE SSDやフラッシュストレージのような最近のストレージには「ディスク（円

> 盤状の部品）」が存在しないのですが、歴史的な理由から、このレッスンを通じて、ストレージメディアを「ディスク」と表記します。

パーティションテーブル〜MBRとGPT

ディスクにパーティション情報を保存する2つの方式があります。MBR（Master Boot Record）とGPT（GUID Partition Table）です。

MBR

MS-DOS（より正確には1983年のPC-DOS 2.0）という初期の時代から採用されてきた方式で、数十年間にわたりPCの標準的なパーティション形式でした。ディスクの最初のセクターがパーティションテーブルになり、その直後に ブートセクター とブートローダー（Linuxでは通例 GRUB ブートローダー）が書き込まれます。2TB以上のディスクに対応できない、1つのディスクに4つまでしか基本パーティションを作成できないといったいった制限があり、最近のシステムでは使われなくなりつつあります（訳注：物理的な新しいマシンでは使われなくなっていますが、手軽なので小規模な仮想マシンでは今もよく使われています）。

GPT

MBRの制約を解消するために規定されたパーティション方式（GUID Parittion Table）です。ディスクサイズに実際上の制限はありませんし、作成できるパーティションの数もOSによる制限しか受けません。UEFI（Unified Extensible Firmware Interface）と呼ばれる、旧来のBIOSを置き換える新しいファームウェア仕様の一部であり、新しいマシンはほぼすべてがこちらのパーティション方式を採用しています。

システム管理の仕事をするならきっとMBRとGPTの両方に出くわすことでしょう。ですから、それぞれの方式でパーティションを作成・削除・変更するツールを使いこなせるようになることが肝要です。

FDISKでMBRパーティションを管理する

LinuxでMBRパーティションを管理する標準的なユーティリティは `fdisk` です。メニューに沿って対話的に使います。`fdisk` に続けて編集するディスクに対応するデバイス名を入力して実行します。例えばこのように実行します。

```
# fdisk /dev/sda
```

このコマンドを実行すると、SATA接続の最初のデバイス（`sda`）のパーティションテーブルを編集することになります。`/dev/sda1` のようなパーティションではなく、`/dev/sda` のような物理ディスクに対応するデバイスを指定することに注意してください。

NOTE このレッスン中のディスクに関連する操作はすべて `root` ユーザー（システム管理者）として実行するか、`sudo` を使ってroot権限で実行します。

`fdisk` コマンドを実行すると、ウェルカムメッセージと警告が表示され、コマンドの入力待ち状態になります。

```
# fdisk /dev/sda
Welcome to fdisk (util-linux 2.33.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help):
```

ここで表示される警告には重要なことが書かれています。パーティションを作成・編集・削除しても `w`（write）コマンドを実行するまではディスクへの書き込みは行われません。`w` キーさえ押さなければデータを失うことを心配せずに「練習」できます。変更を保存せずに終了するには `q` コマンドを実行します。

NOTE

`w` キーさえ押さなければデータを失うことを心配せずに「練習」できるとはいえ、誤って `w` キーを押してしまうリスクが常につきまといますから、重要なディスクで練習することは絶対にやめてください。練習には不要なUSBフラッシュドライブなどの外部ディスクを使ってください（訳注：外部ディスクを使って練習する際にはデバイス名を間違えないように何度も確認してください。また、VirtualBoxやKVMなどの仮想マシンを作ってその中で練習してもよいです）。

現在のパーティションテーブルを表示する

`p`（print）コマンドで現在のパーティションテーブルの情報を表示できます。一例を挙げます。

```
Command (m for help): p
Disk /dev/sda: 111.8 GiB, 120034123776 bytes, 234441648 sectors
Disk model: CT120BX500SSD1
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x97f8fef5

Device     Boot     Start       End   Sectors   Size Id Type
/dev/sda1             4096 226048942 226044847 107.8G 83 Linux
/dev/sda2       226048944 234437550   8388607     4G 82 Linux swap / Solaris
```

各列の意味は次のとおりです。

`Device`
パーティションに割り当てられているデバイスです。

`Boot`
パーティションに「ブート可能フラグ」がセットされているか否かが表示されます。

`Start`
パーティションの開始セクター番号です。

`End`
パーティションの終了セクター番号です。

`Sectors`
パーティションのセクター総数です。これにセクターサイズ（上部のSector　sizeに表示されていて、この例では512バイト）を掛けるとパーティションのサイズをバイト単位で求めることができます。

`Size`
人間に読みやすい形でパーティションサイズが表示されます。上の例ではギガバイト単位で表示されています。

`Id`
パーティションタイプを表す数値（コード）です。

`Type`
パーティションタイプです。

基本パーティションと論理パーティション

MBRでは4つまでの　基本パーティション　を作れますが、ディスクの大容量化などからすぐに4つでは足りないことがわかりました。互換性を保ったまま、より多くのパーティションを扱えるように　拡張パーティション　が規定されて、より多くのパーティションを扱えるようになりました。ただし、多くのOSは基本パーティションからしかブートできません。

拡張パーティション　は、基本パーティションの1つをさらに分割して、複数の　論理パーティション　　を閉じ込めたものです。拡張パーティション　　はいわば「容器」であり、それ自体をディスクとして使用する事はできません。拡張パーティション　　に含むことができる 論理パーティション の数は、OSによって異なります。

Linuxでは、基本パーティションと論理パーティションを全く同じように扱えますから、拡張パーティションを作成したからといって不都合はありません。

パーティションを作成する

パーティションを作成するには、`fdisk` の `n` コマンドを使用します。状況に応じて、`p`・`e`・`l` のどのパーティションを作成するかが問い合わせられて、その後パーティションの開始セクターと終了セクターが問い合わせられます。

開始セクターには、未割り当て領域の先頭セクター番号がデフォルトで入力されていますから、特段の事情がなければデフォルトのままでよいでしょう。終了セクターを指定する代わりに K、M、G、T、P（キロ、メガ、ギガ、テラ、ペタ）の文字を使ってサイズを指定することもできます。1GBのパーティションを作成したいとすれば、終了セクターに `+1G` と指定します。次の例を見てください。

```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-3903577, default 2048): 2048
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-3903577, default 3903577): +1G
```

未割り当て領域を確認する

ディスクの未割り当て領域を調べるには、`F` コマンドを使用します。拡張パーティションがあれば、その中の未割り当て領域も表示されます。次に一例を示します。

```
Command (m for help): F
Unpartitioned space /dev/sdd: 881 MiB, 923841536 bytes, 1804378 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes

  Start     End Sectors  Size
2099200 3903577 1804378  881M
```

パーティションを削除する

パーティションを削除するには、`d` コマンドを使用します。複数のパーティションがあれば、削除するパーティション番号の入力を求められます。1つしかパーティションがない場合にはパーティション番号の入力を求められることなく 即座に削除されます。

拡張パーティションを削除すると、その中にある論理パーティションも全部削除されることに注意してください。

隙間に注意

新しいパーティションは、ディスクの　連続した　未割り当て領域に作成するものであることに気をつけてください。例えば、以下のようなパーティションマップだとしましょう。

```
Device     Boot   Start     End Sectors  Size Id Type
/dev/sdd1          2048 1050623 1048576  512M 83 Linux
/dev/sdd2       1050624 2099199 1048576  512M 83 Linux
/dev/sdd3       2099200 3147775 1048576  512M 83 Linux
```

ここでパーティション2を削除してから、空き領域を調べてみます。

```
Command (m for help): F
Unpartitioned space /dev/sdd: 881 MiB, 923841536 bytes, 1804378 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes

  Start     End Sectors  Size
1050624 2099199 1048576  512M
3147776 3903577  755802  369M
```

未割り当て領域は合計で881MBですが、連続する　　領域は512MBしかありませんから、700MBのパーティションを作ろうとしても次のように失敗します。

```
Command (m for help): n
Partition type
   p   primary (2 primary, 0 extended, 2 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (2,4, default 2): 2
First sector (1050624-3903577, default 1050624):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (1050624-2099199, default 2099199):
+700M
Value out of range.
```

一般的に、4個以上のパーティションを作成したい場合には、1〜3番目のパーティションを基本パーティションとして作成し、4番目を拡張パーティションとした上で、5番目以降を論理パーティションとして作成します。

パーティションタイプを変更する

ここまで説明してきませんでしたが、それぞれのパーティションは、用途（OS種別や後述のファイルシステム種別）に応じた　パーティションタイプ　というパラメータを持っています。Linuxの　`fdisk`　コマンドで作成したパーティションには、Linux用のパーティションであることを示す16進数で83という番号のパーティションタイプが、デフォルトで割り当てられています。ただし、Linuxではパーティションタイプにシステム管理上の単なるメモ以上の意味はありません。

利用可能なパーティションタイプを確認するには、`fdisk`　の　`l`　コマンドを使用します。パーティションタイプのコード（16進数）を確認したら、`t`　コマンドを入力し、タイプを変更するパーティション番号を指示してから、コードを入力します。

パーティションタイプとファイルシステムとを混同しないようにしてください。当初はパーティションタイプとファイルシステムとに対応関係がありましたが、今では対応関係がありません。パーティションタイプに関係なく、どのファイルシステムでも作成することができます（後述）。

TIP

メモという意味では、Linuxタイプのパーティションのコード `83`、Linuxのスワップ領域を示すコード `82`（Linux Swap）、Linuxの論理ボリュームマネージャ（LVM）を示すコード `8e` の3種を使い分ける程度です。

GDISKでGPTパーティションを管理する

ディスクにMBRパーティションを作成する場合には　`fdisk`　を使いますが、GPTパーティションを作成する場合には　`gdisk`　を使います。`gdisk`　のインターフェイスは　`fdisk`　がモデルになっているので、`gdisk /dev/sda`　のように実行してから表示されるプロンプトでほとんど同じコマンドを使います。`w`　コマンドを実行しない限り、ディスクに書き込みが行われないのも同じです。

現在のパーティションテーブルを表示する

`p`（print）コマンドで現在のパーティションテーブルの情報を表示できます。一例を挙げます。

```
Command (? for help): p
Disk /dev/sdb: 3903578 sectors, 1.9 GiB
Model: DataTraveler 2.0
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): AB41B5AA-A217-4D1E-8200-E062C54285BE
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 3903544
Partitions will be aligned on 2048-sector boundaries
Total free space is 1282071 sectors (626.0 MiB)
```

```
Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         2099199   1024.0 MiB  8300  Linux filesystem
   2         2623488         3147775   256.0 MiB   8300  Linux filesystem
```

冒頭部分から `fdisk` との違いがいくつかあります。

- 各ディスクに割り当てられた一意のディスク識別子（GUID）が表示されています。これは128ビットの16進数で、パーティションテーブルを作成したときにランダムに割り当てられます。この数値は3.4 × 10^{38}通りありますから、2つのディスクに同じGUIDが割り当てられる可能性は皆無に等しいです。GUIDは、どのパーティションからOSをブートするか、あるいは、どのパーティションをマウントするかなどを指定するために使えます。GUIDを使えば `/dev/sdb` のようなデバイスパスを指定しなくてもよくなります。
- `Partition table holds up to 128 entries` という文言に注目してください。GPTディスクでは128個までのパーティションを扱うことができ、拡張パーティションのような複雑な仕組みを使う必要はありません。
- 空き領域が最終行に表示されますから、`fdisk` での `F` コマンドに相当するコマンドを実行する必要はありません。

パーティションを作成する

`fdisk` と同様に、`n` コマンドでパーティションを作成できます。パーティション番号と開始セクター及び終了セクター（またはサイズ）に加えて、パーティションタイプが尋ねられます。GPTでは、16進数4桁でパーティションタイプを指定し、MBRよりも多くのパーティションタイプをサポートしています。`L` コマンドでサポートされているタイプの一覧を確認できますので、用途に応じたパーティションタイプを選択します。メモであることはMBRと同様ですから、通常はデフォルトの `8300`（Linux filesystem）を選択するのが無難でしょう。

パーティションを削除する

パーティションを削除するには、`d` コマンドを使用して、パーティション番号を入力します。`fdisk` とは異なり、ディスクにパーティションが1つしかなくても自動的にパーティションが選択されることはありません。

GPTでは、パーティション番号が飛び飛びにならないようにソートできます。`s` コマンドを実行するだけなので簡単です。例えば、次のようなパーティションテーブルだとします。

```
Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         2099199   1024.0 MiB  8300  Linux filesystem
   2         2099200         2361343   128.0 MiB   8300  Linux filesystem
```

```
   3         2361344         2623487   128.0 MiB   8300  Linux filesystem
```

2番目のパーティションを削除すると次のようになります。

```
Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         2099199   1024.0 MiB  8300  Linux filesystem
   3         2361344         2623487   128.0 MiB   8300  Linux filesystem
```

`s` コマンドを実行すると次のようになります。

```
Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         2099199   1024.0 MiB  8300  Linux filesystem
   2         2361344         2623487   128.0 MiB   8300  Linux filesystem
```

3番目のパーティションが2番目のパーティションになりました。

復元オプション

GPTでは、パーティション情報のバックアップコピーがディスク上に保存されますから、データが破損しても復元することができます。`gdisk` がパーティションテーブルの破損を検出した時や、`r` コマンドを実行した時に表示されるリカバリーメニューから処理方法を指定します。

リカバリーメニューからは、MBRをGPTに変換したり、逆に（可能であれば）GPTをMBRに変換することもできます。リカバリーメニューで `?` と入力すると、リカバリーコマンドの一覧が説明付きで表示されます。

ファイルシステムの作成

OSがディスクにファイルを保存できるようにするには、パーティション（ないしはディスク全体）に ファイルシステム を作成してフォーマットする必要があります。ファイルシステム とは、ディスク上にディレクトリやファイルを保管する方法を規定する仕様であり、その仕様に則って保存されたデータであり、またそれらへのアクセス方法を実装したプログラムでもあります。

Linuxでは多くのファイルシステムがサポートされています。ext（Extended Filesystem）系のようなLinuxネィテイブのファイルシステムだけでなく、MS-DOSのFAT、Windows NTのNTFS、macOSのHFSやHFS+なども利用できます。

Linuxでファイルシステムを作成する（フォーマットする）には、`mkfs` コマンドを使います。このコマンドは、ファイルシステムごとの専用のコマンド（たとえば `mkfs.ext4` など）を呼び出すラッパーです。

ext2/ext3/ext4ファイルシステムを作成する

Extended Filesystem（ext）はLinuxのために開発された最初のファイルシステムです。ext2、ext3、ext4と順に進化してきました。現在ではext4が多くのLinuxディストリビューションでデフォルトのファイルシステムになっています。

`mkfs.ext2` でext2、`mkfs.ext3` でext3、`mkfs.ext4` でext4のファイルシステムを作成できます。実は、これらのユーティリティは `mke2fs` という別のプログラムへのシンボリックリンクです。`mke2fs` は呼び出され方に応じて、作成するファイルシステムを切り替えますが、動作やパラメータはどの場合でも同じです。

一番シンプルな使い方は次のとおりです。

```
# mkfs.ext2 TARGET
```

`TARGET` はファイルシステムを作成するデバイス名です。例えば、パーティション `/dev/sdb1` にext3ファイルシステムを作成するなら、次のようなコマンドになります。

```
# mkfs.ext3 /dev/sdb1
```

作成するファイルシステムに応じたコマンドを使う代わりに、`mke2fs` コマンドの `-t` オプションの後にファイルシステムの名前を指定することもできます。例えば、次の2つコマンドは同じはたらきをします。どちらのコマンドでも `/dev/sdb1` にext4ファイルシステムを作成します。

```
# mkfs.ext4 /dev/sdb1
# mke2fs -t ext4 /dev/sdb1
```

コマンドラインパラメータ

`mke2fs` には様々なオプションとパラメータがあります。重要なものをいくつか紹介します。これらは `mkfs.ext2`、`mkfs.ext3`、`mkfs.ext4` でも共通です。

`-b SIZE`
: デバイスのデータブロックのサイズを設定します。`SIZE` には1024、2048、4096のいずれかを指定します。（訳注：ブロックサイズを大きくするとシーケンシャルアクセスのパフォーマンスが向上しますが、小さなファイルが多い場合のディスクの利用効率が悪くなります。）

`-c`
: ファイルシステムを作成する前にデバイスの不良ブロックをチェックします。`mkfs.ext4`

`-c -c TARGET` のように `-c` を2回繰り返すと、時間はかかるもののより綿密なチェックを行います。

`-d DIRECTORY`

指定した `DIRECTORY` の内容を、新しく作成するファイルシステムのルートディレクトリにコピーします。あらかじめ用意されたファイルがすでにある状態のディスクを作成できるということです。

`-F`

危険なオプションです。他のオプションが指定されていようが、意味をなさない指定であろうが、ターゲットデバイスが不適切であろうが、強制的にファイルシステムを作成します。`-F -F` のように2回繰り返すと、マウントされていて使用中のデバイス上にファイルシステムを作成することさえできてしまいます。そのようなことは絶対にしてはいけません。

`-L VOLUME_LABEL`

ボリュームラベルを `VOLUME_LABEL` に設定します。最大で16文字です。

`-n`

このオプションは利用価値があります。ファイルシステムの作成をシミュレートして結果を表示します。お試しモードだと考えてください（訳注：dry-runと呼ぶことも多いです）。ディスクに変更を加える前に結果を試してみることはよい習慣です。

`-q`

静音モードです。通常と同じように実行されますが、ターミナルへの出力が行われません。スクリプトから実行するときに便利です。

`-U UUID`

パーティションのUUID（Universally Unique Identifier）を指定します。UUIDは16進数表記の128ビットの数値で、これによりOSがパーティションを識別できます。この数値は、`D249E380-7719-45A1-813C-35186883987E` のように、8桁-4桁-4桁-4桁-12桁という形で表されます。UUIDを指定する代わりに、`clear` を指定するとUUIDをクリア、`random` を指定するとランダムに作成されたUUIDを付与、`time` を指定すると現在時刻に基づくUUIDを付与します。

`-v`

冗長モードです。通常よりも多くの情報を出力します。進行状況を確認したいときなどに使います。

XFSファイルシステムを作成する

XFSは、1993年にSilicon Graphics社がIRIX OS（訳注：商用UNIXの一種）のために開発したファイルシステムです。特に大きなファイルが多い場合のパフォーマンスに優れているもののメモリ使用量が多いと言われていて、大規模なサーバー環境などでよく使われていま

す。

XFSファイルシステムを管理するツールは `xfsprogs` パッケージに含まれています。Linuxディストリビューションによってはデフォルトで入っていないこともあり、その場合は手動でインストールします。Red Hat Enterprise Linuxなどのディストリビューションではデフォルトで XFSファイルシステムを使っています。

XFSファイルシステムは、ジャーナリングファイルシステム の一種であり、内部的には ログセクション と データセクション の2つの部分に分けられます。ファイルシステム操作をいったんログセクション記録しておき、後にまとめてデータセクションを更新することで、障害が発生したときにファイルシステムの不整合が起きにくい仕組みです。ログセクションとデータセクションを別のディスクに置くことで、パフォーマンスをより向上させることもできます。（訳注：ext3、ext4もジャーナリングファイルシステムの一種です。）

XFSファイルシステムを作成するには `mkfs.xfs TARGET` コマンドを使用します。`TARGET` にはファイルシステムを作成するデバイス名を指定します。例えば、`mkfs.xfs /dev/sda1` のようなコマンドを実行します。

`mke2fs` と同じように `mkfs.xfs` には様々なオプションがあります。重要なオプションをいくつか紹介します。

`-b size=VALUE`

ファイルシステムのブロックサイズを `VALUE` で指定します（単位はバイト）。デフォルトは4096バイト（4 KiB）、最小は512バイト、最大は65536バイト（64 KiB）です。

`-m crc=VALUE`

（このオプションや次のオプションのように）`-m` で始まるオプションはメタデータを変更するオプションです。このオプションはディスクのメタデータのCRC32整合性チェックを `VALUE` が `1` なら有効に、`0` なら無効にします。エラー検知とハードウェアクラッシュからの回復に役立つのでデフォルトでは有効になっています。パフォーマンスへの悪影響は最小限にとどめられていますから、無効にする理由は普通ありません。（訳注：LPIC1の範囲外です。）

`-m uuid=VALUE`

パーティションのUUIDを指定します。UUIDは、`1E83E3A3-3AE9-4AAC-BF7E-29DFFECD36C0` のように、8桁-4桁-4桁-4桁-12桁という形で表される、16進数の32文字（128ビット）の数値です。

`-f`

ターゲットデバイスに既にファイルシステムがある場合でも、上書きしてファイルシステムを強制的に作成します。

`-l logdev=DEVICE`

ログセクションをデータセクションの内部に作らず指定したデバイスに作ります。（訳

注：LPIC1の範囲外です。）

`-l size=VALUE`

ログセクションのサイズを `VALUE` で指定します。単位をつけなければバイト単位になり、メガバイトを表す `m` やギガバイトを表す `g` を数値の後につけることもできます。例えば `-l size=10m` と指定すると、ログセクションのサイズが10メガバイトになります。（訳注：LPIC1の範囲外です。）

`-q`

静音モードです。作成するファイルシステムのパラメータを出力しなくなります。

`-L LABEL`

ラベルを `LABEL` に設定します。最大で12文字です。

`-N`

`mke2fs` の `-n` オプションと同様に、実際にはファイルシステムを作成せずパラメータを表示します。

FATまたはVFATファイルシステムを作成する

MS-DOSと共に1981年に登場したFATファイルシステムは、年月を経てFAT16、VFATへと改良され、1996年にはWindows 95 OSR2とともにリリースされたFAT32に結実しました。

FAT16は、オリジナルのFAT（FAT12）を改良し、より大きなファイルを扱えるようにしたものです。VFATはFAT16の拡張形式で、最大255文字までの長いファイル名をサポートしたものです（訳注：ファイル名の扱いが変わっただけで、ファイルシステムとしてはFAT16と全く同じです）。FATもVFATも同じ `mkfs.fat` で扱えます。`mkfs.vfat` は `mkfs.fat` のエイリアス（別名）です。

FATファイルシステムには大きなディスクでの使用を制約する重大な欠点があります。例えば、FAT16は、最大で4GBのボリューム、2GBのファイルまでしか扱えません。FAT32は2PBのボリュームまで扱えるようになりましたが、4GBのファイルまでしか扱えません。この制約があるため、FATファイルシステムは、容量が2GB以内の小さいフラッシュドライブやメモリーカード、あるいは他のファイルシステムをサポートしていない古いデバイスやOSでよく使われます。

FATファイルシステムを作成するには、`mkfs.fat TARGET` コマンドを使用します。`TARGET` にはファイルシステムを作成するデバイス名を指定します。例えば、`mkfs.fat /dev/sdc1` のようなコマンドを実行します。

`mkfs.fat` にも様々なオプションがあります。重要なオプションをいくつか紹介します。`man mkfs.fat` コマンドを実行すると表示されるマニュアルで全オプションとその説明を確認できます。

-c
ファイルシステムを作成する前に、デバイスの不良ブロックをチェックします。

-C FILENAME BLOCK_COUNT
FILENAME で指定した名前のファイルを作成し、その中にFATファイルシステムを作成します。後で dd のようなユーティリティでデバイスに書き込んだりループバックデバイスとしてマウントしたりできるように、空の「ディスクイメージ」を作成する際に便利です。このオプションを指定するときには、デバイス名の後にファイルシステムのブロック数（BLOCK_COUNT）を指定することが必須になっています。

-F SIZE
FAT（File Allocation Table）のサイズを設定します。SIZE には12（FAT12）、16（FAT16）、32（FAT32）のいずれかを指定します。このオプションを指定しなければ作成するファイルシステムのサイズに応じて自動的に適切なサイズが選ばれます。

-n NAME
ファイルシステムのボリュームラベル（名前）を NAME に設定します。最大11文字です。デフォルトでは名前なしになります。

-v
冗長モードです。通常よりも多くの情報を出力します。

NOTE
mkfs.fat はブート可能なファイルシステムを作成することが できません。マニュアルページに「これは思ったほど簡単なことではありません」と記載されており、実装される見込みはありません。

exFATファイルシステムを作成する

exFATは、ディスクサイズとファイルサイズに上限があるというFAT32の重大な制約に対処するために2006年にMicrosoftが編み出したファイルシステムです。exFATでは最大ファイルサイズは16エクサバイトで（FAT32では4GBでした）、最大ディスクサイズは128ペタバイトです。

3つの主要なOS（Windows、Linux、mac OS）でサポートされていますから、大容量のフラシュドライブやメモリーカードや外付けハードディスクのようにOSをまたいで使う場合に適しています。実際に、32GBを超えるSDXCメモリーカードのデフォルトファイルシステムとして SDアソシエーション はexFATを採用しています。

exFATファイルシステムを作成するデフォルトのユーティリティは mkfs.exfat で、これは mkexfatfs へのリンクのことがあります。mkfs.exfat TARGET のように使用し、TARGET にはファイルシステムを作成するデバイス名を指定します。例えば、mkfs.exfat /dev/sdb2 のようなコマンドを実行します。（訳注：インストールされていない場合は、exfatprogsないしexfat-utilsパッケージでインストールできます。）

ここまで紹介してきた他のユーティリティとは対照的に、`mkfs.exfat` のオプションは僅かです。（訳注：指定できるオプションが異なるバージョンを採用しているディストリビューションもあるようです。）

`-i VOL_ID`

ボリュームIDとして `VOL_ID` に指定した32ビットの16進数を設定します。このオプションを指定しなければ現在時刻に基づくIDが設定されます。

`-n NAME`

ボリュームラベルを `NAME` に設定します。最大15文字です。デフォルトでは名前なしになります。

`-p SECTOR`

ディスク上の最初のパーティションの最初のセクターを指定します。デフォルトは0です。

`-s SECTORS`

1クラスターあたりの物理セクターの数を指定します。1、2、4、8などのように2の累乗の値を指定します。

Btrfsファイルシステムを理解する

Btrfs（正式には B-Tree Filesystem）は、2007年からOracle CorporationとFujitsu、Red Hat、Intel、SUSEなどがLinux用に開発している次世代ファイルシステムです。（訳注：日本では「バターエフエス」や「ビーツリーエフエス」と呼ぶ人が多いようです。）

Btrfsは、現在の大容量ストレージのニーズに応えるために開発された、新しいファイルシステムです。（RAIDでのストライピングやミラーリングのような）複数デバイスのサポート、透過圧縮、SSD最適化、増分バックアップ、スナップショット、オンラインデフラグ、オフラインチェック、（クォータが設定できる）サブボリューム、重複排除などの機能を備えています。

Btrfsは CoW（Copy On Write） ファイルシステムの一種で、クラッシュに強いです。また、多機能な割にはシンプルであり、多くのLinuxディストリビューションでサポートされています。SUSEのようにデフォルトのファイルシステムに採用しているディストリビューションもあります。（訳注：Btrfsに対する評価は分かれており、Red HatはRHEL7以降でBtrfsのサポートを打ち切っています。）

NOTE

従来のファイルシステムでは、ファイルの上書きをする際に、古いデータを新しいデータで直接置き換えます。CoW ファイルシステムでは、新しいデータを空き領域に書き込み、ファイルのメタデータがその新しいデータを指すように更新してから、必要が無くなった古いデータを削除します。この仕組みでは、新しいデータをきちんと書き込み、もう必要なくなったことを確認してから古いデータを削除するので、クラッシュした際にデータが失われる可能性が低くなります。また、古いデータを消さずにスナップショット

として残しておくことも容易になります。

Btrfsファイルシステムを作成する

Btrfsファイルシステムを作成するには `mkfs.btrfs` を使います。次のように、オプションを指定せずにデバイス名だけを指定したコマンドを実行すると、指定したデバイスにBtrfsファイルシステムを作成します。

```
# mkfs.btrfs /dev/sdb1
```

TIP システムに `mkfs.btrfs` ユーティリティが入っていなければ、パッケージマネージャーから `btrfs-progs` をインストールしてください。（訳注：Red Hat系のディストリビューションではサポートされていません。）

`-L` オプションでファイルシステムのラベル（名前）を設定できます。最大256文字で、改行文字は使用できません。

```
# mkfs.btrfs /dev/sdb1 -L "New Disk"
```

TIP ラベルにスペースを含む場合は上の例のように `"`（クオート）で囲んでください。

`mkfs.btrfs` コマンドで複数のデバイスを指定できるというBtrfsの特徴は注目に値します。Btrfsには、複数のハードディスクを組み合わせて1つの仮想的なディスクとして扱うRAIDやLVMに相当する機能が内包されています。複数のディスクにデータを分散・冗長化する方法を、`-m` オプション（メタデータの冗長化）と `-d` オプション（データブロックの冗長化）で指定します。詳細は `mkfs.btrs` のマニュアルで確認して下さい。

たとえば、2つのパーティション `/dev/sdb1` と `/dev/sdc1` を連結して、1つの大きなファイルシステムを作成するには、次のコマンドを実行します。

```
# mkfs.btrfs -d single -m single /dev/sdb /dev/sdc
```

WARNING 上記のように複数のパーティションを連結するファイルシステムには利点があるようにも見えますが、データの安全性の観点からは問題があります。1つのディスクが壊れると、ファイルシステム全体が利用できなくなってしまうからです。たくさんのディスクを連結して巨大なファイルシステムを作成したい場合は、（LPIC1の範囲外ですが）データを複数のディスクに分散して格納するRAIDに相当するオプションを指定します。

サブボリュームを管理する

サブボリュームは、既存のボリュームに割り当てられた領域の一部を利用して、独立したボリュームを作成するものです。作成したサブボリュームは、そのまま親ボリューム内のディレクトリとして使うこともできますし、独立したファイルシステムとして任意の位置にマウントして利用することもできます。サブボリュームごとにクォータ（容量制限）をかけることもできますので、体系だったシステム管理が行いやすくなります。

NOTE サブボリュームはパーティションではありません。パーティションはドライブ上にあらかじめ決まった領域を割り当てるので、あるパーティションの領域が逼迫しているのに別のパーティションには余裕があるといった問題が後々生じることがあります。サブボリュームは親ボリューム（ファイルシステム）の領域の一部をそのまま使用するので、そのような問題とは無縁です。

`/mnt/disk` にマウントされたBtrfsファイルシステムがあり、その中にバックアップを保存するための `BKP` という名前のサブボリュームを作成するには、次のコマンドを実行します。

```
# btrfs subvolume create /mnt/disk/BKP
```

`/mnt/disk` ファイルシステムの内容を一覧表示してみましょう。サブボリュームと同じ名前の新しいディレクトリがあることがわかります。

```
$ ls -lh /mnt/disk/
total 0
drwxr-xr-x 1 root   root     0 jul 13 17:35 BKP
drwxrwxr-x 1 carol carol 988 jul 13 17:30 Images
```

NOTE サブボリューム も 通常のディレクトリと同じようにアクセスできます。

次のコマンドを実行すると、サブボリュームがアクティブであることを確認できます。

```
# btrfs subvolume show /mnt/disk/BKP/
    Name:           BKP
    UUID:           e90a1afe-69fa-da4f-9764-3384f66fa32e
    Parent UUID:        -
    Received UUID:      -
    Creation time:      2019-07-13 17:35:40 -0300
    Subvolume ID:       260
    Generation:         23
    Gen at creation:    22
    Parent ID:      5
    Top level ID:       5
```

```
    Flags:          -
    Snapshot(s):
```

`mount` コマンドに `-t btrfs -o subvol=NAME` オプションを指定して、このサブボリュームを `/mnt/BKP` にマウントします（`NAME` の部分は先ほど作成したサブボリュームの名前である `BKP` とします）。

```
# mount -t btrfs -o subvol=BKP /dev/sdb1 /mnt/bkp
```

NOTE `-t` オプションの後にはマウントするファイルシステムのタイプを指定します（トピック104.3で解説します）。

スナップショットを操作する

サブボリュームの一種に、スナップショット があります。これは、作成時に親ボリュームの指定したディレクトリ以下の内容がそのまま含まれている点が、先ほど説明したサブボリュームと異なります。

スナップショットの作成時点ではスナップショットと元ボリュームの内容が全く同じですが、次第に異なっていきます。元ボリュームの変更（ファイルの作成、名前の変更、削除など）はスナップショットに反映されませんし、スナップショットの変更も元ボリュームに反映されません。

スナップショットはファイルを複製しないので、作成時にはディスク領域をほとんど使いません。ファイルシステムのツリーを複製するだけで、同じ元データを参照しています。どちらかのデータを変更すると、Copy on Writeの仕組みによって新しいデータブロックが作成されて、元のデータブロックはそのまま残ります。親データとスナップショットデータは、すこしずつ異なっていきますが、消費される容量は「変更された」データブロックの大きさだけです。

スナップショットを作成するコマンドはサブボリュームを作成するコマンドとほとんど同じで、`btrfs subvolume snapshot` を使います。次のコマンドは `/mnt/disk` にマウントされているBtrfsファイルシステムのスナップショットを `/mnt/disk/snap` に作成します。データをコピーしないので、スナップショットの作成はほぼ一瞬で完了します。

```
# btrfs subvolume snapshot /mnt/disk /mnt/disk/snap
```

`/mnt/disk` には以下の内容があるとします。

```
$ ls -lh
total 2,8M
-rw-rw-r-- 1 carol carol 109K jul 10 16:22 Galaxy_Note_10.png
```

```
-rw-rw-r-- 1 carol carol 484K jul  5 15:01 geminoid2.jpg
-rw-rw-r-- 1 carol carol 429K jul  5 14:52 geminoid.jpg
-rw-rw-r-- 1 carol carol 467K jul  2 11:48 LG-G8S-ThinQ-Mirror-White.jpg
-rw-rw-r-- 1 carol carol 654K jul  2 11:39 LG-G8S-ThinQ-Range.jpg
-rw-rw-r-- 1 carol carol  94K jul  2 15:43 Mimoji_Comparativo.jpg
-rw-rw-r-- 1 carol carol 112K jul 10 16:20 Note10Plus.jpg
drwx------ 1 carol carol  366 jul 13 17:56 snap
-rw-rw-r-- 1 carol carol 118K jul 11 16:36 Twitter_Down_20190711.jpg
-rw-rw-r-- 1 carol carol 324K jul  2 15:22 Xiaomi_Mimoji.png
```

作成したスナップショットがsnapディレクトリとして現れていることに注目してください。いくつかファイルを削除してからディレクトリの内容を確認してみましょう。

```
$ rm LG-G8S-ThinQ-*
$ ls -lh
total 1,7M
-rw-rw-r-- 1 carol carol 109K jul 10 16:22 Galaxy_Note_10.png
-rw-rw-r-- 1 carol carol 484K jul  5 15:01 geminoid2.jpg
-rw-rw-r-- 1 carol carol 429K jul  5 14:52 geminoid.jpg
-rw-rw-r-- 1 carol carol  94K jul  2 15:43 Mimoji_Comparativo.jpg
-rw-rw-r-- 1 carol carol 112K jul 10 16:20 Note10Plus.jpg
drwx------ 1 carol carol  366 jul 13 17:56 snap
-rw-rw-r-- 1 carol carol 118K jul 11 16:36 Twitter_Down_20190711.jpg
-rw-rw-r-- 1 carol carol 324K jul  2 15:22 Xiaomi_Mimoji.png
```

snapディレクトリの中には先ほど削除したファイルが存在していますから、そこからコピーして元に戻すこともできます。

```
$ ls -lh snap/
total 2,8M
-rw-rw-r-- 1 carol carol 109K jul 10 16:22 Galaxy_Note_10.png
-rw-rw-r-- 1 carol carol 484K jul  5 15:01 geminoid2.jpg
-rw-rw-r-- 1 carol carol 429K jul  5 14:52 geminoid.jpg
-rw-rw-r-- 1 carol carol 467K jul  2 11:48 LG-G8S-ThinQ-Mirror-White.jpg
-rw-rw-r-- 1 carol carol 654K jul  2 11:39 LG-G8S-ThinQ-Range.jpg
-rw-rw-r-- 1 carol carol  94K jul  2 15:43 Mimoji_Comparativo.jpg
-rw-rw-r-- 1 carol carol 112K jul 10 16:20 Note10Plus.jpg
-rw-rw-r-- 1 carol carol 118K jul 11 16:36 Twitter_Down_20190711.jpg
-rw-rw-r-- 1 carol carol 324K jul  2 15:22 Xiaomi_Mimoji.png
```

読み取り専用のスナップショットを作成することもできます。スナップショットの内容を変更できないことを除いては、通常の書き込み可能なスナップショットと同じです。読み取り

専用のスナップショットを使うとある時点での内容を「固定」することができます。スナップショットの作成時に `-r` パラメータをつけると読み取り専用スナップショットになります。人為的なミスに対する保険としてはとても便利ですが、同じディスク上にデータがあるのですからバックアップにはなりません。

```
# btrfs subvolume snapshot -r /mnt/disk /mnt/disk/snap
```

圧縮に関して少々

Btrfsは透過圧縮をサポートしています。3種類のアルゴリズムから選べます。`-o compress` オプションをつけてファイルシステムをマウントすれば、ファイルごとに自動で圧縮されます。圧縮できないファイルを圧縮しようとはしませんから、システムリソースが浪費されるということはありません。1つのディレクトリに圧縮ファイルと非圧縮ファイルが共存していても問題ありません。デフォルトの圧縮アルゴリズムはZLIBですが、LZO（速いけれども圧縮率は低い）やZSTD（ZLIBより速く圧縮率は同等）も選べます。圧縮レベルも選べます（詳細については `btrfs(5)` の Mount Optionsを参照してください）。

GNU Parted

GNU Parted はパーティションの作成・削除・移動・リサイズ・コピーができる非常に強力なパーティションエディタです。GPTとMBRのどちらでも使え、ディスク管理はこれだけでほとんどカバーできます。

GNOMEデスクトップ環境の GParted やKDEデスクトップ環境の KDE Partition Manager のように、グラフィカルに `parted` が使えるフロントエンドツールがいろいろあります。それでも、グラフィカルなデスクトップ環境が使えないサーバーのセットアップではコマンドラインを使うことになりますから、`parted` の使い方を知っておいたほうがよいでしょう。

WARNING

`fdisk` や `gdisk` とは異なり、`parted` はコマンドを実行すると 即座に ディスクに書き込みますので、Writeコマンドはありません。`parted` コマンドの練習をするときは、不要なUSBフラッシュドライブなどの外部ディスクを使ってください（訳注：外部ディスクを使って練習する際にはデバイス名を間違えないように何度も確認してください。また、VirtualBoxやKVMなどの仮想マシンを作ってその中で練習してもよいです）。

パーティションの操作をするには `parted DEVICE` と入力して実行します。`DEVICE` はパーティションを操作するデバイスです。例えば `parted /dev/sdb` のようなコマンドを実行します。`fdisk` や `gdisk` と同じように、コマンドを入力するプロンプトが表示される対話的なコマンドラインインターフェイスが起動します。

```
# parted /dev/sdb
```

```
GNU Parted 3.2
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.

(parted)
```

WARNING デバイスを指定しないとプライマリーディスク（通常は `/dev/sda`）がデフォルトで選択されるので気をつけてください。

ディスクを選択する

起動時にコマンドラインで指定したディスクを変更するには `parted` のプロンプトで `select` コマンドを実行します。次に示すように、`select` の後にデバイス名を指定します。

```
(parted) select /dev/sdb
Using /dev/sdb
```

情報を取得する

`print` コマンドでシステムに接続されているパーティションやディスクの詳細情報を見ることができます。

`print` とだけ入力すると、現在選択しているデバイスのパーティション情報が表示されます。パーティションの開始位置や終了位置の表示単位は、`unit` コマンドで切り替えられます。

```
(parted) print
Model: ATA CT120BX500SSD1 (scsi)
Disk /dev/sda: 120GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system     Flags
 1      2097kB  116GB   116GB   primary  ext4
 2      116GB   120GB   4295MB  primary  linux-swap(v1)
```

`print devices` と入力すると、すべてのブロックデバイス（ディスク）の一覧が表示されます。

```
(parted) print devices
```

```
/dev/sdb (1999MB)
/dev/sda (120GB)
/dev/sdc (320GB)
/dev/mapper/cryptswap (4294MB)
```

`print all` と入力すると接続されているデバイスの情報を一望できますし、`print free` と入力すると各デバイスの空き領域を確認できます。

```
(parted) print free
Model: ATA CT120BX500SSD1 (scsi)
Disk /dev/sda: 120GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system     Flags
        32.3kB  2097kB  2065kB           Free Space
 1      2097kB  116GB   116GB   primary  ext4
        116GB   116GB   512B             Free Space
 2      116GB   120GB   4295MB  primary  linux-swap(v1)
        120GB   120GB   2098kB           Free Space
```

空のディスクにパーティションテーブルを作成する

`mklabel` コマンドの後にパーティションテーブルのタイプを指定すると、空のディスクにパーティションテーブルを作成できます。

パーティションテーブルのタイプにはいろいろありますが、MBRパーティションテーブルを作成する `msdos` とGPTパーティションテーブルを作成する `gpt` の2つを覚えておけば十分です。MBRパーティションテーブルを作成するなら次のコマンドを実行します。

```
(parted) mklabel msdos
```

GPTパーティションテーブルを作成するなら次のコマンドを実行します。

```
(parted) mklabel gpt
```

パーティションを作成する

`mkpart PARTTYPE FSTYPE START END` コマンドでパーティションを作成できます。

PARTTYPE

パーティションタイプです。MBRパーティションの場合、`primary`、`logical`、`extended` のどれかを指定します。GPTの場合は、パーティション名（`Linux filesystem` など）を指定します。

FSTYPE

ファイルシステムを指定します。`parted` はファイルシステムを実際には 作成せず、OSに向けたフラグを設定するだけだということに注意してください。`ext4`、`xfs`、`fat32` などのファイルシステムタイプ名を指定しますが、GPTの場合は省略できます。

START

パーティションの開始位置を指定します。指定できる単位が数種類あります。`2s` はディスクの2つ目のセクター、`1m` はディスクの先頭から1メガバイトになります。`B`（バイト）や `g`（ギガ）、`%`（ディスク容量のパーセンテージ）でも指定できます。（訳注：実際の位置（開始セクター）はディスクに応じた適当な値に調整されますので、大まかにパーセンテージを使って指定するのがお勧めです。）

END

パーティションの終了位置を指定します。パーティションサイズではなく、ディスクの先頭からの終了位置を指定することに気をつけてください。例えば、`100m` と指定したらそのパーティションはディスクの先頭から100MB辺りの適当な位置で終了するということです。START の部分と同じ単位で指定できます。

```
(parted) mkpart primary ext4 1m 100m
```

このコマンドは、ディスクの先頭から1メガバイト辺りから始まり100メガバイト辺りで終了する、`ext4` の基本パーティションを作成します。

パーティションを削除する

`rm` コマンドの後にパーティション番号を指定すると、そのパーティションを削除できます。パーティション番号は `print` コマンドで確認できます。例えば、`rm 2` は現在選択しているディスクの2番目のパーティションを削除します。

パーティションを復元する

`parted` は削除されたパーティションを復元することもできます。最初に次のようなパーティションだったとします。

```
Number  Start   End     Size    File system  Name     Flags
 1      1049kB  99.6MB  98.6MB  ext4         primary
 2      99.6MB  200MB   100MB   ext4         primary
```

```
 3      200MB   300MB   99.6MB  ext4         primary
```

誤って `rm 2` を実行してパーティション2を削除してしまったとしましょう。`rescue START END` コマンドを実行するとこのパーティションを復元できます。`START` は復元するパーティションのおおよその開始位置、`END` はおおよその終了位置です。

`parted` はディスクをスキャンしてパーティションを探索し、発見したパーティションを復元します。上の例ではパーティション2は99.6MBから開始して200MBで終了しているので、次のコマンドで復元できます。

```
(parted) rescue 90m 210m
Information: A ext4 primary partition was found at 99.6MB -> 200MB.
Do you want to add it to the partition table?

Yes/No/Cancel? y
```

これでパーティション2を復元します。ただし、ファイルシステムが作成されているパーティションしか復元できません。ファイルシステムが作成されていない空のパーティションは探索できないのです。

ext2/3/4パーティションをリサイズする

`parted` ではパーティションのリサイズもできますが、注意事項がいくつかあります。

- リサイズ中にパーティションを使用したりマウントしたりしてはいけません。
- サイズを拡張するには、そのパーティションの *後* に、拡張したいサイズ以上の空き領域が必要になります。

`resizepart` の後にパーティション番号と終了位置を指定します。次のようなパーティションテーブルがあるとします。

```
Number  Start   End     Size    File system  Name     Flags
 1      1049kB  99.6MB  98.6MB  ext4         primary
 2      99.6MB  200MB   100MB   ext4
 3      200MB   300MB   99.6MB  ext4         primary
```

`resizepart` でパーティション1を拡張しようとしてもエラーになります。パーティション1の後にパーティション2があって空き領域がないからです。パーティション3はその後に空き領域がありますから、リサイズできます。`print free` で確認してみましょう。

```
(parted) print free
```

```
Model: Kingston DataTraveler 2.0 (scsi)
Disk /dev/sdb: 1999MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

 Number  Start   End     Size    File system  Name     Flags
         17.4kB  1049kB  1031kB  Free Space
 1       1049kB  99.6MB  98.6MB  ext4         primary
 2       99.6MB  200MB   100MB   ext4
 3       200MB   300MB   99.6MB  ext4         primary
         300MB   1999MB  1699MB  Free Space
```

次のコマンドでパーティション3を350MBのところまで拡張してみましょう。

```
(parted) resizepart 3 350m

(parted) print
Model: Kingston DataTraveler 2.0 (scsi)
Disk /dev/sdb: 1999MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name     Flags
 1      1049kB  99.6MB  98.6MB  ext4         primary
 2      99.6MB  200MB   100MB   ext4
 3      200MB   350MB   150MB   ext4         primary
```

拡張後の終了位置はディスクの開始位置から数えるので、350mと入力すると、終了位置がこれまでの300MBから350MBに変更され、50MB分だけ拡張します。

パーティションのリサイズだけでは片手落ちです。ファイルシステムのリサイズも必要になります。 ext2/3/4ファイルシステムは `resize2fs` コマンドでリサイズできます。上の例では、まだファイルシステムのリサイズをしていないので、マウントしたときに古いサイズが表示されます（訳注：古いサイズは99.6MBでここに表示されている88Mとは異なりますが、ディスクの全ての領域をデータ保存に使うわけではないので `df` で表示されるサイズは少し小さくなります）。

```
$ df -h /dev/sdb3
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb3        88M  1.6M   80M   2% /media/carol/part3
```

`resize2fs DEVICE SIZE` コマンドでファイルシステムをリサイズできます。DEVICE にはリサイズするデバイス名を、SIZE には新しいサイズを指定します。SIZE を省略すると、そのパーティションいっぱいまで拡張されます。リサイズする前にマウントを解除しておきます。

次のコマンドで実行します。

```
$ sudo resize2fs /dev/sdb3
resize2fs 1.44.6 (5-Mar-2019)
Resizing the filesystem on /dev/sdb3 to 146212 (1k) blocks.
The filesystem on /dev/sdb3 is now 146212 (1k) blocks long.

$ df -h /dev/sdb3
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb3       135M  1.6M  123M   2% /media/carol/part3
```

パーティションを 縮小 するときは、この手順を逆に実行します。まず ファイルシステムを新しいサイズに縮小してから、`parted` コマンドでパーティションを縮小します。

WARNING パーティションを縮小するときはこの順番に気をつけてください。ファイルシステムを縮小する前にパーティションを縮小するとデータが失われます。

上の例でパーティションを縮小して元のサイズに戻してみます。

```
# resize2fs /dev/sdb3 88m
resize2fs 1.44.6 (5-Mar-2019)
Resizing the filesystem on /dev/sdb3 to 90112 (1k) blocks.
The filesystem on /dev/sdb3 is now 90112 (1k) blocks long.

# parted /dev/sdb3
(parted) resizepart 3 300m
Warning: Shrinking a partition can cause data loss, are you sure
you want to continue?

Yes/No? y

(parted) print
Model: Kingston DataTraveler 2.0 (scsi)
Disk /dev/sdb: 1999MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:
```

```
Number  Start   End     Size    File system  Name     Flags
 1      1049kB  99.6MB  98.6MB  ext4         primary
 2      99.6MB  200MB   100MB   ext4
 3      200MB   300MB   99.7MB  ext4         primary
```

TIP 縮小後のサイズを指定する代わりに `-M` パラメータを指定して `resize2fs` を実行することもできます。その場合、現存するファイルに合わせたサイズに縮小します。

スワップパーティションの作成

Linuxの仮想メモリでは、スワッピング と言って、メモリページを必要に応じてRAMとディスクの間で移動します。スワッピングに使用するディスク上の領域として、独立したパーティションを使うケースが一般的です。そのためのパーティションを スワップパーティション あるいは スワップ領域 と呼びます。スワップパーティション専用のパーティションタイプがあり、使用前には `mkswap` コマンドで初期化しておきます。

`fdisk` や `gdisk` でスワップパーティションを作成できます。これまでに説明した通常のパーティションの作成と同じ手順です。パーティションタイプを Linux swap に設定します。

- `fdisk` では `t` コマンドでタイプを指定します。対象のパーティションを選んでから `82` と入力します。そして `w` コマンドで書き込んで終了します。
- `gdisk` でも `t` コマンドでタイプを指定しますが、コードは `8200` です。そして `w` コマンドで書き込んで終了します。

`parted` ではパーティションの作成時にファイルタイプとして `linux-swap` と指定します。例えば、次のコマンドを実行するとディスクの300MBから800MBまでの500MBのスワップパーティションを作成します。

```
(parted) mkpart primary linux-swap 301m 800m
```

パーティションを作成してタイプを設定したら、パーティションを指定して `mkswap` コマンドを実行して初期化します。

```
# mkswap /dev/sda2
```

スワップパーティションを追加したことをカーネルに通知して、スワップ領域としての利用を開始させるために、`swapon` コマンドを実行します。設定を永続化したいときは、`/etc/fstab` に記載します（トピック104.3で解説します）。

```
# swapon /dev/sda2
```

スワップ領域としての利用を停止するには、`swapoff` コマンドを実行します。

Linuxでは、パーティションではなく ファイル をスワップ領域として使うこともできます。`dd`（ないしは `fallocate`）コマンドで適当なサイズの空ファイルを作成してから、`mkswap` コマンドと `swapon` コマンドを実行します。

カレントディレクトリに `myswap` という名前の1GBの空ファイルを作ってから、スワップファイルとして使えるようにしてみましょう。

ここでは、`dd` コマンドでスワップファイルを作成します。

```
$ dd if=/dev/zero of=myswap bs=1M count=1024
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 7.49254 s, 143 MB/s
```

`if=` で入力ファイル（input file）を指定します。この例では `/dev/zero` デバイスを指定していますから、`NULL` 文字が無限に読み込まれます。`of=` で出力ファイル（output file）を指定します。`bs=` は1回に読み出すデータブロックのサイズ（block size）で、ここでは1MBを指定しています。`count=` はデータブロックを読み込む回数です。ここでは1024回を指定していますから、1MBを1024回読み出して1GBになります。つまり、1GBのファイル `myswap` が作成されます。

```
# mkswap myswap
Setting up swapspace version 1, size = 1024 MiB (1073737728 bytes)
no label, UUID=49c53bc4-c4b1-4a8b-a613-8f42cb275b2b

# swapon myswap
```

上のコマンドを実行すると、`myswap` ファイルがスワップ領域として利用され始めます。再起動しても `myswap` ファイルは存在していますが、自動的にスワップ領域として使われるようになるわけではありません。`/etc/fstab` にスワップファイルの設定を加えることで自動的にスワップファイルとして使えるようにできます（トピック104.3で説明します）。

TIP

`mkswap` コマンドと `swapon` コマンドはスワップファイルのパーミッションが適切でないとエラーになります。適切なパーミッションは `0600` で、ファイルの所有者とグループは `root` です。

演習

1. 3TBのハードディスクに1GBの3つのパーティションを作成する場合、GPTとMBRのどちらを使うべきでしょうか？ 理由とともに答えてください。

2. `gdisk` コマンドでディスクの空き領域を表示してください。

3. `/dev/sdc1` デバイスに、ランダムなUUIDで、「MyDisk」というラベルを付けて、不良ブロックのチェックをしてから、ext3ファイルシステムを作成してください。

4. `parted` コマンドで、ディスクの500MBの位置から開始する300MBのext4パーティションを 作成してください。

5. `/dev/sda1` と `/dev/sda2` にそれぞれ20GBのパーティションがあるとします。これら2つのパーティションを、ミラーリング冗長化（Raid1相当）する1つのBtrfsファイルシステムにしてください。そのファイルシステムのサイズも答えてください。

発展演習

1. 次のようなMBRパーティションテーブルの2GBのディスクがあるとします。

```
Disk /dev/sdb: 1.9 GiB, 1998631936 bytes, 3903578 sectors
Disk model: DataTraveler 2.0
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x31a83a48

Device     Boot   Start     End Sectors  Size Id Type
/dev/sdb1          2048 1050623 1048576  512M 83 Linux
/dev/sdb3       2099200 3147775 1048576  512M 83 Linux
```

   600MBのパーティションを作成できますか？ 理由も示してください。

2. `/dev/sdc` ディスクの最初に1GBのパーティションがあり、ext4ファイルシステムで241MBのファイルが入っています。`parted` コマンドでこのファイルがぴったり収まるサイズにパーティションを縮小してください。

3. `/dev/sdb` ディスクの最初に1GBのパーティションを作成しました。具体的には、`parted /dev/sdb` を実行してから `mkpart primary linux-swap 0 1024M` を実行してパーティションを作成しました。次に、このパーティションをスワップパーティションとして使うために `swapon /dev/sdb1` コマンドを実行したところ、次のエラーメッセージが表示されました。

   ```
   swapon: /dev/sdb1: read swap header failed
   ```

   何が問題だったのでしょうか？

4. `parted` コマンドで誤ってハードディスク上の3番目のパーティションを削除してしまったとします。その3番目のパーティションは、250MBのEFIパーティションと4GBのスワップパーティションの後にあり、10GBのサイズだということはわかっています。`parted` コマンドでその3番目のパーティションを復元してください。

5. `/dev/sda3` に4GBの未利用のパーティションがあるとします。`fdisk` コマンドを使用

してこのパーティションをスワップパーティションとして使えるようにする一連の操作を実行してください。

まとめ

このレッスンでは、次のことを学びました。

- `fdisk` コマンドによるMBRパーティションテーブルの作り方とパーティションの作成及び削除
- `gdisk` コマンドによるGPTパーティションテーブルの作り方とパーティションの作成及び削除
- ext2、ext3、ext4、XFS、VFAT、exFATタイプのパーティションの作り方
- `parted` コマンドによるMBRとGPTパーティションの作成、削除、復元
- ext2、ext3、ext4タイプのパーティションのリサイズ
- スワップパーティションとスワップファイルの作成及び設定

このレッスンでは、次のコマンドを説明しました:

- `fdisk`
- `gdisk`
- `mkfs.ext2`、`mkfs.ext3`、`mkfs.ext4`、`mkfs.xfs`、`mkfs.vfat`、`mkfs.exfat`
- `parted`
- `btrfs`
- `mkswap`
- `swapon`、`swapoff`

演習の解答

1. 3TBのハードディスクで1GBの3つのパーティションを作成する場合、GPTとMBRのどちらを使うべきでしょうか？ 理由とともに答えてください。

   GPTです。MBRは2TBのハードディスクまでしかサポートしないので不適です。

2. `gdisk` コマンドでディスクの空き領域を表示してください。

   `p` コマンドを実行します。パーティションテーブルの情報の前にディスクの空き領域が表示されます。

3. `/dev/sdc1` デバイスに、ランダムなUUIDで、「MyDisk」というラベルを付けて、不良ブロックのチェックをしてから、ext3ファイルシステムを作成してください。

   `mkfs.ext3 -c -L MyDisk -U random /dev/sdc1` コマンドを実行します。`mkfs.ext3` の部分は `mke2fs -t ext3` でも構いません。

4. `parted` コマンドで、ディスクの500MBの位置から開始する300MBのext4パーティションを 作成してください。

   `parted` コマンドを実行すると起動するプロンプトで、`mkpart primary ext4 500m 800m` コマンドを実行します。`parted` コマンドはファイルシステムを作成しませんから、`mkfs.ext4` コマンドを実行してファイルシステムを作成することを忘れないでください。

5. `/dev/sda1` と `/dev/sda2` にそれぞれ20GBのパーティションがあるとします。これら2つのパーティションを、ミラーリング冗長化（Raid1相当）する1つのBtrfsファイルシステムにしてください。そのファイルシステムのサイズも答えてください。

   `mkfs.btrfs /dev/sda1 /dev/sdb1 -m raid1` コマンドを実行します。ミラーリングなのでファイルシステムのサイズは20GBになります。

発展演習の解答

1. 次のようなMBRパーティションテーブルの2GBのディスクがあるとします。

```
Disk /dev/sdb: 1.9 GiB, 1998631936 bytes, 3903578 sectors
Disk model: DataTraveler 2.0
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x31a83a48

Device     Boot   Start     End Sectors  Size Id Type
/dev/sdb1          2048 1050623 1048576  512M 83 Linux
/dev/sdb3       2099200 3147775 1048576  512M 83 Linux
```

   600MBのパーティションを作成できますか？ 理由も示してください。

   作成できません。連続した空き領域が足りないからです。デバイスのリストを見れば何かが欠けていることがわかります。`/dev/sdb1` と `/dev/sdb3` があるのに `/dev/sdb2` がありません。

   次にパーティションの開始位置と終了位置を見ます。パーティション1は `1050623` セクターで終了しており、パーティション2は `2099200` セクターから開始しています。2099200 − 1050623の1048577セクターの隙間があります。1セクターは512バイトですから、これは512 × 1048577で536,871,424バイトとなります。1024で割ってキロバイト単位にするなら524,288KBです。さらに1024で割ると512MBだとわかります。これが「隙間」のサイズになります。

   ディスクは2GBですから、パーティション3の後にあるのは最大でも512MBです。トータルでは約1GBの未割り当ての空き領域があるとしても、最大の連続する空き領域は512MBですからMB、600MBのパーティションを作成するには足りません。

2. `/dev/sdc` ディスクの最初に1GBのパーティションがあり、ext4ファイルシステムで241MBのファイルが入っています。`parted` コマンドでこのファイルがぴったり収まるサイズにパーティションを縮小してください。

   2段階に分けて考えます。1段階目では `resize2fs` コマンドでファイルシステムを縮小します。縮小後のサイズを直接指定せずに `-M` パラメータを使うとファイルがぴったり収まるサイズに縮小できます。 `resize2fs -M /dev/sdc1` コマンドを実行するということです。

   2段階目では `resizepart` コマンドでパーティションの縮小をします。ディスク上で最初のパーティションですから、1から始まり241MBで終わるようにします。`resizepart 1`

`241M` コマンドを実行するということです。

3. `/dev/sdb` ディスクの最初に1GBのパーティションを作成しました。具体的には、`parted /dev/sdb` を実行してから `mkpart primary linux-swap 0 1024M` を実行してパーティションを作成しました。次に、このパーティションをスワップパーティションとして使うために `swapon /dev/sdb1` コマンドを実行したところ、次のエラーメッセージが表示されました。

   ```
   swapon: /dev/sdb1: read swap header failed
   ```

   何が問題だったのでしょうか？

   パーティションを適切なタイプ（`linux-swap`）で作成していますが、`mkpart` はパーティションを作成するだけです。`mkswap` コマンドでそのパーティションをスワップ領域として初期化していなかったことが問題でした。

4. `parted` コマンドで誤ってハードディスク上の3番目のパーティションを削除してしまったとします。その3番目のパーティションは、250MBのEFIパーティションと4GBのスワップパーティションの後にあり、10GBのサイズだということはわかっています。`parted` コマンドでその3番目のパーティションを復元してください。

   パーティションを復元するのに十分な情報は手元にありますから、落ち着いて計算して（`parted` のプロンプトで）`rescue` コマンドを実行しましょう。復元するパーティションの前には250MB ＋ 4.096 MB（4 × 1024）がありますから、開始位置はおよそ4346MBです。復元するパーティションのサイズである10,240MB（10 × 1024）を加えた14,586MBが終了位置です。よって、`rescue 4346m 14586m` コマンドで復元できます。開始位置を少し前に、終了位置を少し後にして余裕を持たせてもよいでしょう。

5. `/dev/sda3` に4GBの未利用のパーティションがあるとします。`fdisk` コマンドを使用してこのパーティションをスワップパーティションとして使えるようにする一連の操作を実行してください。

   まず、`fdisk /dev/sda` を実行してプロンプトを起動し、`t`、`3`、`82`、`w` と順に入力してそのパーティションタイプを `Linux Swap` にして、変更を書き込んで終了します。次に `mkswap /dev/sda3` コマンドでそのパーティションをスワップ領域に初期化します。そして `swapon /dev/sda3` を実行すれば、そのパーティションをスワップパーティションとして使えるようになります。

104.2 ファイルシステムの整合性を維持する

LPI目標への参照

LPIC-1 version 5.0, Exam 101, Objective 104.2

総重量

2

主な知識分野

- ファイルシステムの整合性を確認する。
- 空き領域とinodeを監視する。
- シンプルなファイルシステムの問題を修復する。

用語とユーティリティ

- `du`
- `df`
- `fsck`
- `e2fsck`
- `mke2fs`
- `tune2fs`
- `xfs_repair`
- `xfs_fsr`
- `xfs_db`

104.2 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	104 デバイス、Linuxファイルシステム、ファイルシステム階層標準
Objective:	104.2 ファイルシステムの整合性を維持する
Lesson:	1 of 1

はじめに

最近のLinuxファイルシステムは、ジャーナリングファイルシステムです。ファイルシステム操作をいったんログ（ジャーナル）に記録して、後に（ファイルシステム管理において非常に重要な）メタデータをまとめて書き換えます。これにより、カーネルパニックや電源喪失などのシステムエラーにより処理が中断した場合であっても、ジャーナルの記録を参照して処理をやり直すことでメタデータの整合性を維持して、ファイルシステム全体の破損を防ぎます。

この仕組みのおかげでファイルシステムの整合性をチェックをする頻度を大幅に減らすことができますが、それでも手動でチェックを行う必要があり得ます。現れる症状は実にさまざまですが、たとえばあるディレクトリのファイルにアクセスすると出会ったことがないエラーが表示されるとか、大きなファイルを途中のある位置までしか読み込めないといった不可解な状況に出会ったときには、ファイルシステムのチェックを行ってみるとよいでしょう。

このレッスンでは、ファイルシステム管理の第一歩として、まずファイルシステムの使用量を監視するツールを紹介します。その後に、ファイルシステムの破損をチェックして修復する方法を説明します。

ディスク使用量の確認

ファイルシステムの使用量を表示する `du`（Disk Usage）コマンドと、空き容量を表示する `df`（Disk Free）コマンドを紹介します。

`du` は、引数に指定したディレクトリが使用している容量を表示するコマンドです。引数を省略した場合には、カレントディレクトリを指定したものとみなして、そこに含まれるすべてのファイルならびにサブディレクトリが使用しているディスク容量をキロバイト単位で表示します。たとえば、いくつかのファイルだけが存在するディレクトリで `du` コマンドを実行すると、それらのファイルが使用している合計量をキロバイト単位で表示します。ファイルの長さの合計ではなく、実際に使用しているディスクブロックの数に基づく容量であることに注意してください。

```
$ du
4816    .
```

これではわかりづらいので、`-h` オプションを指定して人間が読みやすい（human readable）単位で表示します。

```
$ du -h
4.8M    .
```

デフォルトではディレクトリが使用している合計量を表示しますが、`-a` オプションを指定するとディレクトリ内の各ファイルの使用量を表示します。

```
$ du -ah
432K    ./geminoid.jpg
508K    ./Linear_B_Hero.jpg
468K    ./LG-G8S-ThinQ-Mirror-White.jpg
656K    ./LG-G8S-ThinQ-Range.jpg
60K ./Stranger3_Titulo.png
108K    ./Baidu_Banho.jpg
324K    ./Xiaomi_Mimoji.png
284K    ./Mi_CC_9e.jpg
96K ./Mimoji_Comparativo.jpg
32K ./Xiaomi FCC.jpg
484K    ./geminoid2.jpg
108K    ./Mimoji_Abre.jpg
88K ./Mi8_Hero.jpg
832K    ./Tablet_Linear_B.jpg
332K    ./Mimoji_Comparativo.png
4.8M    .
```

サブディレクトリがある場合には、その使用量を表示してから、最後に　サブディレクトリの使用量を含めて　　指定したディレクトリ（この場合はカレントディレクトリ）全体の使用量を表示します。

```
$ du -h
4.8M    ./Temp
6.0M    .
```

上の例では、`Temp`　サブディレクトリが4.8MBの使用量で、`Temp`　サブディレクトリを含めたカレントディレクトリが6.0MBの使用量だとわかります。`-S`　　オプションを指定すると、ディレクトリごとに使用量が集計され、親ディレクトリの使用量に、そこに含まれるサブディレクトリの使用量が含まれなくなります（訳注：端数処理の関係で、サブディレクトリを含めた親ディレクトリの使用量からサブディレクトリの合計使用量を引いた値と、`-S` オプションを指定して表示される値とが一致しないことがあります）。

```
$ du -Sh
4.8M    ./Temp
1.3M    .
```

TIP

オプションは大文字と小文字を区別しますから、`-s`　と　`-S`　は異なります。`-s`　　オプションを指定すると、サブディレクトリの使用量を表示せずに、引数に指定したディレクトリ以下の合計使用量のみが表示されます。たとえば `du -hs /home/*` とすると、ユーザーごとのホームディレクトリ使用量が一望できます。

`-S`　オプションと　`-c`　オプションを併用すれば、親ディレクトリ自体の（サブディレクトリを含まない）使用量と、サブディレクトリを含む合計使用量の両方が表示されます。

```
$ du -Shc
4.8M    ./Temp
1.3M    .
6.0M    total
```

表示するサブディレクトリの深さは、`-d N` オプションを指定することで調整できます。Nは表示する最大深度です。例えば、`-d 1` オプションを指定すると、カレントディレクトリとカレントディレクトリ直下のサブディレクトリは表示しますが、サブディレクトリのサブディレクトリ（孫ディレクトリ）以下は表示しません。

`-d` オプションなしでは次のように表示します。

```
$ du -h
```

```
216K    ./somedir/anotherdir
224K    ./somedir
232K    .
```

`-d 1` オプションを指定して最大深度を1にすると、次のように表示します（訳注：`-d` と数値の間に半角スペースがあってもなくても同じように動作します）。

```
$ du -h -d1
224K    ./somedir
232K    .
```

深度2の `anotherdir` を表示していませんが、その使用量は表示しているディレクトリ（somedir）の使用量に含まれています。

ある種のファイルを除外して使用量を知りたい場合には、`--exclude="PATTERN"` オプションを指定します。`PATTERN` にはグロブパターンを記述します。カレントディレクトリ内のファイルが次のとおりだとします。

```
$ du -ah
124K    ./ASM68K.EXE
2.0M    ./Contra.bin
36K ./fixheadr.exe
4.0K    ./README.txt
2.1M    ./Contra_NEW.bin
4.0K    ./Built.bat
8.0K    ./Contra_Main.asm
4.2M    .
```

`--exclude` を指定して `.bin` サフィックスのファイルを除外します。

```
$ du -ah --exclude="*.bin"
124K    ./ASM68K.EXE
36K ./fixheadr.exe
4.0K    ./README.txt
4.0K    ./Built.bat
8.0K    ./Contra_Main.asm
180K    .
```

除外したファイルは、最終行の合計からも除外されています。

空き容量の確認

`du` はファイルシステムの使用量を表示しますが、`df` コマンドはファイルシステムの空き容量を表示します。

`df` コマンドは、システム上で使用できる（マウントされている）ファイルシステムのリストを表示します。サイズ、使用量、空き容量、使用率、マウントポイントを表示します。

```
$ df
Filesystem     1K-blocks      Used Available Use% Mounted on
udev             2943068         0   2943068   0% /dev
tmpfs             595892      2496    593396   1% /run
/dev/sda1      110722904  25600600  79454800  25% /
tmpfs            2979440    951208   2028232  32% /dev/shm
tmpfs               5120         0      5120   0% /run/lock
tmpfs            2979440         0   2979440   0% /sys/fs/cgroup
tmpfs             595888        24    595864   1% /run/user/119
tmpfs             595888       116    595772   1% /run/user/1000
/dev/sdb1          89111      1550     80824   2% /media/carol/part1
/dev/sdb3          83187      1550     75330   3% /media/carol/part3
/dev/sdb2          90827      1921     82045   3% /media/carol/part2
/dev/sdc1      312570036 233740356  78829680  75% /media/carol/Samsung Externo
```

1KBブロック単位でサイズを表示されてもわかりづらいので、`du` と同様に `-h` オプションを指定するとわかりやすい出力になります。

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            2.9G     0  2.9G   0% /dev
tmpfs           582M  2.5M  580M   1% /run
/dev/sda1       106G   25G   76G  25% /
tmpfs           2.9G  930M  2.0G  32% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           2.9G     0  2.9G   0% /sys/fs/cgroup
tmpfs           582M   24K  582M   1% /run/user/119
tmpfs           582M  116K  582M   1% /run/user/1000
/dev/sdb1        88M  1.6M   79M   2% /media/carol/part1
/dev/sdb3        82M  1.6M   74M   3% /media/carol/part3
/dev/sdb2        89M  1.9M   81M   3% /media/carol/part2
/dev/sdc1       299G  223G   76G  75% /media/carol/Samsung Externo
```

`-i` オプションを指定すると、ブロックサイズの代わりにinode（iノード）の情報を表示します。inodeというのは、それぞれのファイルのメタデータ（ファイル本体を収めたディ

スクブロックの位置、所有者、所有グループ、パーミッションなど）を収めたデータ構造で、mkfsでファイルシステムを作成する際に、ディスクサイズに応じて適当な個数の領域が準備されます。ファイルやディレクトリごとに1つのinodeが消費されますので、小さなデータファイルが大量に収められるファイルシステムでは、データ領域が余っているにもかかわらず、inodeが足りないためにファイルを作成できないということもあり得ます。空き容量と合わせて、時々は使用状況をチェックしておきましょう。

```
$ df -i
Filesystem      Inodes  IUsed   IFree IUse% Mounted on
udev            737142    547  736595    1% /dev
tmpfs           745218    908  744310    1% /run
/dev/sda6      6766592 307153 6459439    5% /
tmpfs           745218    215  745003    1% /dev/shm
tmpfs           745218      4  745214    1% /run/lock
tmpfs           745218     18  745200    1% /sys/fs/cgroup
/dev/sda1        62464    355   62109    1% /boot
tmpfs           745218     43  745175    1% /run/user/1000
```

`-T` オプションを指定すると、それぞれのファイルシステムのタイプも表示します。

```
$ df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
udev           devtmpfs  2.9G     0  2.9G   0% /dev
tmpfs          tmpfs     582M  2.5M  580M   1% /run
/dev/sda1      ext4      106G   25G   76G  25% /
tmpfs          tmpfs     2.9G  930M  2.0G  32% /dev/shm
tmpfs          tmpfs     5.0M     0  5.0M   0% /run/lock
tmpfs          tmpfs     2.9G     0  2.9G   0% /sys/fs/cgroup
tmpfs          tmpfs     582M   24K  582M   1% /run/user/119
tmpfs          tmpfs     582M  116K  582M   1% /run/user/1000
/dev/sdb1      ext4       88M  1.6M   79M   2% /media/carol/part1
/dev/sdb3      ext4       82M  1.6M   74M   3% /media/carol/part3
/dev/sdb2      ext4       89M  1.9M   81M   3% /media/carol/part2
/dev/sdc1      fuseblk   299G  223G   76G  75% /media/carol/Samsung Externo
```

ファイルシステムのタイプがわかれば、その情報で出力をフィルタリングできます。`-t TYPE` オプションを指定すると指定したタイプだけを、`-x TYPE` オプションを指定すると指定したタイプ以外を表示します。次の例を見てみましょう。

`tmpfs` ファイルシステム以外を表示します。

```
$ df -hx tmpfs
```

```
Filesystem      Size  Used Avail Use% Mounted on
udev            2.9G     0  2.9G   0% /dev
/dev/sda1       106G   25G   76G  25% /
/dev/sdb1        88M  1.6M   79M   2% /media/carol/part1
/dev/sdb3        82M  1.6M   74M   3% /media/carol/part3
/dev/sdb2        89M  1.9M   81M   3% /media/carol/part2
/dev/sdc1       299G  223G   76G  75% /media/carol/Samsung Externo
```

`ext4` ファイルシステムだけを表示します。

```
$ df -ht ext4
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       106G   25G   76G  25% /
/dev/sdb1        88M  1.6M   79M   2% /media/carol/part1
/dev/sdb3        82M  1.6M   74M   3% /media/carol/part3
/dev/sdb2        89M  1.9M   81M   3% /media/carol/part2
```

`df` で `--output=` オプションを指定すると、何をどの順番で表示するかをカスタマイズできます。`=` の後には出力する項目を `,`（カンマ）区切りで順番に書きます。以下の項目があります。

`source`
ファイルシステムが置かれているデバイス名

`fstype`
ファイルシステムのタイプ

`size`
ファイルシステムのサイズ

`used`
ファイルシステムの使用量

`avail`
ファイルシステムの空き容量

`pcent`
ファイルシステムの使用率

`target`
マウントポイント（ファイルシステムがどこにマウントされているか）

マントポイント、デバイス名、ファイルシステムタイプ、使用率を表示するなら次のコマンドを実行します。

```
$ df -h --output=target,source,fstype,pcent
Mounted on                    Filesystem     Type     Use%
/dev                          udev           devtmpfs   0%
/run                          tmpfs          tmpfs      1%
/                             /dev/sda1      ext4      25%
/dev/shm                      tmpfs          tmpfs     32%
/run/lock                     tmpfs          tmpfs      0%
/sys/fs/cgroup                tmpfs          tmpfs      0%
/run/user/119                 tmpfs          tmpfs      1%
/run/user/1000                tmpfs          tmpfs      1%
/media/carol/part1            /dev/sdb1      ext4       2%
/media/carol/part3            /dev/sdb3      ext4       3%
/media/carol/part2            /dev/sdb2      ext4       3%
/media/carol/Samsung Externo  /dev/sdc1      fuseblk   75%
```

`--output=` オプションの後に以下の項目を指定すると、inodeの情報を表示できます。

`itotal`
: ファイルシステムのinode総数

`iused`
: ファイルシステムの使用済みinode数

`iavail`
: ファイルシステムの空きinode数

`ipcent`
: ファイルシステムのinode使用率

デバイス名とタイプに加えて、inodeの総数、使用数、使用率を表示するには、次のコマンドを実行します。

```
$ df --output=source,fstype,itotal,iused,ipcent
Filesystem     Type      Inodes  IUsed IUse%
udev           devtmpfs  735764    593    1%
tmpfs          tmpfs     744858   1048    1%
/dev/sda1      ext4     7069696 318651    5%
tmpfs          tmpfs     744858    222    1%
tmpfs          tmpfs     744858      3    1%
```

```
tmpfs           tmpfs      744858      18    1%
tmpfs           tmpfs      744858      22    1%
tmpfs           tmpfs      744858      40    1%
```

ext2、ext3、ext4ファイルシステムの保守

ファイルシステムの整合性を確認し、必要に応じて修復するには `fsck`（FileSystem ChecK）コマンドを使用します。ファイルシステムが正常であることはシステムにとって極めて重要ですから、実行条件や頻度はディストリビューションによって異なりますが、Linuxの起動時には必要に応じて自動的にfsckを実行する仕組みが備わっています。すなわち、手動でfsckを実行することは多くありませんが、システム管理者は何ができるかを知っておく必要があります。

`fsck` でファイルシステムの整合性を確認するには、コマンドの引数にチェック対象のデバイス名を指定して実行します。

```
# fsck /dev/sdb1
fsck from util-linux 2.33.1
e2fsck 1.44.6 (5-Mar-2019)
DT_2GB: clean, 20/121920 files, 369880/487680 blocks
```

WARNING

マウントされているファイルシステムに対して `fsck` などのファイルシステムチェック関連コマンドを実行してはいけません。必ずアンマウントしてからコマンドを実行します。ルートファイルシステムはアンマウントできませんから、レスキューモードでシステムをブートして作業を行います。多くの場合は、インストールメディアからシステムをブートし、メニューからレスキューモードを選択します。

`fsck` はファイルシステムのタイプに応じて適切なプログラムを呼び出すラッパーコマンドです。上の例では、ファイルシステムのタイプを指定していないので、ext2/3/4に対応する `e2fsck` をデフォルトで呼び出します。

ファイルシステムを指定するには、`fsck -t vfat /dev/sdc` のように、`-t` オプションに続けてファイルシステムのタイプ名を指定します。あるいは、ファイルシステム固有のユーティリティを直接呼び出しても構いません。例えば、FATファイルシステムなら、`fsck.msdos` を実行するということです。

TIP

ターミナルで `fsck` と入力してから `Tab` を2回押すと、（シェルのコマンド名補完機能によって）システムにインストールされているファイルシステム固有のユーティリティの一覧が表示されます。

`fsck` のオプションのうち、利用頻度が高いものを紹介します。

-A

`/etc/fstab` に記載されているすべてのファイルシステムをチェックします。

-C

ファイルシステムのチェック中にプログレスバーを表示します。ext2/3/4でしかプログレスバーは表示されません。

-N

呼び出すファイルシステムに固有のプログラム名を表示します。

-R

`-A` と組み合わせて指定すると、ルートファイルシステムのチェックをスキップします。

-V

冗長モードです。通常よりも表示される情報が増えます。

ext2、ext3、ext4ファイルシステムに固有のユーティリティは `e2fsck` です（`fsck.ext2`、`fsck.ext3`、`fsck.ext4` はどれも `e2fsck` へのシンボリックリンクです）。デフォルトでは対話モードになります。ファイルシステムのエラーが見つかれば、そこでいったん停止して、ユーザーの入力を待ちます。問題を修復するなら `y`、そのままにするなら `n`、見つかった問題をすべて修復するなら `a` と入力します。

サイズの大きなファイルシステムのチェックをするときに、入力を求められるまでターミナルの前で待つのは時間の無駄です。以下のオプションを指定すると、`e2fsck` を非対話モードで実行します。

-p

見つかったエラーの自動修復を試みます。システム管理者の介在が必要なエラーが見つかった場合には、その説明を表示して終了します。

-y

すべての問いに対して `y`（yes）を選択します。

-n

`-y` とは逆に、すべての問いに対して `n`（no）を選択します。ファイルシステムを読み取り専用で開くので、修復は行われません。

-f

ファイルシステムがcleanな場合（正常にアンマウントされた場合）でもファイルシステムのチェックを行います。

extファイルシステムの微調整

ext2、ext3、ext4ファイルシステムには必要に応じて調整できるパラメータがあります。パラメータの表示や変更を行うユーティリティは `tune2fs` です。

ファイルシステムの現在のパラメータを表示するには、`tune2fs` コマンドに `-l` オプションとデバイス名を指定して実行します。次の例では最初のディスクの最初のパーティション（`/dev/sda1`）のパラメータを表示しています。

```
# tune2fs -l /dev/sda1
tune2fs 1.44.6 (5-Mar-2019)
Filesystem volume name:   <none>
Last mounted on:          /
Filesystem UUID:          6e2c12e3-472d-4bac-a257-c49ac07f3761
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype
needs_recovery extent 64bit flex_bg sparse_super large_file huge_file dir_nlink
extra_isize metadata_csum
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              7069696
Block count:              28255605
Reserved block count:     1412780
Free blocks:              23007462
Free inodes:              6801648
First block:              0
Block size:               4096
Fragment size:            4096
Group descriptor size:    64
Reserved GDT blocks:      1024
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         8192
Inode blocks per group:   512
Flex block group size:    16
Filesystem created:       Mon Jun 17 13:49:59 2019
Last mount time:          Fri Jun 28 21:14:38 2019
Last write time:          Mon Jun 17 13:53:39 2019
Mount count:              8
Maximum mount count:      -1
Last checked:             Mon Jun 17 13:49:59 2019
```

```
Check interval:           0 (<none>)
Lifetime writes:          20 GB
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:           256
Required extra isize:     32
Desired extra isize:      32
Journal inode:            8
First orphan inode:       5117383
Default directory hash:   half_md4
Directory Hash Seed:      fa95a22a-a119-4667-a73e-78f77af6172f
Journal backup:           inode blocks
Checksum type:            crc32c
Checksum:                 0xe084fe23
```

extファイルシステムでは マウント回数（mount count） が記録されていて、最大マウント回数（maximum mount count） に達すると、システム起動時に `e2fsck` が実行されてファイルシステムの整合性がチェックされます（訳注：上の例のように最大マウント回数（maximum mount count）が-1に設定されていると、マウント回数のチェックが行われなくなります。現在、主要なディストリビューションのほとんどは、マウント回数によるチェックを行いません）。

`-c N` オプションは最大マウント回数をNに設定し、`-C N` オプションは現在のマウント回数をNに設定します（`N` の部分には設定したい数値を指定します）。

ファイルシステムのチェックを行うタイミングを、期間で定めることもできます。`-i` オプションの後に数値と期間の単位を表す文字（日は `d`、月は `m`、年は `y`）を指定します。例えば、`-i 10d` と指定すると、10日ごとにファイルシステムをチェックします。数値に0を指定すると、一定期間ごとにファイルシステムのチェックを行うことはなくなります。

`-L` オプションの後に16文字以下の文字列を指定すると、ファイルシステムにボリューム名（ラベル）を付けられます。また `-U` オプションの後に16進数表記の128ビットの数値であるUUIDを指定すると、ファイルシステムのUUIDを設定できます。上の例でUUIDは `6e2c12e3-472d-4bac-a257-c49ac07f3761` です。ボリューム名とUUIDは、ファイルシステムをマウントする際に、`/dev/sda1` のようなデバイス名の代わりに使えます。

`-e BEHAVIOUR` オプションで、ファイルシステムのエラーが見つかったときにカーネルがどう対処するかを指定します。3種類の対処法があります。

`continue`
エラーが見つからなかったときと同じように実行を継続します。

`remount-ro`
読み取り専用（read-only）で再マウントします。

`panic`
カーネルパニックを発生させます。

デフォルトは `continue` です。特に重要なデータを扱う場合は、エラーが発生したときに被害が拡大しないよう `remount-ro` にしてもよいかもしれません。いずれの場合も、次回の起動時にe2fsckによってファイルシステムの整合性がチェックされます。

ext3ファイルシステムは、ext2ファイルシステムにジャーナリング機能を追加したもので、上位互換性があります。`tune2fs` コマンドで、ext2ファイルシステムをext3ファイルシステムに変換できます。`tune2fs` コマンドで `-j` オプションを指定し、ファイルシステムに対応するデバイス名を入力して実行します。例を示します。

```
# tune2fs -j /dev/sda1
```

このコマンドを実行するとファイルシステムタイプが ext3 に変更されますから、マウント時には ext2 ではなく、ext3 を指定してください（トピック104.3で解説します）。

ジャーナリングファイルシステムでは、`-J` オプションでジャーナルに関するパラメータを変更できます。ジャーナルサイズを変更するには、`-J size=` に続けてMB単位でサイズを指定します。ジャーナルを置く位置（ディスクブロック）を変更するには、`-J location=` に続けてブロック番号、あるいはファイルシステム先頭からの位置を `M` や `G` などの単位を付けて指定します。また、ジャーナルを置くディスクデバイスを変更するには、`-J device=` に続けてデバイス名を指定します。

これらのパラメータは、`,`（カンマ）で区切って複数を指定できます。例えば、`-J size=10,location=100M,device=/dev/sdb1` を実行すると、`/dev/sdb1` デバイスの先頭から100MBの位置に10MBのサイズのジャーナルを作成します。

WARNING

`tune2fs` でファイルシステムを変更（ないし変換）する前には、必ずバックアップを取り、変換後にはe2fsckで正常性を確認してください。`tune2fs` による処理には長時間かかるものがありますが、途中で処理を中断するとファイルシステムはほぼ確実に壊れます。実際のところ、バックアップを（tar等で）取り、ファイルシステムを（mkfsで）作り直し、バックアップからリストアした方が安全です。

XFSファイルシステムの保守

XFSファイルシステムで `fsck` に相当するコマンドは `xfs_repair` です。ファイルシステムに問題があると感じたら、まずこのコマンドを実行してスキャンしてみてください。

`xfs_repair` で `-n` オプションに続けてデバイス名を指定するとスキャンできます。`-n`

オプションはno
modify（修正しない）に由来しています。ファイルシステムのチェックをしてエラーの報告をしますが、修復は行いません。

```
# xfs_repair -n /dev/sdb1
Phase 1 - find and verify superblock...
Phase 2 - using internal log
        - zero log...
        - scan filesystem freespace and inode maps...
        - found root inode chunk
Phase 3 - for each AG...
        - scan (but do not clear) agi unlinked lists...
        - process known inodes and perform inode discovery...
        - agno = 0
        - agno = 1
        - agno = 2
        - agno = 3
        - process newly discovered inodes...
Phase 4 - check for duplicate blocks...
        - setting up duplicate extent list...
        - check for inodes claiming duplicate blocks...
        - agno = 1
        - agno = 3
        - agno = 0
        - agno = 2
No modify flag set, skipping phase 5
Phase 6 - check inode connectivity...
        - traversing filesystem ...
        - traversal finished ...
        - moving disconnected inodes to lost+found ...
Phase 7 - verify link counts...
No modify flag set, skipping filesystem flush and exiting.
```

エラーが見つかったときは、`-n` オプションを指定せずに実行してエラーを修復します。つまり、`xfs_repair /dev/sdb1` のようなコマンドを実行するということです。

`xfs_repair` には様々なオプションがあります。以下で紹介します。

`-l LOGDEV`、`-r RTDEV`

ログセクション（ジャーナルが置かれる）やリアルタイムセクション（ストリーミング用のリアルタイムデータが置かれる）が外部デバイスに置かれている場合に、それぞれのデバイスを指定します。`LOGDEV` と `RTDEV` の部分にデバイス名を指定します（訳注：リアルタイム機能は一般に利用されません）。

`-m N`

`xfs_repair` が使うメモリの量を `N` メガバイトに制限します。サーバー環境ではメモリを使いすぎてサーバー機能に支障が出ないように制限をかけたほうがよいことがあります。マニュアルによると、デフォルトでは `xfs_repair` がシステムの物理メモリの75%まで使うことがあるとのことです。

`-d`

読み取り専用でマウントされたファイルシステムを修復するdangerous（危険な）モードです。ルートファイルシステムの修復を行う場合などに使用することがあります。

`-v`

冗長モードです。`-v` の数を増やすとそれだけverbosity（冗長さ）が増します（例えば、`-v -v` は `-v` よりも多くの情報を出力します）。

`xfs_repair` では、同じCPUアーキテクチャを持つマシンでしかdirty（汚い）ログ（メタデータの変更を含むログ）を持つファイルシステムを修復することができません（エラーが表示されます）。最後の手段として、`-L` オプションで破損したログをクリアすることができますが、ファイルシステムが破損してデータを失う可能性が高くなります。

XFSは1990年代の商用UNIXに起源を持つファイルシステムであり、独特の機能やツールが多く備わっています。たとえば、ファイルシステムの管理構造を詳細に分析して変更する `xfs_db` や、ファイルが連続したブロックに格納されるように並び替えて（デフラグ）シーケンシャルアクセスを高速化する `xfs_fsr` などがあります。（いずれもLPICの出題範囲を超えています。）

演習

1. `du` コマンドでカレントディレクトリにあるファイルのディスク使用量の合計だけを表示してください。

2. `df` コマンドで、全てのext4ファイルシステムを対象として、デバイス名、マウントポイント、inode総数、空きinode数、ファイルシステムの空き容量の割合（使用率）を、この順番に表示してください（訳注：ファイルシステムの空き容量の割合を知りたいときは、ファイルシステムの使用率を表示して、100%からその値を引いて求めます）。

3. `e2fsck` コマンドを、`/dev/sdc1` のファイルシステムを対象として、自動的にエラーを修復するように非対話モードで実行してください。

4. `/dev/sdb1` はext2ファイルシステムです。このファイルシステムをext3に変換し、マウント回数をリセットして、ボリューム名を `UserData` に変更するコマンドを実行してください。

5. XFSファイルシステムを、修復せずにエラーチェックしてください。

発展演習

1. `/dev/sda1` にext4ファイルシステムがあり、`tune2fs -l /dev/sda1` の実行結果（一部抜粋）は次のとおりでした。

```
Mount count:              8
Maximum mount count:      -1
```

   `tune2fs -c 9 /dev/sda1` を実行すると、次回のシステム起動時に何が起こるでしょうか。

2. `du -h` を実行すると次のように表示されました。

```
$ du -h
216K    ./somedir/anotherdir
224K    ./somedir
232K    .
```

   カレントディレクトリにあるファイルのディスク使用量はいくらですか？　また、カレントディレクトリにあるファイルのディスク使用量を明示するコマンドを実行してください。

3. 次のコマンドを実行すると、`/dev/sdb1` のext2ファイルシステムはどうなりますか？

```
# tune2fs -j /dev/sdb1 -J device=/dev/sdc1 -i 30d
```

4. `/dev/sda1` にあるXFSファイルシステムのエラーチェックを実行してください。ログセクションは `/dev/sdc1` にあります。修復はせずにチェックだけを行います。

5. `df` の `-T` オプションと `-t` オプションはどう違いますか？

まとめ

このレッスンでは、次のことを学びました。

- ファイルシステムの使用量と空き容量の確認方法
- `df` の出力項目の調整方法
- `fsck` と `e2fsck` によるファイルシステムの整合性チェックと修復方法
- `tune2fs` によるextファイルシステムの調整方法
- `xfs_repair` によるXFSファイルシステムのチェックと修復方法

このレッスンでは、次のコマンドを説明しました。

`du`
指定したディレクトリ以下のディスク使用量を確認します。

`df`
ファイルシステムの空き容量を確認します。

`fsck`
ファイルシステムをチェックして修復します（ファイルシステムのタイプに応じて適切なプログラムを呼び出すラッパーコマンドです）。

`e2fsck`
ext2/3/4ファイルシステムをチェックして修復します。

`tune2fs`
ext2/3/4ファイルシステムのパラメータを調整します。

`xfs_repair`
XFSファイルシステムをチェックして修復します。

演習の解答

1. `du` コマンドでカレントディレクトリにあるファイルのディスク使用量の合計だけを表示してください。

   サブディレクトリを含まないカレントディレクトリ内のファイル使用量の合計を表示するために `-S` オプションを指定します。`-d 0` オプションを指定して、表示する最大深度を0（サブディレクトリを表示しない）にします。人間が読みやすい（human readable）単位で表示するよう、`-h` オプションも指定します。まとめると、次のコマンドになります。

   ```
   $ du -S -h -d 0
   ```

   または

   ```
   $ du -Shd 0
   ```

2. `df` コマンドで、全てのext4ファイルシステムを対象として、デバイス名、マウントポイント、inode総数、空きinode数、ファイルシステムの空き容量の割合（使用率）を、この順番に表示してください（訳注：ファイルシステムの空き容量の割合を知りたいときは、ファイルシステムの使用率を表示して、100%からその値を引いて求めます）。

   `-t` オプションの後にファイルシステムのタイプを指定するとともに、`--output=source,target,itotal,iavail,pcent` と指定して出力を調整します。次のコマンドになります。

   ```
   $ df -t ext4 --output=source,target,itotal,iavail,pcent
   ```

3. `e2fsck` コマンドを、`/dev/sdc1` のファイルシステムを対象として、自動的にエラーを修復するように非対話モードで実行してください。

   `-p` オプションを指定すると自動的にエラーを修復するので、次のコマンドを実行します。

   ```
   # e2fsck -p /dev/sdc1
   ```

4. `/dev/sdb1` はext2ファイルシステムです。このファイルシステムをext3に変換し、マウント回数をリセットして、ボリューム名を `UserData` に変更するコマンドを実行してください。

   ext2ファイルシステムからext3ファイルシステムへの変換は、ジャーナリング機能を追

加するということなので、`-j` オプションを指定します。マウント回数をリセットするオプションは `-C 0` です。ボリュームを変更するオプションは `-L UserData` です。まとめると、次のコマンドになります。

```
# tune2fs -j -C 0 -L UserData /dev/sdb1
```

5. XFSファイルシステムを、修復せずにエラーチェックしてください。

   `xfs_repair -n /dev/sdb1` のように `-n` オプションを指定して、その後にデバイス名を指定します。

発展演習の解答

1. `/dev/sda1` にext4ファイルシステムがあり、`tune2fs -l /dev/sda1` の実行結果（一部抜粋）は次のとおりでした。

```
Mount count:              8
Maximum mount count:      -1
```

   `tune2fs -c 9 /dev/sda1` を実行すると、次回のシステム起動時に何が起こるでしょうか。

   このコマンドはファイルシステムの最大マウント回数を9に設定します。現在のマウント回数は8ですから、次回のシステム起動時に `e2fsck` が自動的に起動してファイルシステムのチェックを行います。

2. `du -h` を実行すると次のように表示されました。

```
$ du -h
216K    ./somedir/anotherdir
224K    ./somedir
232K    .
```

   カレントディレクトリにあるファイルのディスク使用量はいくらですか？ また、カレントディレクトリにあるファイルのディスク使用量を明示するコマンドを実行してください。

   合計の使用量が232Kで、`somedir` サブディレクトリ（その中のサブディレクトリも含む）の使用量が224Kですから、232K−224K＝8Kがカレントディレクトリのファイルによるディスク使用量です。`-S` オプションを指定して実行すると、サブディレクトリを含まないカレントディレクトリ内のファイルによる使用量の合計を表示します。

3. 次のコマンドを実行すると、`/dev/sdb1` のext2ファイルシステムはどうなりますか？

```
# tune2fs -j /dev/sdb1 -J device=/dev/sdc1 -i 30d
```

   `/dev/sdb1` のファイルシステムにジャーナリング機能が追加されて、ext3に変換されます。ジャーナルは `/dev/sdc1` に保存され、ファイルシステムのチェックが30日ごとに行われます。

4. `/dev/sda1` のXFSファイルシステムのエラーチェックを実行してください。ログセクションは `/dev/sdc1` にあります。修復はせずにチェックだけを行います。

`xfs_repair` コマンドで、ログセクションの場所を `-l /dev/sdc1` オプションで指定し、修復はせずにチェックだけを行う `-n` オプションを指定します。まとめると、次のコマンドになります。

```
# xfs_repair -l /dev/sdc1 -n /dev/sda1
```

5. `df` の `-T` オプションと `-t` オプションはどう違いますか？

   `-T` オプションは各ファイルシステムのタイプも表示するようにします。`-t` オプションはその後に続けて指定したタイプのファイルシステムだけを表示するフィルターです。

104.3 ファイルシステムのマウントとアンマウント

LPI目標への参照

LPIC-1 version 5.0, Exam 101, Objective 104.3

総重量

3

主な知識分野

- ファイルシステムの手動マウントとアンマウント。
- 起動時のファイルシステムのマウントを設定する。
- ユーザがマウント可能なリムーバブルファイルシステムを設定する。
- ファイルシステムの特定とマウントに必要な、ラベルとUUIDの利用。
- systemdマウントユニットに知識。

用語とユーティリティ

- `/etc/fstab`
- `/media/`
- `mount`
- `umount`
- `blkid`
- `lsblk`

104.3 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	104 デバイス、Linuxファイルシステム、ファイルシステム階層標準
Objective:	104.3 ファイルシステムのマウントとアンマウントを制御する
Lesson:	1 of 1

はじめに

ここまでのレッスンで、ディスクをパーティションに分割する方法と、ファイルシステムを作成して保守する方法を学びました。このレッスンでファイルシステムを　マウント　する方法を学べば、Linuxでファイルシステムにアクセスできるようになります。

マウントするということは、システムのディレクトリツリーの特定の枝（マウントポイント）に、別のデバイス上のファイルシステムを接ぎ木（アタッチ）するということです。ファイルシステムをマウントする方法は、手動の方法にせよ自動の方法にせよ、いろいろあります。このレッスンでその方法を学習します。

ファイルシステムのマウントとアンマウント

ファイルシステムを手動でマウントするコマンドは `mount` で、次のように実行します。

```
mount -t TYPE DEVICE MOUNTPOINT
```

それぞれの項目について説明します。

TYPE
マウントされるファイルシステムのタイプ（例：ext4, btrfs, exfat）

DEVICE
ファイルシステムが収められたデバイス名（例：`/dev/sdb1`）

MOUNTPOINT
ファイルシステムがマウントされる場所。空のディレクトリでなくても構いませんが、ディレクトリは既に存在していなくてはなりません。別のファイルシステムがマウントされている間、そのディレクトリ内に元からあったファイルやディレクトリにはアクセスできなくなります。

例えば、USBフラッシュドライブ `/dev/sdb1` にあるexFATファイルシステムを、現在のユーザーのホームディレクトリ直下の `flash` という名前のディレクトリにマウントするには、次のコマンドを実行します。

```
# mount -t exfat /dev/sdb1 ~/flash/
```

TIP Bashでは、`~`（チルダ）が現在のユーザーのホームディレクトリを表します。現在のユーザーが `john` なら、`~` が `/home/john` を表します。

このようにしてマウントすると、USBドライブのファイルシステムの内容が `~/flash` ディレクトリ以下に現れて、アクセスできるようになります。

```
$  ls -lh ~/flash/
total 469M
-rwxrwxrwx 1 root root 454M jul 19 09:49 lineage-16.0-20190711-MOD-quark.zip
-rwxrwxrwx 1 root root  16M jul 19 09:44 twrp-3.2.3-mod_4-quark.img
```

マウントされているファイルシステムの一覧表示

引数なしで `mount` を実行すると、現在システムにマウントされているすべてのファイルシステムを一覧表示します。システムにアタッチされたディスクだけでなく、様々な目的でメモリ上に作成されるランタイムファイルシステムがたくさんあるので、この一覧表示は長くなります。次のように、`-t` オプションに続けてファイルシステムのタイプを指定すると、そのタイプのファイルシステムだけを表示するように出力結果をフィルタリングできます。

```
# mount -t ext4
/dev/sda1 on / type ext4 (rw,noatime,errors=remount-ro)
```

`,`（カンマ）で区切って複数のタイプを指定することもできます。

```
# mount -t ext4,fuseblk
/dev/sda1 on / type ext4 (rw,noatime,errors=remount-ro)
/dev/sdb1 on /home/carol/flash type fuseblk
(rw,nosuid,nodev,relatime,user_id=0,group_id=0,default_permissions,allow_other,blks
ize=4096) [DT_8GB]
```

出力の表示は以下のフォーマットです。

```
SOURCE on TARGET type TYPE OPTIONS
```

SOURCE はデバイス名で、TARGET はマウントされている場所（マウントポイント）です。TYPE はファイルシステムのタイプで、OPTIONS はマウント時に mount コマンドに渡されたオプションです。

その他のオプション

mount コマンドのオプションは、-t 以外にもたくさんあります。利用頻度が高いものを紹介します。

-a
: /etc/fstab に記載されているすべてのファイルシステムをマウントします（詳しくは後述）。

-o または --options
: カンマ区切りのマウントオプションを指定します。これにより、ファイルシステムのマウント方法を詳細に指定します（詳しくは後述）。

-r または -ro
: ファイルシステムを読み取り専用でマウントします。

-w または -rw
: ファイルシステムを書き込み可能でマウントします。

ファイルシステムをアンマウントするには、umount コマンドに続けて、デバイス名ないしマウントポイントのいずれかを指定します。先ほど示した、/dev/sdb1 をホームディレクトリ直下の flash ディレクトリにマウントした例では、以下に示すどちらのコマンドもそのUSBドライブをアンマウントします。

```
# umount /dev/sdb1
# umount ~/flash
```

umount のオプションを紹介します。

-a
: マウントされているすべてのファイルシステム（通常は /etc/mtab に記載されています）をアンマウントします。

-f
: ファイルシステムを強制的にアンマウントします。ネットワークの問題で到達不能になったリモートファイルシステムをアンマウントする場合などに使用します。

-r
: ファイルシステムをアンマウントできない場合に、読み取り専用にします。

ファイルを開いているためにアンマウントできない場合の対処法

ファイルシステムをアンマウントするときに、target is busy というエラーメッセージが表示されることがあります。ファイルシステム内のいずれかのファイルを開いているか、いずれかのディレトリをカレントディレクトリとして動作しているプロセスがある場合にこのエラーが発生します。しかし、開いているファイルがどこにあり、どのプログラムがそのファイルを開いているのか、すぐにはわからないこともあります。

そのような場合には、lsof コマンドにデバイス名を指定して実行すれば、開いているファイルとそのファイルを開いているプロセスの一覧を確認できます。以下に例を示します。

```
# umount /dev/sdb1
umount: /media/carol/External_Drive: target is busy.

# lsof /dev/sdb1
COMMAND  PID   USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
evince  3135 carol   16r   REG   8,17 21881768 5195
/media/carol/External_Drive/Documents/E-Books/MagPi40.pdf
```

COMMAND はファイルを開いているプログラム名で、PID はそのプロセスIDです。NAME は開いているファイル名ないしディレクトリ名です。上の例では、MagPi40.pdf という名前のファイルが evince（PDFビューア）によって開かれていることがわかります。evince を終了すると、このファイルシステムをアウンマウントできるようになるはずです。

NOTE

GNOME 環境では、lsof の出力が表示される前に、以下の警告が表示されることがあります。

```
lsof: WARNING: can't stat() fuse.gvfsd-fuse file system
/run/user/1000/gvfs
```

```
Output information may be incomplete.
```

`fuse.gvfsd-fuse`（GVFS）　はGNOMEが使用する特殊なファイルシステムで、GNOMEを使用中のユーザー以外はrootであってもアクセスできません。そのため　`lsof`　はこのような警告を表示しますが、無視しても差し支えありません。

マウントポイント

ファイルシステムのマウントは、どこにでもできるのですが、慣習に従うことをお勧めします。

伝統的には、`/mnt`　以下にすべての外部デバイスをマウントしていました。CD-ROMドライブ（`/mnt/cdrom`）やフロッピーディスク（`/mnt/floppy`）のように、一般的なデバイスのためのディレクトリがあらかじめ用意されていたのです。

最近では、`/media`　が、リムーバブルメディア（外付けハードディスク、USBフラッシュドライブ、メモリーカードなど）をシステムに接続するデフォルトのマウントポイントになっています。

現在のほとんどのLinuxディストリビューションでは、リムーバブルデバイスを接続すると自動的に　`/media/USER/LABEL`　以下にマウントされるようになっています。`USER`　はユーザー名で、`LABEL`　はデバイスを識別するボリューム名（ラベル）です。例えば、`FlashDrive`というラベルのUSBフラッシュドライブをユーザー　`john`　が接続すると、`/media/john/FlashDrive/`　以下にマウントされます。（デスクトップ環境によっては挙動が異なることがあります）。

手動　でファイルシステムをマウントする際は、伝統に従って　`/mnt`　以下にマウントするとよいでしょう。

起動時にファイルシステムをマウントする

`/etc/fstab`　ファイルにはファイルシステムのマウント情報が記載されており、これに基づいて起動時にファイルシステムのマウントが行われます。`/etc/fstab`　は、ファイルシステムのマウント情報を以下に示す6つのフィールドの順番で各行に記載したテキストファイルです。

```
FILESYSTEM MOUNTPOINT TYPE OPTIONS DUMP PASS
```

それぞれの項目について説明します。

FILESYSTEM
　デバイスファイル名です。デバイスファイル名ではなく、UUIDまたはボリューム名（ラ

ベル）での指定も可能です（次節で詳しく説明します）。

MOUNTPOINT
ファイルシステムがどこにマウントされるかを示します。

TYPE
ファイルシステムのタイプです。

OPTIONS
`mount` コマンドに渡されるマウントオプションです。

DUMP
ext2/3/4ファイルシステムを `dump` コマンドによるバックアップの対象とするかどうかを示します。通常は0（バックアップの対象としない）です。（訳注：現在、`dump` コマンドはほとんど使われませんから、デフォルトで0となっていることがほとんどです。記入していない場合は0と見なされます。）

PASS
起動時のファイルシステムチェックの順序を示します。0だと起動時のファイルシステムチェックの対象になりません。（訳注：ルートファイルシステムは1，その他のファイルシステムでは0ないし2を指定するのが慣例ですが、昔のように起動時に必ずfsckを実行する必要がほぼ無くなっていますので、すべて0でも構いません。記入していない場合は0と見なされます。）

例えば、最初のディスクの最初のパーティション（`/dev/sda1`）について、次のように記載します。

```
/dev/sda1  /  ext4  noatime,errors
```

`OPTIONS` のマウントオプションは、カンマ区切りのパラメータのリストになっています。一般的なパラメータもあれば、ファイルシステムに固有のパラメータもあります。一般的なパラメータには次のようなものがあります。

`atime` および `noatime`
`atime` を指定すると、ファイルにアクセスするたびに最終アクセス日時（Access Time）が更新されます。`noatime` を指定すると、ファイルにアクセスしても最終アクセス日時を更新しません（アクセス速度が向上します）。どちらを指定しても、（`ls` などで表示される）更新日時（Modify Time）は影響をうけず、ファイルへの書き込みが行われるたびに更新されます。（訳注：デフォルトがどちらであるかは、ディストリビューションによって異なります。）

auto および **noauto**

mount -a コマンドで自動的にマウントされるか（auto）、されないか（noauto）を指定します。（訳注：システム起動時に自動的にマウントしたいファイルシステムに auto を指定します。）

defaults

default を指定すると、rw、suid、dev、exec、auto、nouser、async を指定したことになります。

dev および **nodev**

ファイルシステムにあるデバイスファイルが利用できるか（dev）、できないか（nodev）を指定します。（訳注：nodev を指定したファイルシステムでは、デバイスファイルが存在しても機能しません。セキュリティ上、（/devディレクトリがある）ルートファイルシステム以外では、nodev を指定するとよいでしょう。）

exec および **noexec**

ファイルシステム内のバイナリファイルを実行できるか（exec）、できないか（noexec）を指定します。（訳注：セキュリティ上、必要がない限り noexec を指定するとよいでしょう。）

user および **nouser**

一般ユーザーがファイルシステムをマウントできるか（user）できないか（nouser）を指定します。（訳注：user を指定すると、自動的に nodev、noexec、nosuid が指定されます。）

group

デバイスファイルの所有グループに属しているメンバーが、ファイルシステムをマウントできるようにします。（訳注： 自動的に nosuid と nodev が指定されます。）

owner

デバイスファイルの所有者が、ファイルシステムをマウントできるようにします。（訳注： 自動的に nosuid と nodev が指定されます。）

suid および **nosuid**

SUIDビットとSGIDビットを有効にするか（suid）、しないか（nosuid）を指定します（訳注：SUIDとSGIDについてはトピック104.5で解説します。セキュリティ上、必要がない限り nosuid を指定するとよいでしょう。）

ro および **rw**

ファイルシステムを読み取り専用でマウントするか（ro）、書き込み可能でマウントするか（rw）を指定します。

remount

すでにマウントされているファイルシステムを再マウントします。/etc/fstab では用いず、mount -o コマンドで使用します。例えば、すでにマウントされているパーティション /dev/sdb1 を読み取り専用で再マウントするなら、mount -o remount,ro /dev/sdb1 というコマンドを実行します。再マウントする際には、ファイルシステムのタイプを指定する必要はなく、デバイス名かマウントポイントのどちらかを指定します。

sync および async

ファイルシステムの入出力を同期で行うか（sync）、非同期で行うか（async）を指定します。通常は async がデフォルトです。フラッシュドライブやメモリーカードのように書き込み回数の上限があるメディアで sync を指定すると、そのデバイスの寿命を縮めるおそれがあります。

UUIDとボリューム名の使用

マウントする際に、デバイス名を直接指定すると問題を招くことがあります。システムに接続するタイミング等によって、デバイスに割り当てられるデバイス名が変わることがあるからです。例えば、今は /dev/sdb1 に割り当てられているUSBフラッシュドライブが、違うポートに差し込まれたり、別のUSBフラッシュドライブの後に接続したりすると、/dev/sdc1 に割り当てられるかもしれません。

デバイス名の代わりにボリューム名（ラベル）またはUUID（Universally Unique Identifier）を指定すれば、こうした問題を回避できます。どちらもファイルシステムの作成時に設定でき、手動で変更しない限り変化することはありません。

lsblk コマンドでディスクデバイスの情報を調べると、ボリューム名とUUIDを知ることができます。-f オプションに続けてデバイス名を指定します。

```
$ lsblk -f /dev/sda1
NAME FSTYPE LABEL UUID                                 FSAVAIL FSUSE% MOUNTPOINT
sda1 ext4         6e2c12e3-472d-4bac-a257-c49ac07f3761   64,9G    33% /
```

各列の意味は次のとおりです。

NAME

ディスクデバイス名です。

FSTYPE

ファイルシステムのタイプです。

LABEL

ファイルシステムのボリューム名です。

UUID
ファイルシステムのUUIDです。

FSAVAIL
ファイルシステムの空き容量です。

FSUSE%
ファイルシステムの使用率です。

MOUNTPOINT
ファイルシステムがどこにマウントされているか（マウントポイント）です。

`/etc/fstab` では、`UUID=` オプションに続けてUUIDを指定するか、`LABEL=` オプションに続けてボリューム名を指定すれば、デバイスを指定できます。つまり、

```
/dev/sda1  /  ext4  noatime,errors
```

の代わりに

```
UUID=6e2c12e3-472d-4bac-a257-c49ac07f3761  /  ext4  noatime,errors
```

や

```
LABEL=homedisk  /home ext4  defaults
```

のような指定ができます。

`mount` コマンドでも同様に、デバイス名の代わりにUUIDまたはボリューム名でデバイスを指定できます。例えば、UUIDが `56C11DCC5D2E1334` の外部NTFSディスクを `/mnt/external` にマウントするなら、次のコマンドを実行します。

```
$ mount -t ntfs UUID=56C11DCC5D2E1334 /mnt/external
```

Systemdを使用したマウント

Systemd は、多くのLinuxディストリビューションで最初に実行されるプロセス（initプロセス）で、他のプロセスを生成し、各種サービスを起動します。Systemd は、ファイルシステムのマウント（自動マウント）の管理にも利用できます。

そのためには、マウントユニット　という設定ファイルを作成します。マウントするボリュームごとにマウントユニットを作成して、`/etc/systemd/system/` に配置します。

マウントユニットは、サフィックスが　`.mount`　のテキストファイルです。基本的な形式は次のとおりです。

```
[Unit]
Description=

[Mount]
What=
Where=
Type=
Options=

[Install]
WantedBy=
```

`Description=`

`Mounts the backup disk` のようなマウントユニットの説明です。

`What=`

マウント対象のデバイスを指定します。通常は　`/dev/disk/by-uuid/VOL_UUID`　形式（`VOL_UUID`　の部分はそのボリュームのUUID）で指定しますが、デバイス名やボリューム名でも指定できます。

`Where=`

マウントポイントのフルパスです。

`Type=`

ファイルシステムのタイプです。

`Options=`

`mount` コマンドや `/etc/fstab` と同じマウントオプションです。

`WantedBy=`

依存関係の管理です。マウントユニットでは、`multi-user.target`　を指定することが多いでしょう。通常のマルチユーザー環境でシステムを起動する際に、このマウントユニットが処理されます。

先ほどの外部NTFSディスクをマウントする例であれば、次のように記載します。

```
[Unit]
Description=External data disk

[Mount]
What=/dev/disk/by-uuid/56C11DCC5D2E1334
Where=/mnt/external
Type=ntfs
Options=defaults

[Install]
WantedBy=multi-user.target
```

マウントユニットのファイル名をマウントポイントと同じ名前にしないと正常に動作しません。この例では、マウントポイントは `/mnt/external` ですから、マウントユニットのファイル名は `mnt-external.mount` にします。

そして、`systemctl` コマンドでsystemdをリロードしてから、このマウントユニットを起動します。

```
# systemctl daemon-reload
# systemctl start mnt-external.mount
```

これで外部ディスクが `/mnt/external` にマウントされて利用できるようになります。以下のように `systemctl status mnt-external.mount` コマンドを実行すると、マウント状態を確認できます。

```
# systemctl status mnt-external.mount
● mnt-external.mount - External data disk
   Loaded: loaded (/etc/systemd/system/mnt-external.mount; disabled; vendor pres
   Active: active (mounted) since Mon 2019-08-19 22:27:02 -03; 14s ago
    Where: /mnt/external
     What: /dev/sdb1
    Tasks: 0 (limit: 4915)
   Memory: 128.0K
   CGroup: /system.slice/mnt-external.mount

ago 19 22:27:02 pop-os systemd[1]: Mounting External data disk...
ago 19 22:27:02 pop-os systemd[1]: Mounted External data disk.
```

`systemctl start mnt-external.mount` は、そのマウントユニットを1回だけ処理するコマンドです。次回の起動時にも有効にしたければ、以下のように、先ほど実行したコマンドの

`start` の部分を `enable` にして実行します。

```
# systemctl enable mnt-external.mount
```

マウントユニットの自動マウント

マウントポイントにアクセスしたときに自動マウントするように設定できます。ユニットを記述した `.mount` ファイルに加えて、以下の形式の `.automount` ファイルを作成します。

```
[Unit]
Description=

[Automount]
Where=

[Install]
WantedBy=multi-user.target
```

マウントユニットと同様に、`Description=` にはそのファイルの説明を書き、`Where=` にはマウントポイントを記載します。先ほどの例を続けると、`.automount` ファイルは以下のような記載になります。

```
[Unit]
Description=Automount for the external data disk

[Automount]
Where=/mnt/external

[Install]
WantedBy=multi-user.target
```

マウントポイントと同じファイル名（この例では `mnt-external.automount`）で保存し、systemdをリロードしてから、オートマウントユニットを起動します。

```
# systemctl daemon-reload
# systemctl start mnt-external.automount
```

`/mnt/external` ディレクトリにアクセスすると、マウントユニットで指定したディスクが自動的にマウントされます。以下のコマンドを実行すれば、起動時に上記の自動マウント設定を有効にできます。

```
# systemctl enable mnt-external.automount
```

演習

1. `mount` コマンドで、`/dev/sdc1` の `ext4` ファイルシステムを、`/mnt/external` に、読み取り専用で、`noatime` および `async` オプションを指定して、マウントしてください。

2. `/dev/sdd2` のファイルシステムをアンマウントしようとしたら、`target is busy` というエラーメッセージが表示されました。そのファイルシステムのどのファイルがどのプロセスによって開かれているかを示すコマンドを実行してください。

3. `/etc/fstab` に `/dev/sdb1 /data ext4 noatime,noauto,async` というエントリがあります。`mount -a` コマンドを実行したときに、このファイルシステムはマウントされるでしょうか？ 理由とともに答えてください。

4. `/dev/sdb1` のファイルシステムのUUIDを表示するコマンドを実行してください。

5. `mount` コマンドで、`/mnt/data` にマウントされている、UUIDが `6e2c12e3-472d-4bac-a257-c49ac07f3761` のexFATファイルシステムを、読み取り専用で再マウントしてください。

6. 現在マウントされている `ext3` と `ntfs` ファイルシステムを一覧表示するコマンドを実行してください。

発展演習

1. `/etc/fstab` に `/dev/sdc1 /backup ext4 noatime,nouser,async` というエントリがあります。一般ユーザーが `mount /backup` コマンドを実行してこのファイルシステムをマウントできるでしょうか？ 理由とともに答えてください。

2. ネットワーク接続が失われたために、`/mnt/server` にマウントされているリモートファイルシステムに到達できなくなりました。このリモートファイルシステムを強制的にアウンマウントする（それができなければ読み取り専用でマウントする）コマンドを実行してください。

3. `Backup` というボリューム名を付け、defaultsオプションとバイナリの実行を禁止する設定で、btrfsを `/mnt/backup` にマウントするエントリを、`/etc/fstab` ファイルに記述してください。

4. 次に示すsystemdのマウントユニットがあります。

```
[Unit]
Description=External data disk

[Mount]
What=/dev/disk/by-uuid/56C11DCC5D2E1334
Where=/mnt/external
Type=ntfs
Options=defaults

[Install]
WantedBy=multi-user.target
```

   これと同等の `/etc/fstab` エントリを記述してください。

5. systemdで利用できるように、前の問題で示したマウントユニットを、適切なファイル名で適切な場所に配置してください。

まとめ

このレッスンでは、手動あるいは自動で、ファイルシステムをマウントしアンマウントする方法を説明しました。以下のコマンド、ディレクトリ、ファイル、指定方法を説明しました。

- `mount`（デバイスをマウントするコマンド）
- `umount`（デバイスをアンマウントするコマンド）
- `lsof`　（ファイルシステムやファイルないしディレクトリにアクセスしているプロセスを一覧表示するコマンド）
- `/mnt` および `/media`（一般的なマウントポイントとなるディレクトリ）
- `/etc/fstab`（ファイルシステムのマウント情報を記載したテキストファイル）
- `lsblk`（ファイルシステムのタイプとUUIDなどを一覧表示するコマンド）
- UUIDまたはボリューム名を指定してファイルシステムをマウントする方法
- systemdのマウントユニットを利用してファイルシステムをマウントする方法
- systemdのマウントユニットを利用してファイルシステムを自動マウントする方法

演習の解答

1. `mount` コマンドで、`/dev/sdc1` の `ext4` ファイルシステムを、`/mnt/external` に、読み取り専用で、`noatime` および `async` オプションを指定して、マウントしてください。

```
# mount -t ext4 -o noatime,async,ro /dev/sdc1 /mnt/external
```

2. `/dev/sdd2` のファイルシステムをアンマウントしようとしたら、`target is busy` というエラーメッセージが表示されました。そのファイルシステムのどのファイルがどのプロセスによって開かれているかを示すコマンドを実行してください。

   次のように、`lsof` コマンドにデバイス名を指定して実行します。

```
$ lsof /dev/sdd2
```

3. `/etc/fstab` に `/dev/sdb1 /data ext4 noatime,noauto,async` というエントリがあります。`mount -a` コマンドを実行したときに、このファイルシステムはマウントされるでしょうか？ 理由とともに答えてください。

   マウントされません。`mount -a` コマンドでマウントされないようにする `noauto` オプションが指定されているからです。

4. `/dev/sdb1` のファイルシステムのUUIDを表示するコマンドを実行してください。

   次のように、`lsblk -f` に続けてデバイス名を指定して実行します。

```
$ lsblk -f /dev/sdb1
```

5. `mount` コマンドで、`/mnt/data` にマウントされている、UUIDが `6e2c12e3-472d-4bac-a257-c49ac07f3761` のexFATファイルシステムを、読み取り専用で再マウントしてください。

   ファイルシステムがマウントされているので、ファイルシステムのタイプやUUIDのことは気にする必要がありません。次のように、`remount` オプションと `ro` オプションを指定し、マウントポイントを指定します。

```
# mount -o remount,ro /mnt/data
```

6. 現在マウントされている `ext3` と `ntfs` ファイルシステムを一覧表示するコマンドを

実行してください。

次のように、`mount -t` に続けて、ファイルシステムのタイプをカンマ区切りで指定します。

```
# mount -t ext3,ntfs
```

発展演習の解答

1. `/etc/fstab` に `/dev/sdc1 /backup ext4 noatime,nouser,async` というエントリがあります。一般ユーザーが `mount /backup` コマンドを実行してこのファイルシステムをマウントできるでしょうか？ 理由とともに答えてください。

   マウントできません。`nouser` オプションが指定されているので、一般ユーザーはこのファイルシステムをマウントできません。

2. ネットワーク接続が失われたために、`/mnt/server` にマウントされているリモートファイルシステムに到達できなくなりました。このリモートファイルシステムを強制的にアウンマウントする（それができなければ読み取り専用でマウントする）コマンドを実行してください。

   `-f` と `-r` オプションを指定して `umount` コマンドを実行します。つまり、`umount -f -r /mnt/server` を実行します。`umount -fr /mnt/server` とオプションを一括指定して実行しても構いません。

3. `Backup` というボリューム名を付け、defaultsオプションとバイナリの実行を禁止する設定で、btrfsを `/mnt/backup` にマウントするエントリを、`/etc/fstab` ファイルに記述してください。

   `LABEL=Backup /mnt/backup btrfs defaults,noexec` と記述します。

4. 次に示すsystemdのマウントユニットがあります。

   ```
   [Unit]
   Description=External data disk

   [Mount]
   What=/dev/disk/by-uuid/56C11DCC5D2E1334
   Where=/mnt/external
   Type=ntfs
   Options=defaults

   [Install]
   WantedBy=multi-user.target
   ```

   これと同等の `/etc/fstab` エントリを記述してください。

   `UUID=56C11DCC5D2E1334 /mnt/external ntfs defaults` と記述します。

5. systemdで利用できるように、前の問題で示したマウントユニットを、適切なファイル名で適切な場所に配置してください。

マウントユニットのファイル名はマウントポイントと同じにしなければならないので、`mnt-external.mount` という名前で、`/etc/systemd/system` に配置します。

104.5 ファイルのパーミッションと所有権を管理する

LPI目標への参照

LPIC-1 version 5.0, Exam 101, Objective 104.5

総重量

3

主な知識分野

- 通常のファイルと特別なファイル、ディレクトリのアクセス権を管理する。
- セキュリティを維持するために、suid、sgid、スティッキービットなどのアクセスモードを使用する。
- ファイル作成マスクを変更する方法を知っている。
- グループフィールドを使用して、グループメンバーへのファイルアクセスを許可します

用語とユーティリティ

- `chmod`
- `umask`
- `chown`
- `chgrp`

104.5 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	104 デバイス、Linuxファイルシステム、ファイルシステム階層標準
Objective:	104.5 ファイルのアクセス許可と所有権を管理する
Lesson:	1 of 1

はじめに

Linuxはマルチユーザーシステムですから、ファイルの所有者とアクセス権限を管理する仕組みが必要です。共同作業ができるように一部のファイルについては複数のユーザーがアクセスできるようにしつつ、別のファイルについては機密扱いにしてユーザーのプライバシーを確保する仕組みです。

Linuxでは、3種類のパーミッションシステムでその仕組みを実現しています。各ファイルには所有者と所有グループがあり、所有者・所有グループ・その他の3種類のパーミッションがあります。このレッスンでは、ファイルのパーミッションを確認し、その意味を把握し、パーミッションの設定や変更を行います。

ファイルとディレクトリの情報を確認する

`ls` コマンドはディレクトリの内容を一覧表示します。オプションをつけずに `ls` を実行すると、ファイル名を表示します（訳注：Linuxではディレクトリなども一種のファイルとして扱いますので、このレッスンではディレクトリなども含めて「ファイル」と表記することがあります）。

```
$ ls
```

```
Another_Directory  picture.jpg  text.txt
```

各ファイルには、タイプ、サイズ、所有者・所有グループなど、もっと多くの情報があります。`ls` に `-l`（long form）オプションをつけて実行すると、それらの情報を確認できます。

```
$ ls -l
total 536
drwxrwxr-x 2 carol carol   4096 Dec 10 15:57 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
```

このレッスンに関連するものを中心に、出力の各列の意味を説明します。

- 1列目 はファイルタイプとパーミッションです。`drwxrwxr-x` を例にとって説明します。
  - 最初の文字の `d` は、ファイルタイプを示しています。
  - 次の3文字の `rwx` は、ファイルの所有者（`u`）のパーミッションを示しています。
  - 次の3文字の `rwx` は、ファイルの所有グループ（`g`）のパーミッションを示しています。
  - 最後の3文字の `r-x` は、その他（`o`）のパーミッションを示しています。

TIP その他のパーミッションはワールドパーミッションと呼ばれることもあります。

- 3列目 と 4列目 は、所有者と所有グループです。
- 最後の 7列目 はファイル名です。

2列目 はそのファイルを指すハードリンクの数です。5列目 はファイルサイズです。 6列目 は最終更新日時です。これらはこのレッスンの内容とは直接関係しません。

ディレクトリ

`ls -l` の引数にディレクトリを指定すると、そのディレクトリの内容を表示します。

```
$ ls -l Another_Directory/
total 0
-rw-r--r-- 1 carol carol 0 Dec 10 17:59 another_file.txt
```

`ls` に `-d` オプションを指定すると、そのディレクトリ自体の情報を表示します。

```
$ ls -l -d Another_Directory/
drwxrwxr-x 2 carol carol 4096 Dec 10 17:59 Another_Directory/
```

隠しファイル

先ほど `ls -l` を実行して取得したディレクトリの内容一覧は不完全でした。

```
$ ls -l
total 544
drwxrwxr-x 2 carol carol   4096 Dec 10 17:59 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
```

このディレクトリにはあと3つのファイルがあるのですが、それらは隠されています。Linuxでは、`.`（ドット）で始まる名前のファイルは自動的に隠されます。`ls` に `-a` オプションを指定して実行すると、隠しファイルを表示します。

```
$ ls -l -a
total 544
drwxrwxr-x 3 carol carol   4096 Dec 10 16:01 .
drwxrwxr-x 4 carol carol   4096 Dec 10 15:56 ..
drwxrwxr-x 2 carol carol   4096 Dec 10 17:59 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
-rw-r--r-- 1 carol carol      0 Dec 10 16:01 .thisIsHidden
```

`.thisIsHidden` ファイルは `.` で名前が始まっているので隠しファイルです。

`.` と `..` は特別なファイルです。`.` はカレントディレクトリを、`..` は親ディレクトリを指します。Linuxでは、どのディレクトリにも `.` と `..` があります。

TIP 他のコマンドと同様に、`ls` でも複数のオプションを一括指定できますから、`ls -l -a` を `ls -la` とすることもできます。

ファイルタイプを理解する

先ほど説明したように、`ls -l` の出力の最初の文字はファイルタイプを示しています。一般的なファイルタイプは以下の3種類です。

`-` (通常ファイル)
どのようなデータも保持でき、変更、移動、コピー、削除できます。

`d` (ディレクトリ)
: 他のファイルやディレクトリをまとめて整理できます。技術的には、ディレクトリは特別なファイルの一種です。

`l` (シンボリックリンク)
: あるファイルに対する別の名前を示す、エイリアスです。

これらの3種類のファイルタイプに加えて、さらに3種類の特殊なファイルタイプがあります。このレッスンでは詳しく説明しませんが、こうしたファイルタイプがあるのだということは知っておいたほうがよいです。

`b` (ブロックデバイス)
: 1台目のハードディスク `/dev/sda` などのように、ブロック単位でデータをやり取りする物理デバイスや仮想デバイスを表します。技術的にはカーネル内のデバイスドライバとのインターフェイスです。

`c` (キャラクタデバイス)
: メインのターミナル `/dev/ttyS0` などのように、文字単位でデータをやり取りする物理デバイスや仮想デバイスを表します。技術的にはカーネル内のデバイスドライバとのインターフェイスです。

`s` (ソケット)
: 2つのプログラムの間でデータをやりとりするための通信路を表します。

WARNING	自分が何をやろうとしているのかを正確に理解していない限り、ブロックデバイス、キャラクタデバイス、ソケットのパーミッションを変更しないでください。システムが動かなくなる恐れがあります。

パーミッションを理解する

`ls -l` の出力では、ファイルタイプの直後に、`r`、`w`、`x` という順番の3文字でパーミッションが示されます。ここではその意味を説明します。`-` はパーミッションがないことを示します。

ファイルのパーミッション

`r`
: 読み取り可能（read） を表します。8進数の `4` です（詳しくは後述）。ファイルを開いてその内容を読み取れることを示すパーミッションです。

`w`
: 書き込み可能（write） を表します。8進数の `2` です。ファイルを編集し削除できることを示すパーミッションです。

x

実行可能（execute） を表します。8進数の `1` です。プログラムやスクリプトとして実行できることを示すパーミッションです。

例えば、`rw-` というパーミッションのファイルは、読み書きができてますが、実行することはできません。

ディレクトリのパーミッション

r

読み取り可能（read） を表します。8進数の `4` です。内部に存在するファイルの名前など、ディレクトリの内容を読み取れることを示すパーミッションです。そのディレクトリ内のファイルを読み取れることを示すパーミッションではありません。

w

書き込み可能（write） を表します。8進数の `2` です。そのディレクトリ内にファイルを作成したり削除したりできることを示すパーミッションです。

書き込み可能（write） のパーミッションだけではそのディレクトリ内にファイルを作成したり削除したりすることはできません。ファイルを作成したり削除したりしてそのディレクトリに変更を加えるには検索可能（`x`）のパーミッションも必要になります。

x

ファイルでは 実行可能（execute） を表しますが、ディレクトリでは 検索可能（search） を表します。8進数の `1` です。そのディレクトリの中に入れることを示すパーミッションです。そのディレクトリ内のファイル名を一覧表示するためには読み取り可能（`r`）のパーミッションが必要になります。

ディレクトリの 検索可能（`x`）は少しわかりづらいです。例えば、以下のパーミッションである `Another_Directory` という名前のディレクトリがあるとします。

```
$ ls -ld Another_Directory/
d--x--x--x 2 carol carol 4,0K Dec 20 18:46 Another_Directory
```

そして、そのディレクトリ内に、以下のパーミッションである `hello.sh` という名前のシェルスクリプトがあるとします。

```
-rwxr-xr-x 1 carol carol 33 Dec 20 18:46 hello.sh
```

ユーザー `carol` が `Another_Directory` の内容を一覧表示しようとするとエラーになります。そのディレクトリの読み取り可能（`r`）パーミッションがないからです。

```
$ ls -l Another_Directory/
ls: cannot open directory 'Another_Directory/': Permission denied
```

しかし、ユーザー carol には、Another_Directory ディレクトリの検索可能（x）パーミッションがあるので、そのディレクトリに入ることはできます。よって、ユーザー carol は、ディレクトリ内のファイルに対するパーミッションがあれば、そのファイルにアクセスできます。ユーザー carol にはスクリプト hello.sh に対する完全なパーミッション（rwx）があるので、このスクリプトを実行できます。Another_Directory ディレクトリの内容を読み取ることはできないのですが、ファイル名を知っていればそのディレクトリ内のファイルを実行できるのです。（訳注：ディレクトリの中に入ってファイルに「アクセスできる」パーミッションだと考えてください）。

```
$ sh Another_Directory/hello.sh
Hello LPI World!
```

先にも述べたように、パーミッションには、所有者・所有グループ・その他という順序があります。この順序でパーミッションがチェックされます。

ファイルを操作しようとしているユーザーが所有者であれば、所有者のパーミッションが適用されます。所有者でなく所有グループに属していれば、所有グループのパーミッションが適用されます。所有者にも所有グループにも該当しなければ、その他のパーミッションが適用されます。

そのユーザーがファイルの所有者であれば、所有者のパーミッションだけが適用されます。所有グループやその他のパーミッションのほうが緩かった（できることが多かった）としても、所有者のパーミッションが適用されるのです。

パーミッションを変更する

chmod コマンドでパーミッションを変更できます。第一引数には変更後のパーミッションを、第二引数には変更対象のファイルないしディレクトリを指定します。パーミッションを変更できるのは、所有者とシステム管理者（root）だけです。

変更後のパーミッションの記述には、2種類の表記方法（モード）があります。

1つ目は 記号表記 で、所有者・所有グループ・その他のそれぞれについて、個別にパーミッションを付与したり剥奪したりできます。2つ目は 数値表記 で、所有者・所有グループ・その他のパーミッションをまとめて設定します。

どちらの表記でも同じ結果になります。例えば、

```
$ chmod ug+rw-x,o-rwx text.txt
```

と

```
$ chmod 660 text.txt
```

は、どちらのコマンドを実行しても以下の同じ結果になります。

```
-rw-rw---- 1 carol carol  765 Dec 20 21:25 text.txt
```

それぞれの表記について詳しく見ていきましょう。

記号表記

記号表記　の最初の文字は、誰のパーミッションを変更するかを表します。所有者は　`u`、所有グループは `g`、その他は `o`、全員は `a` です。

次の記号は動作を表します。付与は `+`、剥奪は `-`、設定（セット）は `=` です。

最後の記号はパーミッションを表します。読み取り可能は　`r`、書き込み可能は　`w`、実行（検索）可能は `x` です。

例えば、以下のパーミッションの `text.txt` というファイルがあるとします。

```
$ ls -l text.txt
-rw-r--r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

所有グループに属するメンバーに書き込み可能のパーミッションを与えるなら、第一引数は　`g+w`　になります。グループ（group）の　`g`、付与の　`+`、書き込み可能（write）の　`w`　と考えます。実行するコマンドは次のとおりです。

```
$ chmod g+w text.txt
```

`ls` を実行して結果を確認してみましょう。

```
$ ls -l text.txt
-rw-rw-r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

次に、所有者の読み取り可能パーミッションを剥奪してみます。ユーザー（user）の　`u`、剥奪の　`-`、読み取り可能（read）の　`r`　と考えます。第一引数は　`u-r`　になるので、実行するコマンドは次のとおりです。

```
$ chmod u-r text.txt
$ ls -l text.txt
--w-rw-r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

全員のパーミッションを `rw-` に設定してみましょう。全員（all）の `a`、設定の `=`、読み取り可能（`r`）、書き込み可能（`w`）、実行不可（`-`）で、実行するコマンドは次のとおりです。

```
$ chmod a=rw- text.txt
$ ls -l text.txt
-rw-rw-rw- 1 carol carol 765 Dec 20 21:25 text.txt
```

`,`（カンマ）で区切ることにより、複数のパーミッションを一度に変更することもできます。

```
$ chmod u+rwx,g-x text.txt
$ ls -lh text.txt
-rwxrw-rw- 1 carol carol 765 Dec 20 21:25 text.txt
```

上の例は、所有者に読み書き実行のパーミッションを与えて（`u+rwx`）、所有グループから実行のパーミッションを剥奪する（`g-x`）と読みます。

`chmod` コマンドの第二引数にディレクトリを指定すると、そのディレクトリのパーミッションを変更します。そのディレクトリ以下のすべてのファイルとサブディレクトリのパーミッションを変更したければ、再帰オプションを指定して `chmod` を実行します。再帰オプションは `-R` で、第一引数の前に指定します。

```
$ chmod -R u+rwx Another_Directory/
```

このコマンドは、再帰的に（`-R`）、所有者に読み書き実行のパーミッションを与える（`u+rwx`）と読みます。

WARNING

`-R` の再帰オプションを指定するときには注意してください。第二引数に指定するディレクトリ内に多数のファイルやサブディレクトリがある場合には、予期せぬパーミッションの変更をしてしまいがちです。（訳注：`x` はファイルとディレクトリで意味が異なるので、結果がどうなるかをよく考えてから実行しましょう。）

数値表記

数値表記　　では、3桁の8進数でパーミッションを指定します。（訳注：読み書き実行のパーミッションの有無の組み合わせは全部で8通りなので、8進数で過不足なく表せます。）

各パーミッションには値が割り当てられています。読み取り可能（`r`）は4、書き込み可能（`w`）は2、実行（検索）可能（`x`）は1です。どのパーミッションもなければ0です。`rwx`　は`7`（4+2+1）ですし、`r-x`　は　`5`（4+0+1）です。（訳注：`rwx`　は2進数の　`111`　で8進数の　`7`、`r-x` は2進数の `101` で8進数の `5` と、2進数から考えることもできます。）

3桁の数字の最初の桁は所有者（`u`）、2桁目は所有グループ（`g`）、最後の桁はその他（`o`）のパーミッションを表します。`rw-rw----`　　というパーミッションに設定するなら、その数値表記は `660` です。

```
$ chmod 660 text.txt
$ ls -l text.txt
-rw-rw---- 1 carol carol 765 Dec 20 21:25 text.txt
```

記号表記　　と同様に、数値表記　　での第一引数は変更後のパーミッションで、第二引数はパーミッションを変更するファイルないしディレクトリです。

TIP　パーミッションの数値が 奇数 なら実行可能です。

パーミッションを特定の状態に設定したいときには　数値表記　が便利です。例えば　`rw-r-----` に設定するなら `640` です。

現在のパーミッションに関わらず何らかの変更をしたいときには　記号表記　が便利です。例えば、所有グループとその他の現在のパーミッションは変更せず、所有者にのみ実行可能パーミッションを与えるなら、`chmod u+x script.sh` を実行します。

ファイルの所有者と所有グループを変更する

ファイルないしディレクトリの所有者と所有グループを変更するコマンドは　`chown`　です。次のように記述して実行します。

```
chown USERNAME:GROUPNAME FILENAME
```

`text.txt` ファイルで試してみます。

```
$ ls -l text.txt
-rw-rw---- 1 carol carol 1881 Dec 10 15:57 text.txt
```

所有者は `carol` で、所有グループも `carol` です。所有グループを `students` に変更します。

```
$ chown carol:students text.txt
$ ls -l text.txt
-rw-rw---- 1 carol students 1881 Dec 10 15:57 text.txt
```

ファイルの所有者が所有グループに属していなくても何ら問題はありません。上の例では、ユーザー `carol` がグループ `students` に属していなくても大丈夫です。

所有者または所有グループを変更しない場合は、その部分の記述を省略できます。先ほどの例のように、所有者を変更せず所有グループを `students` に変更するなら、`chown :students text.txt` のように省略しても構いません。所有グループを変更せず所有者を `carol` に変更するなら、`chown carol: text.txt` や `chown carol text.txt` のように省略できます。所有グループを変更する専用のコマンド `chgrp` で、`chgrp students text.txt` のように実行することもできます。

ファイルの所有者や所有グループを変更できるのは、その所有者や所有グループのメンバーとシステム管理者（root）だけです。それ以外のユーザーが所有者や所有グループを変更しようとすると、`Operation not permitted` というエラーメッセージが表示されます。

グループを確認する

ファイルの所有者と所有グループを変更する前に、システムにどのようなグループが存在していて、それぞれのグループにどのユーザーが属しているか、あるいは、あるユーザーがどのグループに属しているかを把握しておきたいことがあります。

システムに存在するグループは、`getent group` を実行して確認できます。出力は以下のようになります（出力の最初の部分だけを載せています）。

```
$ getent group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,rigues
tty:x:5:rigues
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:rigues
```

あるユーザーがどのグループに属しているかを知りたければ、`groups` コマンドの引数にユーザー名を指定して実行します。

```
$ groups carol
carol : carol students cdrom sudo dip plugdev lpadmin sambashare
```

あるグループに属しているユーザーを知りたければ、`groupmems` コマンドに `-g` オプションとグループ名を指定し、`-l` オプションをつけて実行します。

```
# groupmems -g cdrom -l
carol
```

TIP `groupmems` コマンドはシステム管理者（root）しか実行できません。rootユーザーとしてログインしていない場合は、コマンドの前に `sudo` をつけて実行してください。

デフォルトのパーミッション

ディレクトリやファイルを作成したときのデフォルトのパーミッションがどうなっているかを、実験で確かめてみましょう。ターミナルで次のコマンドを実行して空のファイルを作成します。

```
$ touch testfile
```

このファイルのパーミッションを確認してみます。システムによって異なりますが、筆者の環境では次のようなパーミッションでした。

```
$ ls -lh testfile
-rw-r--r-- 1 carol carol 0 jul 13 21:55 testfile
```

`rw-r--r--` です。所有者は 読み書き可能 で、所有グループとその他は 読み取り可能です。数値表記では `644` です。

次にディレクトリを作成します。

```
$ mkdir testdir
$ ls -lhd testdir
drwxr-xr-x 2 carol carol 4,0K jul 13 22:01 testdir
```

このディレクトリのパーミッションは `rwxr-xr-x` です。所有者は 読み書き実行（検索）が可能　で、所有グループとその他は　読み取りと実行（検索）が可能　です。数値表記では `755` です。

ファイルシステムのどこでファイルやディレクトリを作成しても、同じパーミッションになります。このパーミッションはどこから来ているのでしょうか。

`umask`（user　mask）というパラメーターから来ています（訳注：環境変数と同様に親プロセスから引き継がれる「環境」の一部です）。これがデフォルトのパーミッションを決めています。`umask` コマンドを実行すると、その値を確認できます。

```
$ umask
0022
```

この　`0022`　という値は、`rw-r--r--`　はおろか、`644`　とも異なります。`-S`　オプションを指定して `umask` コマンドを実行すると、出力が記号表記になります。

```
$ umask -S
u=rwx,g=rx,o=rx
```

これはディレクトリのデフォルトのパーミッションと一致します。しかしファイルのデフォルトのパーミッションとは食い違っています。

デフォルトでファイルの実行可能パーミッションを与えるのは危険です。ディレクトリには実行（検索）可能パーミッションが必要ですが（そうでなければそのディレクトリの中に入れません）、ファイルに実行可能パーミッションは不要です。だから、ファイルにはデフォルトで実行可能パーミッションがなく、`rw-r--r--` になるのです。

`umask`　コマンドは、デフォルトのパーミッションを表示するだけでなく、現在のシェルセッションでのデフォルトのパーミッションを変更することもできます。例えば、次のように実行します。

```
$ umask u=rwx,g=rwx,o=
```

これ以降、ディレクトリを作成した場合のパーミッションは　`rwxrwx---`　になり、ファイルを作成した場合のパーミッションは　`rw-rw----`　になります。`testfile`　と　`testdir`　を作成する実験をもう一度繰り返してパーミッションを確認してみます。

```
$ ls -lhd test*
drwxrwx--- 2 carol carol 4,0K jul 13 22:25 testdir
```

```
-rw-rw---- 1 carol carol    0 jul 13 22:25 testfile
```

`-S` オプションなしで `umask` を実行すると、以下の結果になります。

```
$ umask
0007
```

`-S` オプションなしで `umask` を実行したときに出力される数値は、マスク値であり、パーミッション自体の数値表記とは異なります。以下に対応表を載せておきます。（訳注：マスク値はその名前の通り、パーミッションを 与えない ビットを示すものですから、2進数で表してビットを反転させるとデフォルトのディレクトリのパーミッションがわかります。例えば、マスク値が2なら `010` を反転させた `101`、つまり `r-x` がデフォルトのディレクトリのパーミッションだとわかります。）

値	ファイルのパーミッション	ディレクトリのパーミッション
0	rw-	rwx
1	rw-	rw-
2	r--	r-x
3	r--	r--
4	-w-	-wx
5	-w-	-w-
6	---	--x
7	---	---

`007` は `rwxrwx---` に対応し、先ほど設定したデフォルトのパーミッションと一致しています。4つの数字のうちの最初の数字はとりあえず無視してください（後で説明します）。

特別なパーミッション

所有者・所有グループ・その他について、読み取り可能（`r`）、書き込み可能（`w`）、実行（検索）可能（`x`）に加えて、3つの 特別なパーミッション があります。以下で1つずつ説明します。

スティッキービット

スティッキービットは、削除制限フラグ であり、数値表記では4桁の8進数の最初の桁の1、記号表記ではその他パーミッションの3文字目の `t` です。これは、ディレクトリに対してだけ意味を持ち、ファイルに設定しても意味がありません。スティッキービットが設定されたディレクトリ内のファイルは、そのファイルの所有者かそのスティッキービットが設定

されたディレクトリの所有者しか削除できません。（訳注：`mv` コマンドによる移動や名前の変更は、実質的に削除と等しいので、これもできません。また、rootユーザーは削除できます。）

スティッキービットが設定されたディレクトリは、`ls -l` の出力で、その他パーミッションの `x` が表示される場所に `t` が表示されます。（訳注：`/tmp` や `/var/tmp` など、さまざまなプログラムが一時ファイルを置くディレクトリに設定されています。）

```
$ ls -ld Sample_Directory/
drwxr-xr-t 2 carol carol 4096 Dec 20 18:46 Sample_Directory/
```

数値表記では、特別なパーミッションが4桁の8進数の数字の最初の桁で示されます。例えば、パーミッションが `755` である `Another_Directory` にスティッキービット（数値は `1`）を設定するなら、次のコマンドを実行します。

```
$ chmod 1755 Another_Directory
$ ls -ld Another_Directory
drwxr-xr-t 2 carol carol 4096 Dec 20 18:46 Another_Directory
```

SGID

SGID（Set Group ID）は、数値表記では4桁の8進数の最初の桁の `2`、記号表記では所有グループパーミッションの3文字目の `s` です。実行可能なファイルないしディレクトリに設定します。ファイルに設定すると、そのファイルを実行したときにその所有グループの権限で実行されます。ディレクトリに設定すると、そのディレクトリ内で作成したファイルやディレクトリの所有グループが、SGIDを設定したディレクトリの所有グループと同じになります。

SGIDが設定されたファイルやディレクトリは、`ls -l` の出力で、所有グループパーミッションの `x` が表示される場所に `s` が表示されます。

```
$ ls -l test.sh
-rwxr-sr-x 1 carol root 33 Dec 11 10:36 test.sh
```

記号表記でSGIDを設定するなら、次のコマンドを実行します。

```
$ chmod g+s test.sh
$ ls -l test.sh
-rwxr-sr-x 1 carol root     33 Dec 11 10:36 test.sh
```

以下の例を通じて、ディレクトリにSGIDを設定したときの挙動を確かめます。所有者は `carol`、所有グループは `users` の、`Sample_Directory` というディレクトリがあるとします。パーミッションは次のとおりです。

```
$ ls -ldh Sample_Directory/
drwxr-xr-x 2 carol users 4,0K Jan 18 17:06 Sample_Directory/
```

このディレクトリ内で `touch` コマンドを実行して新しい空のファイルを作成します。

```
$ cd Sample_Directory/
$ touch newfile
$ ls -lh newfile
-rw-r--r-- 1 carol carol 0 Jan 18 17:11 newfile
```

新しく作成されたファイルの所有者は `carol` で所有グループも `carol` です。もしこのディレクトリにSGIDが設定されていたら、結果は異なります。`Sample_Directory` にSGIDを設定します。

```
$ sudo chmod g+s Sample_Directory/
$ ls -ldh Sample_Directory/
drwxr-sr-x 2 carol users 4,0K Jan 18 17:17 Sample_Directory/
```

所有グループパーミッションの3文字目に `s` と表示されていることから、SGIDが設定されているとわかります。もう一度 `touch` コマンドで新しい空のファイルを作成します。

```
$ cd Sample_Directory/
$ touch emptyfile
$ ls -lh emptyfile
 -rw-r--r-- 1 carol users 0 Jan 18 17:20 emptyfile
```

新しく作成されたファイルの所有グループが `users` になっています。これがSGIDの働きです。SGIDを設定した親ディレクトリの所有グループを受け継ぐようになります。（訳注：グループでファイル共有に使用するディレクトリを作成する場合などによく使われます。）

SUID

SUID（Set User ID）は、数値表記では4桁の8進数の最初の桁の `4`、記号表記では所有者パーミッションの3文字目の `s` です。これは、ファイルに対してだけ意味を持ち、ディレクトリに設定しても意味がありません。SGIDと似て、そのファイルの所有者の権限で実行す

るようになります。`ls -l` の出力で、所有者パーミッションの `x` が表示される場所に `s` が表示されます。（訳注：`/usr/bin/passwd` コマンドなど、rootしか書き込めない設定ファイルを変更するコマンドなどに設定されます。セキュリティの抜け道になりやすいので、利用には十分な注意が必要です。）

```
$ ls -ld test.sh
-rwsr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

特別なパーミッションを複数同時に設定することもできます。パーミッションが `755` の `test.sh` に、SGID（数値は `2`）とSUID（数値は `4`）を数値表記で同時に設定するなら、次のコマンドを実行します。

```
$ chmod 6755 test.sh
```

`ls -l` の実行結果は次のとおりです。

```
$ ls -lh test.sh
-rwsr-sr-x 1 carol carol 66 Jan 18 17:29 test.sh
```

TIP

ターミナルがカラー表示に対応していれば（最近のターミナルはたいてい対応しています）、`ls -l` の出力をぱっと見て特別なパーミッションが設定されているかどうかがわかります。スティッキービットが設定されているディレクトリは名前の背景が青色になります。SGIDは背景が黄色に、SUIDは背景が赤色になります。ディストリビューションやターミナルの設定によって色が異なる場合があります。

演習問題

1. まず、`mkdir emptydir` コマンドで `emptydir` という名前のディレクトリを作成します。次に、`ls` コマンドで `emptydir` ディレクトリのパーミッションを表示してください。

2. まず、`touch emptyfile` コマンドで `emptyfile` という名前の空のファイルを作成します。次に、`chmod` コマンドを記号表記で1回だけ実行して、`emptyfile` の所有者に実行可能のパーミッションを付与し、所有者以外の書き込み可能と実行可能のパーミッションを剥奪してください。

3. `umask` の実行結果が `027` であるときに、新しく作成したファイルのパーミッションはどうなりますか？

4. 所有者、所有グループ、パーミッションが次のとおりである `test.sh` という名前のシェルスクリプトがあります。

```
-rwxr-sr-x 1 carol root     33 Dec 11 10:36 test.sh
```

   - ファイルの所有者のパーミッションはどうなっていますか？

   - `chmod` を数値表記で実行して、このファイルに設定されている特別なパーミッションを取り除いてください。

5. `ls -l` を実行すると、次のように表示されました。

```
$ ls -l /dev/sdb1
brw-rw---- 1 root disk 8, 17 Dec 21 18:51 /dev/sdb1
```

   `sdb1` のファイルタイプは何ですか？ 誰がこのファイルに書き込みできますか？

6. 以下の4つのファイル（ディレクトリ）があります。

```
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
----r--r-- 1 carol carol    0 Dec 11 10:55 foo.bar
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
drwxr-sr-x 2 carol users  4,0K Jan 18 17:26 Sample_Directory
```

   各ファイル（ディレクトリ）のパーミッションを4桁の数値表記で書いてください。

`Another_Directory`	
`foo.bar`	
`HugeFile.zip`	
`Sample_Directory`	

発展演習

1. まず、`touch emptyfile` コマンドで `emptyfile` という名前の空のファイルを作成します。次に、`chmod 000 emptyfile` コマンドで、そのファイルのパーミッションをゼロにします。`chmod 4 emptyfile` のように `chmod` の数値表記で第一引数に 1桁の数字 を渡すとどうなりますか？ `chmod 44 emptyfile` のように 2桁の数字 を渡すとどうなりますか？ これらの結果から、`chmod` が数値をどのように解釈するかを推測してください。

2. Linuxのシステムで一時ファイルを格納する `/tmp` ディレクトリのパーミッションは以下のとおりです。

```
$ ls -ld /tmp
drwxrwxrwt  19 root root  16K Dec 21 18:58 tmp
```

   所有者・所有グループ・その他はすべて読み書き実行のパーミッションがあります。一般ユーザーはこのディレクトリ内の どの ファイルでも削除できるのでしょうか？ 理由とともに答えてください。

3. `test.sh` という名前のファイルのパーミッションは `-rwsr-xr-x` で、SUIDが設定されています。以下のコマンドを実行しました。

```
$ chmod u-x test.sh
$ ls -l test.sh
-rwSr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

   これはどういうことでしょうか？ 大文字の `S` は何を表していますか？

4. まず、`Box` という名前のディレクトリを作成してください。次に、`Box` 内に作成されたファイルの所有グループが自動的に `users` になるように設定してください。最後に、`Box` 内に作成されたファイルは、その所有者しか削除できないように設定してください。

まとめ

このレッスンでは、パーミッションと所有者を管理する以下の方法を説明しました。

- `ls` コマンドでパーミッションと所有者の情報を読み取る方法。
- `chmod` コマンドで誰がファイルを読み書き実行できるかを制御する方法（数値表記と 記号表記 の2通り）。
- `chown` コマンドでファイルの所有者と所有グループを変更する方法。
- `chgrp` コマンドでファイルの所有グループを変更する方法。
- `umask` コマンドでデフォルトのパーミッションを確認し変更する方法。

このレッスンでは、次のコマンドを説明しました:

`ls`
ファイルを一覧表示します。`-l` オプションを指定するとパーミッションなどの詳細を表示します。

`chmod`
ファイルやディレクトリのパーミッションを変更します。

`chown`
ファイルやディレクトリの所有者と所有グループを変更します。

`chgrp`
ファイルやディレクトリの所有グループを変更します。

`umask`
ファイルやディレクトリのデフォルトのパーミッションを確認、設定します。

演習の解答

1. まず、`mkdir emptydir` コマンドで `emptydir` という名前のディレクトリを作成します。次に、`ls` コマンドで `emptydir` ディレクトリのパーミッションを表示してください。

   ディレクトリの中身ではなくディレクトリ自体の情報を表示するために、`ls` コマンドに `-d` オプションを指定して実行します。つまり、次のコマンドを実行します。

   ```
   ls -l -d emptydir
   ```

   `ls -ld emptydir` のように2つのオプションを一括指定してもよいです。

2. まず、`touch emptyfile` コマンドで `emptyfile` という名前の空のファイルを作成します。次に、`chmod` コマンドを記号表記で1回だけ実行して、`emptyfile` の所有者に実行可能のパーミッションを付与し、所有者以外の書き込み可能と実行可能のパーミッションを剥奪してください。

   次のように順番に考えていきます。

   - ファイルの所有者に実行可能パーミッションを付与するので、`u+x` です。
   - 所有グループとその他の書き込み可能と実行可能のパーミッションを剥奪するので、`go-wx` です。

     これらをカンマで並べて、次のコマンドを実行します。

     ```
     chmod u+x,go-wx emptyfile
     ```

3. `umask` の実行結果が `027` であるときに、新しく作成したファイルのパーミッションはどうなりますか？

   `rw-r-----` になります。

4. 所有者、所有グループ、パーミッションが次のとおりである `test.sh` という名前のシェルスクリプトがあります。

   ```
   -rwxr-sr-x 1 carol root     33 Dec 11 10:36 test.sh
   ```

   - ファイルの所有者のパーミッションはどうなっていますか？

     所有者のパーミッション（`ls -l` の出力の2文字目から4文字目）は `rwx` です。つ

まり、所有者は、そのファイルの読み書き実行ができます。

- `chmod` を数値表記で実行して、このファイルに設定されている特別なパーミッションを取り除いてください。

  `chmod` の第一引数に最初の数字が `0` の4桁の数字を指定すると、特別なパーミッションを取り除けます。このファイルの現在のパーミッションは `755` ですから、`chmod 0755` を実行します。

5. `ls -l` を実行すると、次のように表示されました。

```
$ ls -l /dev/sdb1
brw-rw---- 1 root disk 8, 17 Dec 21 18:51 /dev/sdb1
```

`sdb1` のファイルタイプは何ですか？ 誰がこのファイルに書き込みできますか？

`ls -l` の出力の最初の文字がファイルタイプを示しています。`b` は ブロックデバイス です。マシンに接続されているディスクはブロックデバイスです。パーミッションを見ると、所有者（`root`）と、所有グループ（`disk`）に属しているユーザーが、このファイル（デバイス）に書き込みできます。

6. 以下の4つのファイル（ディレクトリ）があります。

```
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
----r--r-- 1 carol carol    0 Dec 11 10:55 foo.bar
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
drwxr-sr-x 2 carol users 4,0K Jan 18 17:26 Sample_Directory
```

各ファイル（ディレクトリ）のパーミッションを4桁の数値表記で書いてください。

4桁の数値表記のパーミッションは次のとおりです。

`Another_Directory`	`1755` です。`1` はスティッキービットで、`755` は通常のパーミッションです（所有者は `rwx`、所有グループとその他は `r-x`）。
`foo.bar`	`0044` です。特別なパーミッションはないので最初の数字は `0` です。所有者のパーミッションは何もなく（`---`）、所有グループとその他には読み取り可能のパーミッションだけがあります（`r--`）。

`HugeFile.zip`	`0664` です。特別なパーミッションはないので最初の数字は `0` です。所有者と所有グループのパーミッションは読み書き可能で（`rw-`）、その他のパーミッションは読み取り可能です（`r--`）。
`Sample_Directory`	`2755` です。SGIDの `2`、所有者は `7`（`rwx`）、所有グループとその他は `5`（`r-x`）です。

発展演習の解答

1. まず、`touch emptyfile` コマンドで `emptyfile` という名前の空のファイルを作成します。次に、`chmod 000 emptyfile` コマンドで、そのファイルのパーミッションをゼロにします。`chmod 4 emptyfile` のように `chmod` の数値表記で第一引数に 1桁の数字 を渡すとどうなりますか？ `chmod 44 emptyfile` のように 2桁の数字 を渡すとどうなりますか？ これらの結果から、`chmod` が数値をどのように解釈するかを推測してください。

   `emptyfile` のパーミッションをゼロにしているので、初期状態は次のとおりです。

   ```
   ---------- 1 carol carol    0 Dec 11 10:55 emptyfile
   ```

   `chmod 4 emptyfile` を実行した結果を確認します。

   ```
   $ chmod 4 emptyfile
   $ ls -l emptyfile
   -------r-- 1 carol carol 0 Dec 11 10:55 emptyfile
   ```

   その他のパーミッションが変更されています。`chmod 44 emptyfile` のように2桁の数字を試してみましょう。

   ```
   $ chmod 44 emptyfile
   $ ls -l emptyfile
   ----r--r-- 1 carol carol 0 Dec 11 10:55 emptyfile
   ```

   所有グループ と その他 のパーミッションが設定されています。このことから、`chmod` の数値表記では、その他 から 所有者 へと、値を後ろから読み取るのだとわかります。1桁の数字を渡すと その他 のパーミッションを変更し、2桁の数字を渡すと 所有グループ と その他 のパーミッションを変更し、3桁の数字を渡すと 所有者 と 所有グループ と その他 のパーミッションを変更し、4桁の数字を渡すと特別なパーミッションと 所有者 と 所有グループ と その他 のパーミッションを変更するということです。（訳注：パーミッションは3桁ないし4桁の8進数で指定するということです。）

2. Linuxのシステムで一時ファイルを格納する `/tmp` ディレクトリのパーミッションは以下のとおりです。

   ```
   $ ls -ld /tmp
   drwxrwxrwt  19 root root  16K Dec 21 18:58 tmp
   ```

所有者・所有グループ・その他はすべて読み書き実行のパーミッションがあります。一般ユーザーはこのディレクトリ内の　どの　ファイルでも削除できるのでしょうか？　理由とともに答えてください。

`/tmp`　は誰でも書き込みができるディレクトリです。しかし、別のユーザーが作成したファイルを削除できるのは好ましくないので、スティッキービット　が設定されています（その他　パーミッションの3文字目の　`t`　がスティッキービットを示しています）。よって、一般ユーザーは、`/tmp`　内のファイルのうち、自分が作成したファイルしか削除できません。

3. `test.sh`　という名前のファイルのパーミッションは　`-rwsr-xr-x`　で、SUIDが設定されています。以下のコマンドを実行しました。

```
$ chmod u-x test.sh
$ ls -l test.sh
-rwSr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

これはどういうことでしょうか？　大文字の `S` は何を表していますか？

ファイルの所有者の実行可能パーミッションを剥奪しました。`s`　（や　`t`）は　`ls -l`　の出力の　`x`　の位置に表示されるので、実行可能パーミッションの有無を示す何らかの方法が必要になります。特別なパーミッションを示す文字が小文字か大文字かで実行可能パーミッションの有無を示します。

所有者パーミッションの小文字の　`s`　は、ファイルの所有者に実行可能パーミッションがあり、SUIDが設定されていることを示します。大文字の　`S`　は、ファイルの所有者に実行可能パーミッションがなく（SUIDが設定されていなければ　`-`　と表示される状態です）、SUIDが設定されていることを示します。

SGIDでも同様です。所有グループパーミッションの小文字の　`s`　は、所有グループに実行可能パーミッションがあり、SGIDが設定されていることを示します。大文字の　`S`　は、所有グループに実行可能パーミッションがなく（SGIDが設定されていなければ　`-`　と表示される状態です）、SGIDが設定されていることを示します。

スティッキービットでも同様です。その他パーミッションの小文字の　`t`　は、その他に実行可能パーミッションがあり、スティッキービットが設定されていることを示します。大文字の　`T`　は、その他に実行可能パーミッションがなく、スティッキービットが設定されていることを示します。

4. まず、`Box`　という名前のディレクトリを作成してください。次に、`Box`　内に作成されたファイルの所有グループが自動的に　`users`　になるように設定してください。最後に、`Box`　内に作成されたファイルは、その所有者しか削除できないように設定してください。

順番に考えます。まず以下のコマンドを実行してディレクトリを作成します。

```
$ mkdir Box
```

このディレクトリ内に作成されたファイルの所有グループが自動的に `users` になるようにしたいです。そのためには、このディレクトリの所有グループを `users` にして、SGIDを設定します。所有グループがこのディレクトリ内にファイルを作成できるようにすることも忘れないでください。

所有者とその他のパーミッションは関係ないので、記号表記を用います。以下のコマンドを実行します。

```
$ chown :users Box/
$ chmod g+wxs Box/
```

このコマンドを実行するユーザーが `users` グループに属していなければ、`sudo` をつけてroot権限で上記のコマンドを実行しなければなりません。

最後に、ファイルを作成したユーザーだけがそのファイルを削除できるようにします。このディレクトリに `t` で示されるスティッキービットを設定します。スティッキービットはその他（`o`）のパーミッションに設定するので、次のコマンドを実行します。

```
$ chmod o+t Box/
```

`Box` ディレクトリのパーミッションは、以下のようになるはずです。

```
drwxrwsr-t 2 carol users  4,0K Jan 18 19:09 Box
```

次のコマンドのように、一度の `chmod` の実行でSGIDとスティッキービットを設定してもよいです。

```
$ chmod g+wxs,o+t Box/
```

104.6 ハードリンクとシンボリックリンクの作成と変更

LPI目標への参照

LPIC-1 version 5.0, Exam 101, Objective 104.6

総重量

2

主な知識分野

- リンクを作成する。
- ハードおよび/またはソフトリンクを特定する。
- コピーとファイルのリンク。
- リンクを使用してシステム管理タスクをサポートする。

用語とユーティリティ

- `ln`
- `ls`

Linux Professional Institute

104.6 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	104 デバイス、Linuxファイルシステム、ファイルシステム階層標準
Objective:	104.6 ハードリンクとシンボリックリンクの作成と変更
Lesson:	1 of 1

はじめに

Linuxにはリンクという特別な種類のファイルがあります。このレッスンでは、リンクとは何かを説明し、リンクを移動させるとどうなるかなどを学びます。

リンクを理解する

前のレッスンで説明したように、Linuxではすべてがファイルとして扱われます。ファイルの中には、リンク という特別な種類のファイルがあります。リンクには2種類あります。

シンボリックリンク
ソフトリンク　とも呼ばれます。別のファイルのパスを指し示します。指し示されるファイル（ターゲット）を削除すると、シンボリックリンクが存在していても、指し示されるものがなくなっているので、機能しなくなります。

ハードリンク
元のファイルの2つ目の名前だと考えてください。コピー（複製）ではありません。ディスク上の同じ場所（inode）を指す別の項目（エントリ）です。

TIP　inode　とは、ファイルやディレクトリなどの属性（メタデータ）を格納した

> データ構造です。属性には、パーミッション、所有者・所有グループ、ファイル本体を収めたディスクブロックの位置などがあります。inodeを参照すると、ファイル名（パス名）以外のファイルに関する（データの位置を含む）すべての情報がわかりますから、ファイルそのものを表しているデータ構造であるといえます。すべてのファイルは1つのinodeに結びつけられていて、inodeのリスト（inodeテーブル）はディスク上の特別な位置に置かれています。いわば、inodeテーブルはファイルシステム内の全てのファイルに対する索引（index）であり、index nodeがinodeという名前の由来です。

ハードリンク

ハードリンクを作成する

ハードリンクを作成するコマンドは `ln` です。次のように実行します。

```
$ ln TARGET LINK_NAME
```

`TARGET` にはすでに存在しているファイル（リンクが指し示すことになるファイル）を指定します。`LINK_NAME` には作成するリンクの名前を指定します。`TARGET` も `LINK_NAME` も、カレントディレクトリではない場合には、フルパスを指定します。例えば次のようなコマンドです。

```
$ ln target.txt /home/carol/Documents/hardlink
```

このコマンドは、カレントディレクトリにある `target.txt` へのハードリンクを、`/home/carol/Documents/` ディレクトリ内に `hardlink` という名前で作成します。

第二引数（`LINK_NAME`）を省略すると、パス名を除いたターゲットと同じ名前のリンクをカレントディレクトリに作成します。

ハードリンクを管理する

ハードリンクを作成することは、TARGETと同じinodeを指す、LINK_NAMEというディレクトリエントリを作成することです。ディレクトリの実体はそのディレクトリ内に存在するファイル名とinodeの対応表であり、そのようにしてinodeと対応づけられるファイル名のことをディレクトリエントリと呼びます。作成したハードリンク（LINK_NAME）と元のファイル名（TAREGT）は、異なるディレクトリ内に存在しているとしても、同じinodeを指し示すことになります。ハードリンクからでも元のファイル名からでも、同じinodeのファイルを指すのですから、そのファイルの内容が変更されます。ハードリンクを削除しても元のファイル名を削除しても、残っているほうの名前からそのファイルを参照できます。

ファイルを「削除」するときには、データが実際にディスクから消去されるわけではなく、ディスク上のデータに対応するinodeを指すディレクトリエントリが削除されます。同

じinodeを指す別のディレクトリエントリ（ハードリンク）があれば、データにアクセスできるのです。いわば三叉路（Y字路）のようなものであり、同じ地点に向かう一つの路を塞いだとしても、もう一つの路からたどり着けるというわけです。

`ls` に `-i` オプションを指定して実行すると、このことを確かめられます。以下の例を見てみましょう。

```
$ ls -li
total 224
3806696 -r--r--r-- 2 carol carol 111702 Jun  7 10:13 hardlink
3806696 -r--r--r-- 2 carol carol 111702 Jun  7 10:13 target.txt
```

パーミッションの前に表示されている数字がinodeです。`hardlink` と `target.txt` の両方とも同じ数字（`3806696`）です。これは一方が他方のハードリンクだからです。

上記の例ではファイル名から区別できるように見えるかもしれませんが、実は、どちらが元ファイルでどちらがハードリンクかを区別できません。どちらも仕組みとしてはまったく同じものなのです。

ハードリンクを作成すると リンク数 が増えることにも注目してください。リンク数は、`ls -l` を実行するとパーミッションの直後に表示される数値のことです。ファイルを作成したときの値は `1`（ディレクトリを作成したときの値は `2`）で、ハードリンクを作成するたびに1ずつ増えます。そういうわけで、上記の例ではリンク数が `2` になっています。

シンボリックリンクとは異なり、（ディレクトリのループが発生しうるため）ディレクトリを指し示すハードリンクを作成することはできません。また、（inodeはファイルシステムごとに決まるため）ターゲットと別のファイルシステム内にハードリンクを作成することもできません。

ハードリンクを移動（削除）する

ハードリンクは通常のファイルと同じように扱われますから、`rm` コマンドで削除でき、`mv` コマンドで名前を変更したり移動させたりできます。ハードリンクは、ターゲットと同じinodeを指していますから、リンクを壊すことなく移動させられます。

シンボリックリンク

シンボリックリンクを作成する

`ln` コマンドに `-s` オプションを指定すると、シンボリックリンクを作成できます。例えば次のように実行します。

```
$ ln -s /home/carol/target.txt /home/carol/Documents/softlink
```

このコマンドは、`/home/carol/` ディレクトリにある `target.txt` ファイルを指し示す、`softlink` という名前のシンボリックリンクを `/home/carol/Documents/` ディレクトリ内に作成します。（訳注：カレントディレクトリ以外にシンボリックリンクを作成する場合、TARGETをフルパスで指定するのが原則です。理由は後で説明しています。）

ハードリンクを作成するときと同様に、第二引数を省略すると、パス名を除いたターゲットと同じ名前のリンクをカレントディレクトリに作成します。

シンボリックリンクを管理する

シンボリックリンクはファイルシステム内のパスを指し示します。ファイルだけでなくディレクトリを指し示すシンボリックリンクを作成できますし、別のファイルシステムのファイルを指し示すこともできます。`ls` コマンドの出力でシンボリックリンクはすぐにそれとわかります。

```
$ ls -lh
total 112K
-rw-r--r-- 1 carol carol 110K Jun  7 10:13 target.txt
lrwxrwxrwx 1 carol carol   12 Jun  7 10:14 softlink -> target.txt
```

上の例では、`softlink` ファイルのパーミッションの最初の文字が `l` なので、シンボリックリンクだとわかります。さらに、ターゲット（`target.txt`）を指していることが、ファイル名の後に矢印（`->`）で示されています。

`ls -l` の実行結果で、シンボリックリンク自体のパーミッションは所有者・所有グループ・その他とも常に `rwx` になりますが、実際にアクセスできるかどうかはターゲットのパーミッションによって決まります。

シンボリックリンクを移動（削除）する

ハードリンクと同様に、シンボリックリンクも `rm` コマンドで削除して、`mv` コマンドで名前を変更したり移動させたりできます。ただし、シンボリックリンクを移動させる際には、リンクが壊れてしまわないように注意する必要があります。

シンボリックリンクを作成するときにターゲットの場所をフルパスで指定しなければ、シンボリックリンクが置かれたディレクトリからの相対的なパスだと解釈されます。そのため、シンボリックリンクないしターゲットを移動させると、リンクが壊れてしまいます。

具体的な例で考えましょう。`original.txt` という名前のファイルがカレントディレクトリにあります。次のコマンドを実行して、`softlink` という名前のシンボリックリンクを作成します。

```
$ ln -s original.txt softlink
```

ここまでは何も問題ありませんね。`ls` コマンドで確認します。

```
$ ls -lh
total 112K
-r--r--r-- 1 carol carol 110K Jun  7 10:13 original.txt
lrwxrwxrwx 1 carol carol   12 Jun  7 19:23 softlink -> original.txt
```

矢印（`->`）で示されているように、`softlink` が `original.txt` を指しています。このリンクを一つ上の階層（親ディレクトリ）に移動させ、そのリンクが指す内容を `less` コマンドで表示させようとすると、以下のようなエラーになります。

```
$ mv softlink ../
$ less ../softlink
../softlink: No such file or directory
```

`original.txt` のフルパスを指定していなかったため、リンクと同じディレクトリにある `original.txt` を指すリンクを作成していたのです。`softlink` を一つ上の階層（親ディレクトリ）に移動させたので、リンクと同じディレクトリに `original.txt` ファイルはもはや存在しません。だからリンクが壊れてエラーになったのです。

移動させたときにリンクが壊れてしまわないようにするには、ターゲットをフルパスで指定します。

```
$ ln -s /home/carol/Documents/original.txt softlink
```

このようにターゲットのフルパスを指定して作成すると、シンボリックリンクをどこに移動させても壊れずにターゲットを指し続けます。（訳注：最初の例でTARGETをフルパスで指定した理由も同じです。ここでは説明のためにカレントディレクトリに作成したリンクをわざわざ移動させてリンクが壊れることを示しましたが、シンボリックリンクの指し先が それが置かれたディレクトリからの相対パス として解釈されることに注意すれば、相対パスで指定しても構いません。）

`ls` コマンドで確かめてみます。

```
$ ls -lh
total 112K
lrwxrwxrwx 1 carol carol   40 Jun  7 19:34 softlink ->
/home/carol/Documents/original.txt
```

演習

1. `/home/carol/Documents` ディレクトリ内に `document.txt` という名前のファイルがあります。このファイルへのシンボリックリンクを `text.txt` という名前でカレントディレクトリに作成してください。（訳注：カレントディレクトリがどこであるか明示されていないことに注意してください）。

2. あるファイルのハードリンクを作成することとコピーを作成することとはどう違いますか？

発展演習

1. 以下に示すコマンドを実行して、`recipes.txt` という名前のファイルを作成し、このファイルのハードリンク（`receitas.txt`）を作成して、そのハードリンクのシンボリックリンク（`rezepte.txt`）を作成します。

```
$ touch recipes.txt
$ ln recipes.txt receitas.txt
$ ln -s receitas.txt rezepte.txt
```

   このディレクトリの内容は次のようになるはずです。

```
$ ls -lhi
total 160K
5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 receitas.txt
5388833 -rw-r--r-- 4 carol carol  0K jun 17 17:25 recipes.txt
5388837 lrwxrwxrwx 1 carol carol  12 jun 17 17:25 rezepte.txt -> receitas.txt
```

   ハードリンクの `receitas.txt` は `recipes.txt` と同じinodeを指します。`receitas.txt` を削除すると、シンボリックリンクの `rezepte.txt` はどうなりますか？ 理由とともに答えてください。

2. フラッシュドライブを差し込み、`/media/youruser/FlashA` にマウントしました。そのフラッシュドライブの一番上の階層にある `esquema.pdf` というファイルを指し示す、`schematics.pdf` という名前のリンクをホームディレクトリに作ろうとして、以下のコマンドを実行しました。

```
$ ln /media/youruser/FlashA/esquema.pdf ~/schematics.pdf
```

   このコマンドを実行するとどうなるでしょうか？ 理由とともに答えてください。

3. `ls -lah` を実行すると、以下のように出力されました。

```
$ ls -lah
total 3,1M
drwxr-xr-x 2 carol carol 4,0K jun 17 17:27 .
drwxr-xr-x 5 carol carol 4,0K jun 17 17:29 ..
-rw-rw-r-- 1 carol carol 2,8M jun 17 15:45 compressed.zip
```

```
-rw-r--r-- 4 carol carol  77K jun 17 17:25 document.txt
-rw-rw-r-- 1 carol carol 216K jun 17 17:25 image.png
-rw-r--r-- 4 carol carol  77K jun 17 17:25 text.txt
```

- 上記の出力からわかるのはシンボリックリンクの数ですか、それともハードリンクの数ですか？

- 各ファイルのinodeを確認するには、`ls` コマンドにどのオプションを指定しますか？

4. `~/Documents` ディレクトリ内に、顧客名が書かれた `clients.txt` という名前のファイルと、`somedir` という名前のディレクトリがあります。`somedir` ディレクトリ内に、ファイル名は同じく `clients.txt` ですが別の顧客名が書かれた別のファイルがあります。以下のコマンドを実行すると、この状況を再現できます。

```
$ cd ~/Documents
$ echo "John, Michael, Bob" > clients.txt
$ mkdir somedir
$ echo "Bill, Luke, Karl" > somedir/clients.txt
```

以下のコマンドを実行して、`somedir` ディレクトリ内に、`clients.txt` ファイルを指し示す `partners.txt` という名前のリンクを作成します。

```
$ cd somedir/
$ ln -s clients.txt partners.txt
```

ここまでのところのディレクトリ構造は以下のようになります。

```
Documents
|-- clients.txt
 `-- somedir
    |-- clients.txt
    `-- partners.txt -> clients.txt
```

次に、以下のコマンドを実行して、`partners.txt` を、`somedir` ディレクトリから、`~/Documents` ディレクトリに移動させます。そして、`less` コマンドで `partners.txt` の内容を表示します。

```
$ cd ~/Documents/
$ mv somedir/partners.txt .
$ less partners.txt
```

このリンクは機能するでしょうか？　もし機能するなら、どのファイルを指し示しているでしょうか？　理由とともに答えてください。

5. 以下は `ls -l` の実行結果の抜粋です。

```
-rw-r--r-- 1 carol carol 19 Jun 24 11:12 clients.txt
lrwxrwxrwx 1 carol carol 11 Jun 24 11:13 partners.txt -> clients.txt
```

`partners.txt` のパーミッションは何ですか？　理由とともに答えてください。

まとめ

このレッスンでは、次のことを学びました。

- リンクとは何か。
- シンボリック リンクと ハード リンクはどう違うか。
- どのようにリンクを作成するか。
- どのようにリンクを移動（名前の変更、削除）するか。

このレッスンでは、次のコマンドについて説明しました。

- `ln` ： リンクを作成するコマンドです。オプションなしではハードリンクを作成します。`-s` オプションを指定するとシンボリックリンクを作成します。ハードリンクはターゲットと同じファイルシステム内にしか作成できません。シンボリックリンクはターゲットと別のファイルシステムにも作成できます（ネットワークストレージに作成することもできます）。
- `ls` コマンドに `-i` オプションを指定すると、inodeを表示します。

演習の解答

1. `/home/carol/Documents` ディレクトリ内に `document.txt` という名前のファイルがあります。このファイルへのシンボリックリンクを `text.txt` という名前でカレントディレクトリに作成してください。（訳注：カレントディレクトリがどこであるか明示されていないことに注意してください）。

   シンボリックリンクを作成するコマンドは `ln -s` です。シンボリックリンクがどのディレクトリにあっても機能するように、ターゲットをフルパスで指定します。実行するコマンドは次のとおりです。

   ```
   $ ln -s /home/carol/Documents/document.txt text.txt
   ```

2. あるファイルのハードリンクを作成することとコピーを作成することとはどう違いますか？

   ハードリンクはファイルの別名です。ハードリンクは元ファイルの複製のように見えますが、どちらもディスク上の同じデータを指していますから、同一の実体です。ハードリンクの内容を変更したら元ファイルの内容も変更されますし、元ファイルの内容を変更したらハードリンクの内容も変更されます。ファイルのコピーは、ディスク上の別の場所にある全く別の実体です。コピーの内容を変更しても元ファイルの内容は変更されませんし、元ファイルの内容を変更してもコピーの内容は変更されません。

発展演習の解答

1. 以下に示すコマンドを実行して、`recipes.txt` という名前のファイルを作成し、このファイルのハードリンク（`receitas.txt`）を作成して、そのハードリンクのシンボリックリンク（`rezepte.txt`）を作成します。

```
$ touch recipes.txt
$ ln recipes.txt receitas.txt
$ ln -s receitas.txt rezepte.txt
```

このディレクトリの内容は次のようになるはずです。

```
$ ls -lhi
total 160K
5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 receitas.txt
5388833 -rw-r--r-- 4 carol carol  0K jun 17 17:25 recipes.txt
5388837 lrwxrwxrwx 1 carol carol  12 jun 17 17:25 rezepte.txt -> receitas.txt
```

ハードリンクの `receitas.txt` は `recipes.txt` と同じinodeを指します。`receitas.txt` を削除すると、シンボリックリンクの `rezepte.txt` はどうなりますか？ 理由とともに答えてください。

シンボリックリンクの `rezepte.txt` は機能しなくなります。シンボリックリンクは、inodeではなく名前を指し示すからです。データは存在していて `recipes.txt` という名前でアクセスできるのですが、`rezepte.txt` が指し示す `receitas.txt` という名前はもう存在しないので機能しなくなります。

2. フラッシュドライブを差し込み、`/media/youruser/FlashA` にマウントしました。そのフラッシュドライブの一番上の階層にある `esquema.pdf` というファイルを指し示す、`schematics.pdf` という名前のリンクをホームディレクトリに作ろうとして、以下のコマンドを実行しました。

```
$ ln /media/youruser/FlashA/esquema.pdf ~/schematics.pdf
```

このコマンドを実行するとどうなるでしょうか？ 理由とともに答えてください。

コマンドの実行に失敗し、`Invalid cross-device link□□□□□□□□□□□□□□□□□□` というエラーメッセージが表示されます。このメッセージに示されているように、ハードリンクは異なるファイルシステム（パーティションやデバイス）にあるファイルを指し示すことができません。異なるファイルシステムにあるファイルを指し示すリンクを作成する場合は、`ln` コマンドに `-s` オプションを指定して シンボリック リンクを作成します。

3. `ls -lah` を実行すると、以下のように出力されました。

```
$ ls -lah
total 3,1M
drwxr-xr-x 2 carol carol 4,0K jun 17 17:27 .
drwxr-xr-x 5 carol carol 4,0K jun 17 17:29 ..
-rw-rw-r-- 1 carol carol 2,8M jun 17 15:45 compressed.zip
-rw-r--r-- 4 carol carol  77K jun 17 17:25 document.txt
-rw-rw-r-- 1 carol carol 216K jun 17 17:25 image.png
-rw-r--r-- 4 carol carol  77K jun 17 17:25 text.txt
```

   - 上記の出力からわかるのはシンボリックリンクの数ですか、それともハードリンクの数ですか？

     ハードリンクの数です（パーミッションの直後に表示されています）。シンボリックリンクを作成しても、`ls -l` の出力からわかるリンク数は増えません。

   - 各ファイルのinodeを確認するには、`ls` コマンドにどのオプションを指定しますか？

     `-i` オプションを指定します。以下のように、`ls` の出力の最初の列にinode番号が表示されます。

```
$ ls -lahi
total 3,1M
5388773 drwxr-xr-x 2 carol carol 4,0K jun 17 17:27 .
5245554 drwxr-xr-x 5 carol carol 4,0K jun 17 17:29 ..
5388840 -rw-rw-r-- 1 carol carol 2,8M jun 17 15:45 compressed.zip
5388833 -rw-r--r-- 4 carol carol  77K jun 17 17:25 document.txt
5388837 -rw-rw-r-- 1 carol carol 216K jun 17 17:25 image.png
5388833 -rw-r--r-- 4 carol carol  77K jun 17 17:25 text.txt
```

4. `~/Documents` ディレクトリ内に、顧客名が書かれた `clients.txt` という名前のファイルと、`somedir` という名前のディレクトリがあります。`somedir` ディレクトリ内に、ファイル名は同じく `clients.txt` ですが別の顧客名が書かれた別のファイルがあります。以下のコマンドを実行すると、この状況を再現できます。

```
$ cd ~/Documents
$ echo "John, Michael, Bob" > clients.txt
$ mkdir somedir
$ echo "Bill, Luke, Karl" > somedir/clients.txt
```

以下のコマンドを実行して、somedir ディレクトリ内に、clients.txt ファイルを指し示す partners.txt という名前のリンクを作成します。

```
$ cd somedir/
$ ln -s clients.txt partners.txt
```

ここまでのところのディレクトリ構造は以下のようになります。

```
Documents
|-- clients.txt
 `-- somedir
    |-- clients.txt
    `-- partners.txt -> clients.txt
```

次に、以下のコマンドを実行して、partners.txt を、somedir ディレクトリから、~/Documents ディレクトリに移動させます。そして、less コマンドで partners.txt の内容を表示します。

```
$ cd ~/Documents/
$ mv somedir/partners.txt .
$ less partners.txt
```

このリンクは機能するでしょうか？ もし機能するなら、どのファイルを指し示しているでしょうか？ 理由とともに答えてください。

これは紛らわしい状況です。リンクは機能し、~/Documents ディレクトリ内の clients.txt ファイルを指し示します。John、Michael、Bob という名前が書かれているほうのファイルです。

シンボリックリンク partners.txt を作成する際に、ターゲットである clients.txt のフルパスを指定しなかったので、リンクがあるディレクトリからの相対的なパスだと解釈されます。本問では、ターゲットが、partners.txt リンクと同じディレクトリにあると解釈されます。

そのリンクが、~/Documents/somedir ディレクトリから ~/Documents ディレクトリに移動され、ターゲットがリンクと同じディレクトリに存在しなくなるので、リンクが壊れて機能しなくなるはずです。しかし、偶然にも ~/Documents ディレクトリ内に clients.txt という名前のファイルが存在するため、リンクはこのファイルを指し示すようになります。somedir ディレクトリ内の clients.txt ファイルではありません。

こうしたややこしい状況を避けるために、シンボリックリンクを作成する際には、ターゲットのフルパスを指定するようにするとよいです。

5. 以下は `ls -l` の実行結果の抜粋です。

```
-rw-r--r-- 1 carol carol 19 Jun 24 11:12 clients.txt
lrwxrwxrwx 1 carol carol 11 Jun 24 11:13 partners.txt -> clients.txt
```

`partners.txt` のパーミッションは何ですか？　理由とともに答えてください。

`partners.txt` のパーミッションは `rw-r--r--` です。リンクのパーミッションはターゲットのパーミッションと常に同じです。

104.7 システムファイルを検索し、ファイルを正しい場所に配置する

LPI目標への参照

LPIC-1 version 5.0, Exam 101, Objective 104.7

総重量

2

主な知識分野

- FHSでのファイルの正しい場所を理解する。
- Linuxシステムでのファイルとコマンドの検索。
- FHSで定義されている、重要なファイルとディレクトリの場所と目的を理解する。

用語とユーティリティ

- `find`
- `locate`
- `updatedb`
- `whereis`
- `which`
- `type`
- `/etc/updatedb.conf`

104.7 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	104 デバイス、Linuxファイルシステム、ファイルシステム階層標準
Objective:	104.7 システムファイルを検索し、ファイルを正しい場所に配置する
Lesson:	1 of 1

はじめに

Linuxディストリビューションの見た目やサイズは様々ですが、ほとんどのディストリビューションが ファイルシステム階層標準（Filesystem Hierarchy Standard、FHS） と呼ばれるファイルシステムの “標準レイアウト” を採用しています。ファイルシステム階層標準とは、ファイルシステムの標準的なディレクトリ構成を定めたもので、そのおかげで相互運用やシステム管理がやりやすくなっています。このレッスンでは、ファイルシステム階層標準を説明するとともに、Linuxシステムでファイルを検索する方法を示します。

ファイルシステム階層標準

ファイルシステム階層標準（FHS）は、Linuxシステムにおけるディレクトリの構成と内容を標準化するために、Linux Foundationによって始められた試みです。この標準に従うことが必須要件になっているわけではありませんが、ほとんどのディストリビューションはこれに従っています。

NOTE ファイルシステム階層標準の詳細を知りたければ、http://refspecs.linuxfoundation.org/fhs.shtml で提供されているFHS 3.0 specificationを読んでください。

ファイルシステム階層標準が定める基本的なディレクトリは以下のとおりです。

/
: 階層の頂点をなすルートディレクトリです。他のすべてのディレクトリはこのルートディレクトリの下に置かれます。ファイルシステムはよく逆さまの木にたとえられますが、ルートディレクトリはその根っこに相当します。枝に相当する他のディレクトリは、すべてこの根っこから枝分かれしたものになります。

/bin
: システム起動時に必要となる最も重要なコマンドを格納します。

/boot
: 初期RAMディスク（initrd）やLinuxカーネルなど、カーネルの起動に必要なファイルを格納します。

/dev
: デバイスファイルを格納します。システムに接続された物理デバイス（例えば1番目のSCSIないしSATAディスクである `/dev/sda`）だけでなく、カーネルが提供する仮想デバイスも格納します。

/etc
: ホスト固有の設定ファイルを格納します。プログラムによっては、`/etc` 以下に複数の設定ファイルが入ったサブディレクトリを作成することがあります。

/home
: 各ユーザーの個人用ファイルを置くホームディレクトリをこの中に作成します。通常、各ユーザーのホームディレクトリの名前は、ユーザー名と同じにします。ユーザーJohnのホームディレクトリは、`/home/john` にするということです。rootユーザーは例外で、`/root` ディレクトリがホームディレクトリになります。サービスを実行するシステムユーザーには、`/var/lib` の下に独自のホームディレクトリを持つものや、専用のホームディレクトリを持たないものもあります。

/lib
: システム起動時に必要となる最も重要なコマンド（`/bin` ないし `/sbin` に置かれる）を実行するために必要な共有ライブラリを格納します。64ビットCPUのシステムでは、32ビット用のライブラリと64ビット用のライブラリが分けて置かれることがあります。

/media
: フラッシュドライブ、CDドライブやDVDドライブ、フロッピーディスク、メモリーカードなど、ユーザーがマウントするリムーバブルメディアを、このディレクトリ以下に自動的にマウントします。

/mnt
ファイルシステムを一時的にマウントする際に、このディレクトリをマウントポイントにします。

/opt
システムに標準ではないオプションのソフトウェアを格納します。

/root
rootユーザーのホームディレクトリです。

/run
アプリケーションが実行時に使用するデータ（主にはPIDを記録したファイル）を格納します。

/sbin
システム起動時ないしメンテナンス時にシステム管理者が使用するコマンドの実行可能ファイルを格納します。

/srv
システムサービスが利用するデータを格納します。例えば、ウェブサーバーが提供するページを `/srv/www` に格納します。

/tmp
一時ファイルを格納します。

/usr
システム起動時やメンテナンス時に必要とされない、大部分のプログラムや読み取り専用のデータを格納します。大部分のプログラムやデータは、このディレクトリの下に置かれます。

/proc
実行中のプロセス情報を保持する仮想ファイルを格納します。

/var
印刷キュー、ログ、メールボックス、一時ファイル、ブラウザのキャッシュなど、システムが動いている間は頻繁に書き換えられる可変データを格納します。

`/etc`、`/usr`、`/var` など、いくつかのディレクトリには、上で紹介したディレクトリをサブディレクトリに持つものがあります（`/usr/bin` や `/var/tmp` のようなサブディレクトリが存在します）。

一時ファイル

一時ファイルとは、プログラムが一時的に作業用のデータを保持するファイルのことです。一時ファイルの例としては、実行中のプロセスに関するデータ、クラッシュログ、自動保存されたファイル、ファイルを変換するときの中間ファイル、キャッシュファイルなどがあります。

一時ファイルの保存場所

ファイルシステム階層標準（FHS）では、一時ファイルの保存場所を定めています。用途に応じて複数の保存場所があり、それぞれ挙動が異なります。よって、一時ファイルを保存する際には、開発者がFHSに従って保存場所を決めることが推奨されます。

`/tmp`
: このディレクトリに保存されたファイルは、次にプログラムを呼び出すときまで保存されている保証がありません。システムの起動時にこのディレクトリ内のファイルをすべて消すことが推奨されています。

`/var/tmp`
: このディレクトリに保存されたファイルは、起動時に 消すべきではない とされているので、再起動しても残っていることが多いでしょう。

`/run`
: プロセスID（`.pid`）など、実行中のプロセスが使用する、実行時に変化するデータ（ランタイムファイル）を保存します。複数のランタイムファイルを必要とするプログラムは、このディレクトリにサブディレクトリを作る場合があります。システムの起動時にはこのディレクトリ内のファイルを 消去します。かつては `/var/run` ディレクトリがこの用途に充てられていたので、`/var/run` ディレクトリが `/run` ディレクトリへのシンボリックリンクになっているシステムもあります。

上記とは異なる場所に一時ファイルを作成することもできますが、FHSに従うことが望ましいです。

ファイルの検索

`find` コマンドを使うと、Linuxシステムでファイルを検索できます。このコマンドはとても有用なツールです。オプションが様々あり、動作や出力を必要に応じて調整できます。

`find` コマンドは、検索開始位置と条件式を引数に指定します。例えば、カレントディレクトリとそのサブディレクトリの中から、ファイル名が `.jpg` で終わるファイルを検索するなら、以下のコマンドを実行します（`.` が検索開始位置、`-name '*.jpg'` が条件式です）。

```
$ find . -name '*.jpg'
```

```
./pixel_3a_seethrough_1.jpg
./Mate3.jpg
./Expert.jpg
./Pentaro.jpg
./Mate1.jpg
./Mate2.jpg
./Sala.jpg
./Hotbit.jpg
```

このコマンドは、ファイル名の最後の4文字が `.jpg` であるすべてのファイル名を出力します。`*` はすべての文字を表すワイルドカードなので、`.jpg` の前にどの文字があってもマッチします。`.jpg` の後にも `*` を置くとどうなるかを見てましょう。（訳注： `-name` に指定するパターンは、正規表現ではなくシェルと同様のグロブパターンです。）

```
$ find . -name '*.jpg*'
./pixel_3a_seethrough_1.jpg
./Pentaro.jpg.zip
./Mate3.jpg
./Expert.jpg
./Pentaro.jpg
./Mate1.jpg
./Mate2.jpg
./Sala.jpg
./Hotbit.jpg
```

上記の出力で太字にしている `Pentaro.jpg.zip` ファイルは、先の例では出力されていませんでした。ファイル名に `.jpg` を含んでいるものの、その後に余計な文字があるので、先の例ではマッチしなかったのです。上記のコマンドは、`.jpg` の前後が何でもよいという意味なので、このファイルもマッチします。

TIP `-name` オプションは大文字と小文字を区別します。大文字と小文字を区別せずにマッチさせたい場合は、`-iname` オプションを指定します。

シェルがパターンを解釈してしまわないように、`*.jpg` を `'`（シングルクォート）ないし `"`（ダブルクォート）で囲みます。グロブパターンを引数に使用する場合には、意図しないマッチを防ぐために、クォートで囲む習慣をつけておいたほうがよいです。

デフォルトでは、`find` コマンドは、検索開始位置から再帰的にサブディレクトリをたどっていきます。`-maxdepth N` オプションを指定すると、サブディレクトリをたどる階層を `N` に限定します（訳注：ディレクトリ階層の深い部分まで探りたくない場合などに使用します）。逆に `N` 階層目から検索を始める `-mindepth N` オプションもあります。どちらも、検索開始位置を1としてカウントします。

`-mount` オプションを指定すると、検索開始位置とは異なるファイルシステムにあるディレクトリを検索しなくなります。`-fstype` オプションを指定すると、指定したファイルシステムタイプだけを検索するようになります。`find /mnt -fstype exfat -iname "*report*"` コマンドは、`/mnt` 以下にマウントされたexFATファイルシステムだけを検索します。

ファイルの属性で検索する

以下に示すオプションを指定すると、ファイルの属性を条件として検索できます。例えば、現在のユーザーが書き込み可能なファイル、指定したパーミッションのファイル、指定したサイズのファイルなどを検索できます。

`-user USERNAME`
: ユーザー `USERNAME` が所有者であるファイルにマッチします。

`-group GROUPNAME`
: グループ `GROUPNAME` が所有グループであるファイルにマッチします。

`-readable`
: 現在のユーザーが読み取り可能なファイルにマッチします。

`-writable`
: 現在のユーザーが書き込み可能なファイルにマッチします。

`-executable`
: 現在のユーザーが実行可能なファイルにマッチします。ディレクトリに関しては、現在のユーザーが中に入れる（`x` パーミッションがある）ディレクトリにマッチします。

`-perm NNNN`
: `NNNN` パーミッションのファイルにマッチします。例えば、`-perm 0664` は、所有者と所有グループは読み書き可能でその他は読み取り可能なファイル（パーミッションが `rw-rw-r--` のファイル）にマッチします。

`NNNN` の直前に `-` をつけると、少なくとも `NNNN` のパーミッションのファイルにマッチします。例えば、`-perm -644` は、少なくとも `644`（`rw-r--r--`）のパーミッションのファイルにマッチします。`664`（`rw-rw-r--`）や `775`（`rwxrwxr-x`）のパーミッションのファイルにもマッチします。

`-empty`
: 空のファイルやディレクトリにマッチします。

`-size N`
: サイズが `N` のファイルにマッチします。`N` は、デフォルトでは、512バイトのブロックの数です。`N` に他の単位をつけることもできます。`Nc` はバイト、`Nk` はキビバイ

ト（KiB、1KiBは1024バイト）、NM
はメビバイト（MiB、1MiBは1024×1024バイト）、NG はギビバイト（GiB、1GiBは1024×1024×1024バイト）です。

+ ないし - を N の直前につけると、そのサイズより 大きい ファイルまたは 小さい ファイルを検索できます。例えば、-size -10M は、サイズが10MiBよりも小さいファイルにマッチします。

例えば、ホームディレクトリ以下から、ファイル名のどこかに report というパターンを含み（大文字と小文字を区別しない）、0644 のパーミッションで、10日以上前にアクセスされ、サイズが1MiBより大きいファイルを検索するなら、次のコマンドを実行します。

```
$ find ~ -iname "*report*" -perm 0644 -atime 10 -size +1M
```

時間で検索する

タイムスタンプ属性（アクセス日時・属性変更日時・更新日時）を利用してファイルを検索することもできます。そのために使用するオプションは次のとおりです。

-amin N、 **-cmin N**、 **-mmin N**
: それぞれ、アクセス日時、属性変更日時、更新日時が、N 分前のファイルにマッチします。

-atime N、 **-ctime N**、 **-mtime N**
: それぞれ、アクセス日時、属性変更日時、更新日時が、N×24 時間前（N 日前）のファイルにマッチします。

アクセス日時（atime）は、そのファイルにアクセスされた時に更新されます。属性変更日時（ctime）は、ファイルの属性、すなわちinodeの内容が変更された時に更新されます。パーミッションや所有者が変更された時や、他のタイムスタンプが更新された時にも更新されますから、「何かの操作」を行った時に更新されるものと考えても構いません。更新日時（mtime）は、ファイルの内容が書き換えられた時に更新されます。

例えば、カレントディレクトリ以下から、更新日時が24時間（1日）以内で、サイズが100MiBより大きいファイルを検索するなら、次のコマンドを実行します。

```
$ find . -mtime -1 -size +100M
```

訳注：-exec オプションを使用するとマッチしたファイルに対して任意のコマンドを実行できるなど、find コマンドは多才で強力なツールであり、特にファイルシステムを再帰的にたどって処理する場面には欠かせません。manページなどを参照して、さまざまな機能を概観しておくことをお勧めします。

`locate` と `updatedb` を使用する

`locate` を使うと、Linuxシステムで指定したパターンにマッチするファイルを高速で見つけ出せます。`find` のようにファイルシステムを検索するのではなく、`locate` は `updatedb` によって生成されるデータベースを検索します。そのため、高速で結果を得られますが、データベースが更新されていない場合には結果が不正確になることがあります。

`locate` の後に検索パターン（グロブパターン）を指定して実行すると、そのパターンにマッチするファイルを出力します。例えば、システムに存在するすべてのJPEG画像を検索するなら、`locate jpg` を実行します（訳注：このコマンドはあくまでもファイル名に `jpg` を含むファイルを検索するのであって、`.jpg` ではなく `.jpeg` や `.JPG` がファイル名になっているJPEG画像を検索することはできません）。このコマンドを実行すると、以下のような結果を出力します。

```
$ locate jpg
/home/carol/Downloads/Expert.jpg
/home/carol/Downloads/Hotbit.jpg
/home/carol/Downloads/Mate1.jpg
/home/carol/Downloads/Mate2.jpg
/home/carol/Downloads/Mate3.jpg
/home/carol/Downloads/Pentaro.jpg
/home/carol/Downloads/Sala.jpg
/home/carol/Downloads/pixel_3a_seethrough_1.jpg
/home/carol/Downloads/jpg_specs.doc
```

`jpg` というパターンを指定すると、`locate` はファイル名のどこかに `jpg` を含むファイルをすべて出力します。上の例にある `jpg_specs.doc` のように、`jpg` という文字列をファイル名に含んでいるけれどもサフィックスが `jpg` ではないファイルも出力されます（訳注：`locate .jpg` を実行すると、`.jpg` という文字列をファイル名に含むファイルを検索するので、サフィックスが `jpg` であるファイルの検索に近づきます）。

TIP `locate` コマンドは、ファイル名全体からパターン探すのであって、サフィックスから探すわけではありません。

デフォルトでは大文字と小文字を区別します。`locate jpg` は、パターンが小文字で指定されているので、`JPG` をファイル名に含むファイルを出力しません。大文字と小文字を区別しないようにするには、`-i` オプションを指定します。先ほどの例を改良して実行します。

```
$ locate -i .jpg
/home/carol/Downloads/Expert.jpg
/home/carol/Downloads/Hotbit.jpg
/home/carol/Downloads/Mate1.jpg
/home/carol/Downloads/Mate1_old.JPG
/home/carol/Downloads/Mate2.jpg
```

```
/home/carol/Downloads/Mate3.jpg
/home/carol/Downloads/Pentaro.jpg
/home/carol/Downloads/Sala.jpg
/home/carol/Downloads/pixel_3a_seethrough_1.jpg
```

上で太字にした `Mate1_old.JPG` ファイルが、最初の例では出力されていなかったのですけれども、今回は出力されています。

`locate` の後に半角スペースで区切って複数のパターンを指定できます。以下の例は、大文字と小文字を区別せず、`zip` または `jpg` というパターンにマッチするファイルを検索します。

```
$ locate -i zip jpg
/home/carol/Downloads/Expert.jpg
/home/carol/Downloads/Hotbit.jpg
/home/carol/Downloads/Mate1.jpg
/home/carol/Downloads/Mate1_old.JPG
/home/carol/Downloads/Mate2.jpg
/home/carol/Downloads/Mate3.jpg
/home/carol/Downloads/OPENMSXPIHAT.zip
/home/carol/Downloads/Pentaro.jpg
/home/carol/Downloads/Sala.jpg
/home/carol/Downloads/gbs-control-master.zip
/home/carol/Downloads/lineage-16.0-20190711-MOD-quark.zip
/home/carol/Downloads/pixel_3a_seethrough_1.jpg
/home/carol/Downloads/jpg_specs.doc
```

複数のパターンを指定する際に `-A` オプションを付けると、すべてのパターンにマッチする結果を出力するようにできます。次のコマンドは、`.jpg` と `.zip` の両方にマッチするファイルを出力します。

```
$ locate -A .jpg .zip
/home/carol/Downloads/Pentaro.jpg.zip
```

`-c` オプションを指定すると、マッチしたファイルのフルパスを出力するのではなく、マッチしたファイルの数をカウントして出力します。例えば、以下のコマンドを実行すると、システムに存在する `.jpg` にマッチするファイルの数を出力します。

```
$ locate -c .jpg
1174
```

`locate` コマンドには、`updatedb` によって生成されるデータベース（`/var/lib/ mlocate.db`）に存在するエントリだけを出力するという特徴があります。データベースが古ければ、すでに削除されたファイルを出力します。`-e` オプションを指定すると、ファイルが現存するかを確認してから出力するようになるので、こうした事態を避けられます。

`-e` オプションを指定しても、データベースが更新された後に作成されたファイルは出力されません。`updatedb` コマンドを実行してファイルを調べ直し、データベースを更新することで解決できます。ただし、`updatedb` の実行にはroot権限が必要ですし、ディスクに大量のファイルが存在する場合には `updatedb` コマンドの実行に時間がかかることがあります。

`updatedb` の挙動を設定する

`updatedb` の挙動を `/etc/updatedb.conf` ファイルで設定できます。このファイルは各行が一つの変数であるテキストファイルです。空白行は無視され、`#` で始まる行はコメントとして扱われます。

`PRUNEFS=`
: このパラメータの記載したファイルシステムタイプは、`updatedb` コマンドの対象になりません。大文字と小文字を区別しないので、`NFS` と `nfs` は同じになります。

`PRUNENAMES=`
: このパラメータの後に記載した名前のディレクトリは、`updatedb` コマンドの対象になりません。

`PRUNEPATHS=`
: このパラメータの後に記載したパスは、`updatedb` コマンドの対象になりません。例えば `/var/pool /media` などを記載します。

pruneは「（余計な枝などを）刈り取る」という意味であり、`updatedb` の対象としないものを記載します。複数記載する場合は半角スペースで区切ります。システムがデフォルトで作成している `/etc/updatedb.conf` の内容を見ると要領がわかります。

実行可能ファイル、マニュアルページ、ソースファイルを検索する

`which` コマンドは、実行可能ファイルのフルパスを示します。例えば、`bash` の実行可能ファイルの場所を知りたければ、次のコマンドを実行します。

```
$ which bash
/usr/bin/bash
```

`-a` オプションを付けると、（PATH指定の順で）すべての実行可能ファイルのフルパスを表示します。先頭に表示されるものが、コマンド名だけを入力したときに実行されるもので

す。例を見てみましょう。

```
$ which mkfs.ext3
/usr/sbin/mkfs.ext3

$ which -a mkfs.ext3
/usr/sbin/mkfs.ext3
/sbin/mkfs.ext3
```

TIP どのディレクトリが（which コマンドの検索対象となる）PATH に設定されているかは、echo $PATH コマンドを実行するとわかります。これは、変数 PATH の内容（$PATH）を表示する（echo）コマンドです。

type コマンドは、実行可能ファイルのフルパスと種類を表示します。type に続けて調べたいコマンドの名前を入力します。

```
$ type locate
locate is /usr/bin/locate
```

which と同様に、-a オプションを付けると、すべての実行可能ファイルのフルパスを表示します。例えば次のような出力になります。

```
$ type -a locate
locate is /usr/bin/locate
locate is /bin/locate
```

-t オプションを付けると、コマンドの種類を表示します。コマンドの種類には alias、keyword、function、builtin、file があります。以下にいくつか例を示します。

```
$ type -t locate
file

$ type -t ll
alias

$ type -t type
type is a built-in shell command
```

whereis コマンドは少し視点が異なり、実行可能ファイルだけではなく、マニュアルペー

ジやソースファイル（存在する場合）も検索します。`whereis` の後に調べたいコマンドの名前を入力します。

```
$ whereis locate
locate: /usr/bin/locate /usr/share/man/man1/locate.1.gz
```

上の例では、コマンド本体の実行可能ファイル（`/usr/bin/locate`）と圧縮されたマニュアルページ（`/usr/share/man/man1/locate.1.gz`）が結果として出力されています。

オプションを指定すれば表示する結果を絞り込めます。`-b` オプションを指定すれば実行可能ファイルだけを、`-m` オプションを指定するとマニュアルページだけを、`-s` オプションを指定するとソースファイルだけを表示します。同じ例を続けると、次のようになります。

```
$ whereis -b locate
locate: /usr/bin/locate

$ whereis -m locate
locate: /usr/share/man/man1/locate.1.gz
```

演習

1. プログラムの実行に一時ファイルが必要です。プログラムの実行終了後に使うことはない一時ファイルです。その一時ファイルを作成するのに適しているディレクトリはどれですか？

2. 起動時に必ず消去される一時ファイルを格納するディレクトリはどれですか？

3. `find` コマンドで、カレントディレクトリ以下から、現在のユーザーが書き込み可能で、過去10日以内に更新され、4GiBより大きいサイズのファイルを検索してください。

4. `locate` コマンドで、ファイル名に、`report` というパターンを含み、かつ `updated`、`update`、`updating` のいずれかのパターンを含むファイルを検索してください。（訳注：目的のファイルがすべて結果に出力されるなら多少余計な出力が混じってもよいので、厳密に考えないようにしましょう。）

5. `ifconfig` のマニュアルページの場所を検索してください。

6. `ntfs` ファイルシステムを `updatedb` の対象としないようにするには、`/etc/updatedb.conf` にどう記載しますか？

7. システム管理者として内蔵ディスク（`/dev/sdc1`）を一時的にマウントするときに、FHSに従うなら、どのディレクトリにマウントしますか？

発展演習

1. `locate` コマンドは、`updatedb` によって生成されたデータベースを参照して結果を出力します。そのデータベースが古いせいで、`locate` コマンドが、もう存在しないファイルを結果として出力してしまうことがあります。存在しないファイルを出力しないようにするには、`locate` コマンドにどのオプションを指定しますか？

2. カレントディレクトリとその2階層下のサブディレクトリまでを対象として、カレントディレクトリとは異なるファイルシステムは対象とせず、ファイル名に `Status` または `statute` が含まれるファイルを検索してください。（訳注：目的のファイルがすべて結果に出力されるなら多少余計な出力が混じってもよいので、厳密に考えないようにしましょう。）

3. `/mnt` 以下から、`ext4` ファイルシステムを対象として、少なくとも所有者が読み取りと実行が可能、かつ、所有グループが読み取り可能で、直近2時間に属性が変更されたファイルを検索してください。

4. カレントディレクトリから2階層以上深いサブディレクトリを対象として、更新日時が30日より前の空のファイルを検索してください。

5. ユーザー `carol` と `john` はグループ `mkt` のメンバーです。`john` のホームディレクトリ以下から、`carol` が読み取れるファイルを探してください。

まとめ

このレッスンでは、LinuxシステムにおけるFHSに準拠したファイルシステムの構造と、名前や属性からコマンドやファイルを検索する方法を説明しました。以下のコマンドを取り上げました。

`find`
: 様々な条件からファイルやディレクトリを検索するコマンドです。

`locate`
: データベースを使用して、マシンに保存されているファイルの位置を見つけるコマンドです。

`updatedb`
: `locate` コマンドが使用するデータベースを更新します。

`which`
: 実行可能ファイルのフルパスを表示します。

`whereis`
: 実行可能ファイルとマニュアルページ、ソースファイルの場所を表示します。

`type`
: 実行可能ファイルの場所とコマンドの種類（`□□□□`、`□□□□` など）を表示します。

演習の解答

1. プログラムの実行に一時ファイルが必要です。プログラムの実行終了後に使うことはない一時ファイルです。その一時ファイルを作成するのに適しているディレクトリはどれですか？

   プログラムの実行終了後に使わない一時ファイルなので、`/tmp` ディレクトリが適しています。

2. 起動時に必ず消去される一時ファイルを格納するディレクトリはどれですか？

   `/run` です。`/var/run` が存在するシステムなら `/var/run` もです。

3. `find` コマンドで、カレントディレクトリ以下から、現在のユーザーが書き込み可能で、過去10日以内に更新され、4GiBより大きいサイズのファイルを検索してください。

   `-writable`、`-mtime`、`-size` オプションを指定して、以下のコマンドを実行します。

   ```
   find . -writable -mtime -10 -size +4G
   ```

4. `locate` コマンドで、ファイル名に、`report` というパターンを含み、かつ `updated`、`update`、`updating` のいずれかのパターンを含むファイルを検索してください。（訳注：目的のファイルがすべて結果に出力されるなら多少余計な出力が混じってもよいので、厳密に考えないようにしましょう。）

   `locate` でパターンを「かつ」でつなげるので、`-A` オプションを指定します。

   ```
   locate -A "report" "updat"
   ```

5. `ifconfig` のマニュアルページの場所を検索してください。

   `whereis` コマンドで `-m` オプションを指定します。

   ```
   whereis -m ifconfig
   ```

6. `ntfs` ファイルシステムを `updatedb` の対象としないようにするには、`/etc/updatedb.conf` にどう記載しますか？

   `PRUNEFS=` の後に `updatedb` の対象としないファイルシステムタイプを書くので、`PRUNEFS=ntfs` と記載します。

7. システム管理者として内蔵ディスク（`/dev/sdc1`）を一時的にマウントするときに、FHSに従うなら、どのディレクトリにマウントしますか？

   技術的にはディスクをどこにマウントすることもできますが、FHSに従うなら `/mnt` にマウントします。

発展演習の解答

1. `locate` コマンドは、`updatedb` によって生成されたデータベースを参照して結果を出力します。そのデータベースが古いせいで、`locate` コマンドが、もう存在しないファイルを結果として出力してしまうことがあります。存在しないファイルを出力しないようにするには、`locate` コマンドにどのオプションを指定しますか？

   `locate -e PATTERN` のように `-e` オプションを指定して実行します。

2. カレントディレクトリとその2階層下のサブディレクトリまでを対象として、カレントディレクトリとは異なるファイルシステムは対象とせず、ファイル名に `Status` または `statute` が含まれるファイルを検索してください。（訳注：目的のファイルがすべて結果に出力されるなら多少余計な出力が混じってもよいので、厳密に考えないようにしましょう。）

   `-maxdepth` はカレントディレクトリも含めて階層を数えるので、1+2=3を指定します。次のコマンドを実行します。

   ```
   find . -maxdepth 3  -mount -iname "*statu*"
   ```

3. `/mnt` 以下から、`ext4` ファイルシステムを対象として、少なくとも所有者が読み取りと実行が可能、かつ、所有グループが読み取り可能で、直近2時間に属性が変更されたファイルを検索してください。

   `-fstype` オプションに続けて検索対象とするファイルシステムタイプを指定します。パーミッションを3桁の数値で考えると、所有者が読み取りと実行が可能だということは最初の桁が5以上で、所有グループが読み取り可能だということは2桁目が4以上で、その他のパーミッションは気にしなくてもよいということは最後の桁が0以上です（まとめると、パーミッションが540以上だということです）。属性変更日時がN分以内のファイルを検索する場合には `-cmin N` を指定します。よって、実行するコマンドは次のとおりです。

   ```
   find /mnt -fstype ext4 -perm -540 -cmin -120
   ```

4. カレントディレクトリから2階層以上深いサブディレクトリを対象として、更新日時が30日より前の空のファイルを検索してください。

   `-mindepth N` オプションを指定すると、カレントディレクトリからN-1階層以上深いサブディレクトリを対象として検索します（カレントディレクトリを1として階層を考えることに注意してください）。空のファイルを検索するには `-empty` オプションを指定します。更新日時を日単位で指定するのは `-mtime N` オプションです。よって、実行するコマンドは次のとおりです。

```
find . -empty -mtime +30 -mindepth 3
```

5. ユーザー `carol` と `john` はグループ `mkt` のメンバーです。`john` のホームディレクトリ以下から、`carol` が読み取れるファイルを探してください。

   グループ `mkt` で共有するファイルは、所有グループを `mkt` にするのが一般的です。所有グループが `mkt` で、グループパーミッションが `r`（4）以上のファイルを検索します。よって、実行するコマンドは次のとおりです。

   ```
   find /home/john -group mkt -perm -040
   ```

覚え書き

PDF生成日: 2024-01-20

この覚え書きおよびプロジェクト全体に関する質問やコメントは、learning@lpi.org 宛てに電子メールでお送り下さい。