# Linuxシステムについて

sudo lsof /dev/i2c-1 デバイスを使用しているプロセスのリストを表示

## 主なディストリビューション

|  |  |  |
| --- | --- | --- |
| Red　Hat系 | Red Hat  (RHEL) | Red Hat社が開発・販売。サーポートやアップデートなどの契約を結ぶ事で利用できる商用ディストリビューション |
| Fedora | Red Hat社が支援するコミュニティによって開発  Fedora 22からパッケージ管理に dnf を使う。 |
| CentOS | RHEL から有償部分を除外したもの。サーバー用途で良く使われていた。 |
| AlmaLinux | RHELとバイナリ互換のオペレーティングシステムを提供 |
| Scientific Linux | 科学・物理学系のパッケージが含まれる。  フェルミ研究所が開発。RHELをベース。 |
| Debian系 | Debian | GNUと言うフリーソフトウェアを積極的に採用  Ubuntuに比べるとterminal操作中心。 |
| Ubuntu | Linux初心者にも利用しやすい。LTS(Long Term Support) 有り。人気。  デスクトップ用途が多いがサーバー版も存在する。 |
| Linux Mint | Ubuntuがベース。デスクトップ用途。  洗練され，かつ誰にでも簡単に使いやすいシステムを目指している。 |
|  | openSUSE | SUSE社が支援するコミュニティによって開発。SUSE Linuxから改名した。  パッケージ管理に zypper を使う。ローリングリリース。 |
|  | Arch Linux | 簡素さ，エレガントさ，などを開発理念。  やや上級者向き。ローリングリリース。 |

### CentOSのISOイメージの違い

DVD 大体の場合はこれを選べばよい

Everything 全７パッケージを含む。イメージは16GB以上ある

LiveGNOME GNOMEデスクトップ用のLive（お試し）イメージ

LiveKDE KDEデスクトップ用のLive（お試し）イメージ

Minimal 最小限

NetInstall 最もサイズが小さい。LAN上にCentOSのローカルミラーがある場合。

UnixとLinuxの違い

Unix

まだOSの概念が無かった頃の最も古いOSの１つ。途中からライセンス形式になった。

CUIのイメージが強いが，X Window Systemを用いたGUIも存在する。

なお，近年のMacOSはベースにUnixを用いている。

Linux

Linus Torvaldsによって作られたオープンソースなOS。Unixに似せてはあるが，オリジナル。

### FreeBSD

オープンソースであるBSDの流れをくむUNIX系OSの１つ。

LinuxOSではない。UNIX系のOSの中ではLinuxに次いで人気が高い。

### 確認方法

#### Linuxカーネル

uname -srm

#### Linuxディストリビューション

cat /etc/os-release # 共通（概要）

cat /etc/redhat-release # RedHat系

cat /etc/os-release # Debian系

cat /etc/debian\_version # Debian系2（バージョンのみ）

cat /etc/lsb-release # Ubuntu

## カーネル情報

5.4 リリース日2019年11月26日

・ロックダウン機能

・exFATのサポート

## 有償版

### 有償Linux

|  |  |  |
| --- | --- | --- |
| 運営会社 | 製品名 | 備考 |
| Canonical | Ubuntu Advantage |  |
|  | MIRACLE LINUX |  |
| Oracle | Oracle Linux |  |
| RedHat | RHEL | もっとも有名な有償版 |

### 有償サポート

|  |  |  |
| --- | --- | --- |
| 提供会社 | 対象ディストリビューション | 備考 |
| Hewlett-Packard（HP） | Debian |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## ミニマルシステム

### 概要

Linux以外のOS（特にWindows）で起動する最小限Linuxシステム。

### 比較

|  |  |  |
| --- | --- | --- |
| 名称 | 役割 | 説明 |
| MinGW | コンパイラ | Windows 用の GNU ツールチェーン。とにかく Windows 用の gcc や GNU ld。"Windows 用" とは、Windows 上で動作したり、Windows 用のアプリが作れたりすること。  要するに Visual C++ や Clang/LLVM みたいなもの。 |
| mingw-w64 | コンパイラ | MinGW のフォーク。Win64 向けの改良が加わっており、2021 年現在の主流。 |
| Cygwin | POSIX 環境 | Windows 用の POSIX 環境。cygwin1.dll とリンクすると POSIX 環境をエミュレーションできる。POSIX 環境は Cygwin の目標。 |
| MSYS | POSIX 環境 | MinGW の補助用の POSIX 環境。Cygwin のフォーク。POSIX 環境は互換性のためのもの。 あくまでも configure スクリプトなどの実行用。 |
| MSYS2 | POSIX 環境 | 開発補助用の POSIX 環境。Cygwin のフォーク。mingw-w64 の主な入手手段。POSIX 環境は互換性のためのもの。 |
| MSYS2–  MSYS2 | POSIX 環境 | MSYS2 のための開発環境であり、POSIX エミュレーションが有効。 |
| MSYS2–  MinGW32 | Win32 開発環境 | MSYS2 と mingw-w64 を使用した Win32 ネイティブ アプリの開発環境。(POSIX と互換性なし) ≒Win用gcc |
| MSYS2 - MinGW64 | Win64 開発環境 | MSYS2 と mingw-w64 を使用した Win64 ネイティブ アプリの開発環境。(POSIX と互換性なし) ≒Win用gcc |
| WSL | POSIX 環境 | Microsoft が開発した Windows (NT カーネル) 上で動作する Linux 互換レイヤー。一部のシステムコールが非常に遅い。 |
| WSL 2 | Linux | NT カーネル上ではなく Hyper-V 仮想化環境で動作する Linux カーネルを使用した WSL。 |

（参考）<https://gist.github.com/rz7d/4e699498d339a5837f016b3108631a2b>

### wsl2

#### 概要

WSL2 Windows Subsystem for Linux2

Windows上でLinuxを動作させられる仕組み。Microsoftが公式に対応。（関連）MSYS2

#### 初期設定

設定→アプリと機能→プログラムと機能→Windowsの機能の無効化/有効化 以下を有効化

Linux用Windowsサブシステム

Windowsハイパーバイザー プラットフォーム

仮想マシンプラットフォーム

#### チュートリアル

##### インストール

wsl –set-default-version 2 # WSL2をデフォルトとする

wsl --list --online # ダウンロード可能なイメージのリスト

wsl --install -d Debian # ディストリビューションの作成 →スタートメニューに Debian Linux が現れる

wsl --install -d Ubuntu #

wsl --list # インストール済のディストリビューションの表示

wsl --unregister Debian # ディストリビューションの削除

注意：インストール後に先にカーネル更新プログラムをダウンロードしてインストール

[https://docs.microsoft.com/ja-jp/windows/wsl/install-manual#step-4---download-the-linux-kernel-update-package](https://docs.microsoft.com/ja-jp/windows/wsl/install-manual" \l "step-4---download-the-linux-kernel-update-package)

オプション

-l インストール済のディストリビューションの一覧を表示する。--list

-v バージョンと状態を表示する --verbose

--running 現在実行中

-d destro ディストリビューションを指定。--distribution

-s destro 既定として使用。

--set-version destro ver バージョンを指定する。

##### ファイルへのアクセス

ファイルエキスプローラーで \\wsl$\Ubuntu　と入力

詳細

ルートディレクトリ

C:\Users\username\AppData\Local\Packages\The**Debian**Project.DebianGNULinux\_76v4gfsz19hv4\LocalState\rootfs

C:\Users\username\AppData\Local\Packages\CanonicalGroupLimited.**Ubuntu**onWindows\_79rhkp1fndgsc\LocalState\rootfs

ィレクトリ。その中のrootディレクトリがホームディレクトリで，最初は何も入っていない。

#### トラブルシューティング

##### failed to download

DNSの名前解決ができていない事が原因。下のファイルがDNSの参照先情報

/etc/wsl.conf

nameserver 172.30.208.1 # WSL2の仮想DNSサーバ

このファイルを以下のように書き換える

nameserver 8.8.8.8

補足：Ubuntuならnanoが使える。viはどのディストリビューションでも使えるはず。

#### 外部デバイスの認識

PowerShell（cmdではできない）

GET-WMIOBJECT -query "SELECT \* from Win32\_DiskDrive"

# 表示された「DeviceID」をコピー

> DeviceID : \\.\PHYSICALDRIVE1 # 例

wsl --mount \\.\PHYSICALDRIVE1 # マウント(insiderプログラムに参加する必要がある)2021-09-30

一般の人にinsiderプログラムに参加させるのはどうかと思うので中止 2021-09-30

#### 情報

（公式情報）<https://docs.microsoft.com/ja-jp/windows/wsl/install>

### msys

# 情報

## 基礎編

### OS操作基礎知識

Linuxのユーザーには３種類ある

・管理ユーザー（root）

windowsで言うadminみたいなもの。各種設定ができるスーパーユーザー。全てのファイル・ディレクトリへアクセスでき，また，システムの動作を変更したり，システムを停止することができる。

rootユーザーには、UID(User ID)、GID(Group ID)ともに0が割り当てられる。

・一般ユーザー

システムを利用するためのユーザー。システムに関するファイルやディレクトリに対しては，参照のみかアクセスができないよう制御。

・システムユーザー（システムアカウント）

例えばメールやデータベースなどの特定のアプリケーションを動作させるためのユーザー。基本的にはユーザーが直接動かすことはなく，Linuxへのログインユーザーとしては使用できないように設定されている。

（補足）ユーザーの確認方法

cat /etc/passwd # パスワード設定がされているかで確認する方法（最も一般的な方法）

Linuxのパスワードには数字や記号を入れる必要がある。

コマンドにおける .（ドット）はカレントディレクトリからの**相対パス**となる。

./dir1/file1.txt # カレントディレクトリ以下にあるdir1ディレクトリ内のfile1.txtにアクセス。

/（バックスラッシュ）はroot ディレクトリからの**絶対パス**となる。

頭に「.」がついたファイルは隠しファイルとして扱われる。

ファイルアプリではCtrl+h ，または右上のメニューから「隠しファイルを表示する」

### レスキューモード

HDDのシステムに不具合があった場合などにブートローダーを再設定する機能の事。

HDDからブートする際に，ESCを何度か押すUbuntuの場合 Advanced options for Ubuntu を選ぶ。

DVDやブータブルUSBなどの外部メディアでブートし（要BIOS設定），同じくESC？

・initramfs

初期RAMディスクと呼ばれる。ブートローダからinitへと処理を移す際，必要ファイルを保持するRAM。

カーネルの肥大化を避ける為，必要なドライバのみを選択してはメモリ上に展開してルートファイルシステムをマウントする。参照元のファイルシステムは /boot/initrd.img で，これはミニルートと呼ばれる。

このファイルはlsinitrd コマンドで中身を参照できる。

### 起動の流れ

BIOS（UEFI）

・記憶装置などに関して最低限の認識を行う。

・記憶装置からMBRを読み込み，ブートローダへ制御を移す。

⇩

ブートローダ

LinuxではGRUBが使われる

・記憶装置内のカーネルをロード。

・初期RAMディスク（initramfs）の内容をメモリ上に展開。

⇩

カーネル

・高度にハードウェアを認識・制御。

・初期RAMディスクを使って，ルートファイルシステムのマウント。

・特別なプロセス「init」を起動する。

⇩

init（プロセス）

・PIDが必ず１になる最初のプロセス。

・時代などによってソフトは異なる。近年ではsystemdが使われる。

・/etc/inittabの記述に基づいて自動起動させるプロセスを立ち上げる。

### カーネル

「中核」という訳語が使われる。各種ハードウェアとユーザーが使うソフトウェアの間に立ち，様々な指令を与える。複数のカーネルをまとめる事でOSは作られる。

BIOS，ブートローダの次に起動され，高度にハードウェアを認識・制御し，ルートファイルシステムのマウントなど様々な初期化処理を行う。

なお，カーネルのあと呼び出されるプロセスは時代順に，SysVinit，Upstart，Systemd がある。

（参考）[システム](#_システム)

カーネルパラメータ

カーネルの挙動を操作するパラメータ

カーネルにオプションを通すタイミングは3回ある

1.カーネルをビルドするとき。

2.カーネルを起動するとき（普通，ブートローダから読み込まれた時）。

3.起動中（/proc と /sys にあるファイルを使う）。

### デバイスドライバ

デバイスドライバは，カーネルに予め含まれているか，カーネルモジュールとして提供されている。

カーネルモジュールとはカーネルの一機能（プログラムファイル）であり，必要なタイミングでカーネルへ追加（ロード）や削除（アンロード）を行うことができる。

/dev ディレクトリに作成されるデバイスファイルとは別のもの。

### ランレベル

SysVinit（古い従来型のinit手法）で定められていた動作モードの事

|  |  |  |  |
| --- | --- | --- | --- |
| ラン  レベル | 説明 | 対応するsystemdのユニット | 備考 |
| 0 | システム停止 | poweroff.target |  |
| 1 | シングルユーザーモード  ※「レスキューモード」と認識した方が良い | rescue.target  または  runlevel1.target | **復旧用**でroot のみがログインできる。  覚え方1 = single  **single.targetは存在しない**点に注意 |
| 2 | マルチユーザーモード(テキスト，NFS※なし) | multi-user.target | ※NFSファイル共有 |
| 3 | マルチユーザーモード(テキスト) |  |
| 4 | 未使用 |  |
| 5 | マルチユーザーモード(GUI) | graphical.target | 覚え方：5とgは似ている |
| 6 | システム再起動 | reboot.target | 覚え方：ロク(6)でもないから再起動 |
| ない | 緊急モード | emergency.target | CentOS7から追加。rescue.targetでもログインできない場合などに使用する |

ディストリビューションによってランレベルの定義が異なる。

なお，systemd を使って動作モードを変える時には systemctl isolate を使う。

systemdを利用するシステムではSysVinitとの関係が分かりやすいように，runlevel0.target〜runlevel6.targetまでのシンボリックリンクも用意されている。

systemd.unit=rescue

参考）[システム系 - init/telinit](#_システム系)，[systemctl（コマンド）](#_systemctl（コマンド）_1),[GRUB2](#_GRUB2)

### ショートカットキー

#### 共通

Ctrl + C 割込みによるプロセスの終了

Ctrl + D ランレベルを通常モードに戻す

Ctrl + Z ターミナルからプロセスを一時停止

#### GUI

Ctrl + Alt + T ターミナルを起動

Ctrl + Alt + D デスクトップの表示

Ctrl + Alt + Del システムの再起動？？　SysVinitでは/etc/inittab　が参照される

Ctrl + Alt + F2 コンソールモードにする

Ctrl + Alt + F7 デスクトップにする

Ctrl + Shift + N 新規フォルダ作成

#### CUI

Ctrl + Shift + C ターミナル上でのコピー(or Ctrl +Insert)　 GUIのファイル操作の場合はCtrl+C

Ctrl + Shift + V ターミナル上でのペースト

#### テキスト編集

Ctrl+V，Ctrl+M 続けて押下する事で，制御コード「^M」（CR；キャリッジリターン）を入力

## 中級編

### 概要

・Linuxはマルチユーザー・マルチタスクのOS

ターミナル ：入力（キーボードやマウス），出力（ディスプレイ）環境

コンソール ：Linuxサーバーを直接操作する端末（console：制御卓）

・Linuxシステムを構築する為に，最低限，ルートパーティションとswap領域が必要。

ルートパーティション ：ルートディレクトリ / を持つ**パーティション**の事。

swap領域 ：仮想メモリとして使われる**パーティション**の事

### 初期RAMディスク

/bootディレクトリに格納。

ファイルシステムへアクセスするのに必要なドライバやスクリプトが含まれる

カーネルのバージョンごとに内容が異なる

### ブートローダ(grub)

GRand Unified Bootloader；高機能ブートローダの事

|  |  |  |
| --- | --- | --- |
|  | GRUB Legacy(GRUB1) | grub2 |
| バージョン | 0.9x系 | 1.9以降 |
| 設定ファイル | boot/grub/menu.lst および  boot/grub/grub.**conf** | boot/grub/grub.**cfg**  ※これを直接編集する事はない |
| 編集方法 | 設定ファイルを直編集で，そのままkernelコマンドとして実行される。 | /etc/default/grubファイルおよび  /etc/grub.dディレクトリ内のファイルを間接編集＋コマンドgrub-mkconfig |
| ディスク番号 | 0から数える | 0から数える |
| パーティション番号 | 0から数える | 1から数える |
|  |  |  |

### マスターブートレコード

PCのブートセクタの一種。コンピュータの起動直後に実行される、ブートレコーダの情報を持つ。

全体のサイズは512バイト。

|  |  |  |  |
| --- | --- | --- | --- |
| アドレス | サイズ | 内容 |  |
| 0x0(0) | 446byte | ブートストラップローダ  （別名：ブートローダ） |  |
| 0x01BE | 64byte | パーティションテーブル | 第1～第4パーティション。各16byte |
| 0x01FF | 2byte | ブートシグニチャ |  |

※MBRにインストールされるGRUBのブートストラップローダのサイズは440バイト。

（参考）[GRUB起動プロンプト](#_GRUB起動プロンプト)， [Grub Legacy](#_Grub_Legacy)

### initシステム

時代別のinitシステムの違いをを簡単な表でまとめる。

これらのシステムはBIOS → ブートローダ → カーネルの次に起動される。

|  |  |  |  |
| --- | --- | --- | --- |
|  | SysVinit | Upstart | Systemd |
| 特徴 |  |  |  |
| 起動方式 | スクリプト  /etc/rc.d/ が本体で，リンクが多数存在  /sbin/init を起動し  /etc/inittabの記述に基づいて初期化を行う | イベント駆動型 | ユニット  /etc/systemd/  system/default.target  を実行 |
| 初期化プロセス | init |  |  |
| サービス起動 | 事前に決められた順序 | 並列起動が可能 | 並列起動が可能 |
| サービスのコマンド |  | initctl | systemctl |
| 処理単位 |  | Job（ジョブ） | Unit（ユニット） |
| プロセスの管理 |  | PID | cgroups |
| 備考 | init q で設定を反映 | SysVinitと互換性有り |  |

#### SysVinit

古い従来型のinit手法。以下のような特徴がある。

・設定ファイル/etc/inittabの記述に基づいて，アプリケーションレベルの初期化を行う。

・起動用シェルスクリプトの処理を順次実行する。

・各サービスは事前に決められた順序で起動処理が行われる。

・各ランレベルに応じたスクリプトは /etc/rc[0-6].d に入っている。

K01smartd

一文字目がSの場合はサービスを開始（Start），Kの場合はサービスを停止(Kill)

数字は実行優先順位。サービス名は任意の名前を付ける。

e.g. /etc/rc.1.d/K01bluetooth

#### Upstart

Ubuntuなどで採用されていたが，現在ではsystemdへ置き換えられている。少し古い仕組み。

主な特徴として，以下の３つがある。

・サービスの並列起動が可能

・イベント駆動型

・/etc/inittab を用いない

・サービスやタスクを「Job」という単位で扱う

ジョブとはデーモンやタスク（一度だけ実行させたい処理）をまとめた単位。

イベントとはジョブの実行に必要な前提条件。SysVinitが順次起動だったのと対照的。イベントはinitctlコマンドで発行する事もできる。例：initctl start mysqld

#### Systemd

・処理を行うサービスなどを「Unit」という単位で扱う

・システム起動時，最初に実行されるUnitは /etc/systemd/system/default.target

・.target というUnitで複数のUnitをグループ化できる

・cgroups(Control Group) でプロセスをグループ化して管理する

※cgroupはカーネルのコンテナ機能

・サービスの並列起動によって高速なシステム起動が行える

・設定ファイル /etc/systemd/system

### X Window System

概要

X Window Systemは、ユーザー入力（マウスやキーボード）や出力を受け付ける「Xサーバー」と、Xサーバーの入力情報を受け取って処理し、Xサーバーに対して画面の表示要求を行ったりする「Xクライアント」で成り立っている。

GNOMEやKDEに利用されている。

ユーザー識別子

|  |  |  |  |
| --- | --- | --- | --- |
|  | 意味 | idコマンド | 備考 |
| effective UID  （実効UID） | アクセス権限上のユーザー | デフォルト | ファイル作成とファイルアクセスに影響 |
| real UID  （実UID） | プロセスやファイルの実際の所有者 | -r | シグナル送信の権限に影響 |
| saved UID  （保存UID） | 特権を一時的に高める際に使用 |  |  |
| file system UID | file systemに対するアクセス権を得たいが、実UIDなどを変更したくない場合に使用 |  | 変更していない限り実効UIDと同じ |

#### 関連コマンド

### LXD

システムコンテナマネージャ。ゲストOSはLinuxのみだが，完全/準仮想化同様にゲストOSを起動できる。

### Devicetree Overlay

別名：DTO

#### 概要

OS(CPU)からアクセスできるハードウェア郡をデバイスドライバ目線で記述したデータ構造。

Open Firmware というグループ（Linux開発のグループとは別の方々）により策定されたが、Linuxでも同様のフォーマットで運用されている。

複雑化したSoC（System-on-a-chip）に合わせて複雑化したDevicetreeを簡単にカスタマイズできる仕組みで、Linuxカーネルをリコンパイルする作業を削減できる。

詳細

|  |  |
| --- | --- |
| 項目 | 概要 |
| device tree source (.dts) | makeコマンドでのコンパイル対象のファイル |
| device tree blob (.dtb) | .dtsをコンパイルしてバイナリ化したもの |

補足：RaspberryPi OSの場合、/boot/ ディレクトリ内に存在していた

基本的には「ノード」と呼ばれるグループに「プロパティ」または「子ノード」を属させる

/dts-v1/; /\* 最初の行は必ずこれ \*/

/ { /\* 最上位ノードは必ず「/（ルート）」 \*/

node1abel {

a-string-property = "A string";

child-node2 {

child-property = <1>;

};

};

};

#### チュートリアル

インストール

sudo apt install device-tree-compiler

コンパイル

[ terminal ]

dtc -I dts -O dtb -o sample.dtbo sample.dts #

ファイルの移動

root 権限が必要なので、まずは特定ユーザーのhomeディレクトリに移動させてから下記コマンドを実行すると良い。

sudo mv sample.dtbo /boot/overlays/

#### 詳細

##### dtsファイルに何を書くべきか

Kernel Sourceのドキュメントを読むのが一番

LinuxカーネルソースのDocumentation / devicetreeにあります。 =>どこ？

ラズパイの場合はこのようにするとカーネルソースがダウンロードできるらしい

# 現在のカーネルバージョンを確認。

uname -r

以下で

<https://github.com/Hexxeh/rpi-firmware/commits/master>

FIRMWARE\_REV=`cat /boot/.firmware\_revision`

KERNEL\_REV=`curl -L https://github.com/Hexxeh/rpi-firmware/raw/${FIRMWARE\_REV}/git\_hash`

curl -L https://github.com/raspberrypi/linux/archive/${KERNEL\_REV}.tar.gz > linux-kernel-src.tar.gz

ラズパイ

3B：<https://github.com/raspberrypi/linux/blob/rpi-4.4.y/arch/arm/boot/dts/bcm2710-rpi-3-b.dts>

##### dtsファイル詳細

必要最小限度のオーバーレイファイル

[ minimum.dtso ]

/dts-v1/;

/plugin/;

/ {

fragment@0 { /\* node名は何でも良い \*/

target = <phandle>; /\* overlayするnodeの phandle \*/

　　/\*　もしくは \*/

　　target-path="/path"; /\* overlayするnodeの path \*/

\_\_overlay\_\_ {

/\* オーバーレイする property 定義 たち\*/

/\* オーバーレイする node 定義 たち\*/

};

};

};

@（アドレス）

書式：node名 @ unit-address

unit-address の部分はreg propertyで指定されるアドレスを16進で記述。

reg propertyを持たないnodeではunit-addressは必要なし。

&（ラベル）

予め設定してあるラベル（label:）がある場合、そのノード（のphandle）を指定可能。

"-@" オプション付きでコンパイルされていれば、別ファイルからも参照可能。

<> cell-list（後述）の事。≒配列ととらえる

# #address-cells, #size-cells（後述）でのみ使われる

その他

コメントは　C style (/\* ... \*/) および C++ style (// ...) が使える

同じ階層で同じ node名 に対する定義が出てきた場合は、同じnodeとみなし、そのnodeに対する追加定義となる。

参考元：https://qiita.com/eofz/items/20cfa99601756fc98905

##### 標準Property群

compatible

デバイスドライバとの結び付けに使われる最重要Property。

デバイスドライバを必要とするデバイスには、このpropertyが必要

このvalueで指定されている文字列に一致するcompatible-stringを持つデバイスドライバの初期化ルーチンが呼び出されデバイス登録が行われる。

Text String のリストで "manufacturer,model" という形式。

例:　compatible = "altr,socfpga-stmmac", "snps,dwmac-3.70a", "snps,dwmac";

status

そのデバイスを有効にするかしないか。

Value は "okay" or "disabled“

#address-cells, #size-cells

child node を持つ node で定義。どの様に child node がアドレッシングされるかを定義。

#address-cells ベースアドレス。通常は=1。

#size-cells メモリサイズ。通常は=1、 memory mapped でない場合 0

\_\_overlay\_\_ 対象ノードについかされたbodyを含む（？？？）

情報元：https://developer.ridgerun.com/wiki/index.php/NVIDIA\_Jetson\_-\_Device\_Tree\_Overlay

##### dtsファイルで用いるvalueの種類

Text String

二重引用符で囲む compatible ="arm,cortex-a9";

Array/Cell-list

山括弧<>で区切られた32ビットの符号なし整数の集合

reg = <0xffd04000 0x1000>;

Binary Data

角括弧[]で区切る

mac-address =[12 34 56 ab cd ef];

複数データ

コンマを使用して連結可能

reg =<0xff900000 0x100000>, <0xffb80000 0x10000>;

複合データ（mixed-property）

="a string", [0x01 0x23 0x45 0x67], <0x12345678>;

##### dtc コマンド

インストール

sudo apt install device-tree-compiler

書式：

例：

dtc -I dts -O dtb -o sample.dtbo sample.dts #

オプション

-o <arg> 出力ファイル名。指定しないと標準出力に出力される為、ほぼ必須。 --out

-I <arg> 入力ファイル形式。dts, dtb, fs --in-format

-O <arg> 出力ファイル形式。dts,dtb,yaml,asm(アセンブラソース)

-@ シンボルの生成を有効化？？？ --symbols

トラブルシューティング

・:14.26-27 syntax error 14行目の26-27列に文法エラー

##### overlay.txt

# 拡張子なしで =dtbo\_a, dtbo\_b と

overlay=cma512

##### その他

/proc/device-tree/を見ることで動作を見ることができる？

mコマンド

ここに例があるが…

https://coral.googlesource.com/docs/+/refs/heads/master/GettingStarted.md

#### 資料

##### 用語

phandle

node を示す id 番号をCell-listとして設定するが、通常.dtsでは省略されている。

##### 参考にしたページ

公式1：<https://elinux.org/Device_Tree_Usage>

公式２（英語）：https://www.kernel.org/doc/html/latest/devicetree/overlay-notes.html

参考：<https://qiita.com/keiya-nobuta/items/10ab4eab88276b5b3ebd>

藁をもつかむ情報源：<https://forum.armbian.com/topic/18250-compile-device-tree-overlay/>

<https://www.digi.com/resources/documentation/digidocs/embedded/dey/3.2/cc8x/bsp_r_create-dt-overlays>

### SELinux

Security-Enhanced-Linux

Linuxカーネルの制御機能の１つ（ディストリビューションではない）

細かいアクセス制御が可能で、root権限ですら制限をかけることも可能

permissive SELinuxが適用されていない状態。

enforcing SELinux が適用され，アクセス制御が有効になっている状態

設定ファイル

一般的 /etc/selinux/config

Ubuntu /etc/selinux/semanage.conf

関連コマンド

getenforce #

setenforce 0 # 一時的にpermissiveモードへ変更

setenforce 1 # 一時的にenforcingモードへ変更

sestatus

### デバイスクラス

概要

USBにおいてデバイスの種類を表す分類の俗称。以下の様なものがある

ACM Communication Device モデム，ターミナルアダプタなど。

（補足）ACM；Abstract Control Model

USB Audio Class 文字通りだが，スピーカー，マイク，楽器など

CDC：Communication Device Class モデムなど

HID；Human Interface Device Class キーボード，マウス，ジョイスティックなど

MSC；Mass Storage Class ハードディスク，USBメモリやCD-ROMなど

[https://ja.wikibooks.org/wiki/Linux](https://ja.wikibooks.org/wiki/Linuxハードウェア" \l "各種USBデバイス)ハードウェア#各種USBデバイス

## サービス，プロセス

### サービスとプロセスの違い

|  |  |  |
| --- | --- | --- |
|  | サービス | プロセス |
| 概要 | 複数のプロセスの集まりで，≒デーモン。ただしデーモンでないサービスも存在する。 | あるサービスの特定の機能を担うアプリケーション。  単体のプロセスでデーモンを形成する事もある。 |
| 具体例 | FTPサービス | bash |
| 主な関連コマンド | systemctl | ps |
| 管理 |  | OS |
|  |  |  |

用語：

ゾンビプロセス

役目を終えているのにメモリを開放しない状態のプロセス。killなどする必要がある。

### systemd

#### 概要

・Unitと言う単位で処理を管理する。

・１つのサービスは１つ～複数のプロセスで構成される。

（補足）

Upstartでは処理をjobという単位で扱っていた。

#### 設定ファイル

/etc/systemd/system.conf

#### Unit

systemdで処理を管理する為に使われる１単位で，複数のプロセスで構成される。

Unitファイルの格納先は以下の2カ所。

|  |  |
| --- | --- |
| ディレクトリ | 設定内容 |
| /usr/lib/systemd/system | インストール時の初期設定。当ディレクトリ内のUnitは編集しない。 |
| /etc/systemd/system | ユーザによる個別設定デフォルトの設定を変更する場合は，当ディレクトリにUnitファイルをコピーして編集する（Systemdは当ディレクトリのUnitを優先する） |

##### Unitのタイプ

Unitは役割毎に拡張子が異なる。拡張子がUnitの種類を示す。

|  |  |  |
| --- | --- | --- |
| 拡張子 | 機能 | 補足 |
| .device | 各種デバイスを管理する | udevデーモンによって自動生成。  udevから通知されたデバイスを表す。 |
| .mount | ファイルシステムのマウントを管理する | /etc/fstab の内容からmountユニットが自動生成される。  手動でマウントを行うと，マウントポイント名のmountユニットが生成される |
| .service | サービスを制御する | 指定のバイナリを実行する  主にプロセス（≒デーモン）の起動／停止に関連 |
| .swap | swap領域を管理する | swap領域とはメモリ不足の時に利用するストレージ（HDDなど）上の仮想的なメモリの事 |
| .target | 複数のサービスを１つのグループにする | 複数のUnitをまとめるために使用する  例）multi-user.target |
| .scope | サービスに似ているが， foreign プロセスを開始せずに管理する。 |  |
| .socket | ソケットの監視 | systemdがSocketをListenして、接続があるとプロセスに受け渡す  （xinetdの代替的な機能） |
|  |  |  |
| .path |  | 指定のファイルが作成されると、指定されたサービスを起動する |
| .unit | systemd.unit　のみ  これは，起動時のターゲットを決めるカーネルパラメータでUnitではない | grub2の設定ファイルで以下の様に書き込む。  尚、カーネルパラメータでは .targetが省略可能。  systemd.unit=rescue  （補足）原則，Unit というUnitは存在しない。 |

関連）[systemctl](#_systemctl_各サービスの稼働状況や起動設定を管理_2), [systemd](#_systemd_1),[自作サービスの作り方](#_自作サービスの作り方)

##### 具体的なユニット名（抜粋）

.target

|  |  |
| --- | --- |
| ユニット名 |  |
| runlevel1.target | 文字通り。level1～6まで |
| multi-user.target | （参考）[ランレベル](#_ランレベル) |
| network-online.target | 文字通り |
|  |  |

.service

|  |  |
| --- | --- |
| ユニット名 |  |
| httpd.service | Apache のデーモン |
|  |  |
|  |  |

以下の複数のデーモンプロセスが連携して動作する

systemd-journald

各サービスが出力するログを扱う。

systemd-logind

システムへのログイン，起動停止，電源ボタン操作など

systemd-udevd

従来のudevに代わってデバイス（ホットプラグ）の動的変更，検出を行う。

systemd-analyze

起動パフォーマンスの確認

コマンド

service network restart # systemctl のようなコマンドがうてる

service --status-all # サービスの一覧を表示

service poweroff.target start # shutdown とほぼ同等

#### systemctl（コマンド）

systemdを制御するコマンド

書式：systemctl [options] コマンド [Unit\_name]

※Unit名に拡張子がない場合，.serviceの拡張子を持つUnitが指定されたものとみなす。

サブコマンド

get-default 次回起動時のターゲットを表示

set-default 次回起動時のターゲットを設定する

systemctl set-default rescue.target # デフォルト設定をシングルユーザー（レスキュー）に

list-unit-files インストールされたすべてのUnit定義ファイルを一覧表示する

list-units メモリに読み込まれたユニットの一覧を表示

システムコマンド

rescue レスキューモード（ランレベル１）へ入る

default default として定められているモードへ入る Ctrl+Dと同等

poweroff システムの電源を切る

reboot [ARG] システムを再起動する shutdownサブコマンドは存在しない点に注意！

halt シャットダウンしてシステムを停止

hibernate スリープモードに入る

次ページへ続く

～systemctl コマンド　続き

ユニットコマンド

start *service\_name* サービスを起動する (opp) stop

restart *service\_name* サービスを**再起動**

service *service\_name* restart # systemctl restart と同等の例

systemctl restart reboot.target # システムの再起動

kill ？？？

mask disableの強化版

sudo systemctl mask sleep.target suspend.target hibernate.target hybrid-sleep.target

unmask maskの解除

ユニットファイルコマンド

enable [*service\_name* | path…] # サービスの自動起動を有効 (opp) disable

systemctl disable httpd

list-unit-files すべてのUnit定義ファイルを一覧表示

sudo systemctl list-unit-files --type=service | grep *service\_name* # 特定のサービスを表示する例

status サービスの状態を表示する (compare) is-activeユニットファイルコマンド

systemctl status *service\_name* # 標準の使い方

/etc/init.d/*service\_name* status # ようするにサービスのスクリプトパスを指定して直接実行

service *service\_name* status # これでも確認できる（古いやりかた）

isolate *name* 動作モードの変更

reload *service\_name* サービスの**設定ファイルを再度読み込み**

is-active サービスが稼働しているかを表示する。(compare) statusユニットファイルコマンド

daemon-reload

備考：CentOS6系ではservice コマンドを用いていた。サブコマンドなどはほぼ同じ。

サービスによってはserviceコマンドでしか制御できないものもある。

但し，systemctl [Unit\_name] サブコマンド　と逆になるので注意！

service *httpd* status

具体例

systemctl postfix reload # 設定ファイルの再読み込み

systemctl start reboot.target # サービスの開始。但しこれは再起動となる

systemctl runlevel6.target # 上と全く同じ

sudo systemctl edit getty@tty1.service

#### journalctl

system-journaldで記録したジャーナル（システムログ）を表示する。

このログは全部表示するとものすごい長さになる。

書式：journalctl [options] [検索文字列]

-f 最新のログを更新して表示し続ける。「更新され続けるtail」のイメージ

-k システム起動時のメッセージのを表示 = dmesg --dmesg

20.09.04おそらく**K**ernel log message

-u *unit\_name* 特定のUnitのログのみを出力する

journalctl -u my\_service

-x

-e

※よくjournalctl -xeで使用される

SYSLOG\_IDENTIFIER=hoge

.serviceファイルで指定したSyslogIdentifierでログの絞り込みができる。

#### sytemd関連コマンド（その他）

systemd-logind システムへのログイン，起動停止，電源ボタン操作など

systemd-udevd ホットプラグの管理

設定ファイル：/etc/udev/rule.d

systemd-analyze 起動パフォーマンスの確認

書式：sytemd-analyze [options] コマンド

サブコマンド

time システムの起動にかかった時間を表示する

blame 初期化にかかった時間の長さ順リストアップする

critical-chain 時間のかかっている呼び出しチェーンの表示

関連）[systemctl](#_systemctl), [Unit](#_Unit_1)

### cron

クロン／クーロン

#### 概要

テキストファイルに実行時間と何を実行するかを書き込む方式。

まず使ってみるにはターミナルで下のコマンドを打つ

terminal

crontab -e # crontab の設定を開始する。→ 設定ファイルが開かれる。

※初めてcrontabを使う場合はテキストエディタ選択の対話がある。

ログファイル /var/log/cron

cronからのメッセージ（メール） /var/spool/mail/root

#### 関連コマンドなど

select-editor デフォルトのエディターの確認

現在のデフォルトエディタの情報が格納されるらしい？

echo $EDITOR

#### 設定の仕方

分 時 日 月 曜日 実行するファイルやbashコマンド

具体例

|  |  |
| --- | --- |
| 書式 | 意味 |
| 33 14 \* \* \* | 毎日14:33に実行。（アスタリスクは「すべての条件に一致」の意） |
| 0 04 \* \* 1 | （月）の午前4:00に実行（毎週）　曜日は0（日）～6（土） |
| 0 21 \* \* 1-6 | （月）〜（土）まで21:00に実行。（ハイフンで範囲指定） |
| 0,10,20,30,40,50 \* \* \* \* | 10分おきに実行（毎時）（コンマはorの意） |
| \*/10 \* \* \* \* | 10分おきに実行（毎時） |
| \*/10+1 \* \* \* \* | 10分おきに実行（毎時）かつ1分のoffset |
| \* 1 \* \* \* | 1:00から1:59まで1分おきに実行（毎日） |
| 0 \*/1 \* \* \* | 0分に1時間おきに実行（毎時） |
| 0 \* \* \* \* | 0分に1時間おきに実行（\*/1 と同じ） |
| 2 8-20/3 \* \* \* | 8:02,11:02,14:02,17:02,20:02に実行（8:00から20:00まで3時間おきに実行） |
| 30 5 1,15 \* \* | 1日と15日の5:30に実行。 |
| @reboot | cron デーモンが起動した時に実行される |

設定ファイルにも説明がある。domはDay of Month　dowはDay of Week（曜日）の意。

実行するファイルのパスはフルパスで記述する。

また，コンマ（,）やハイフン（-）を使うと複数の値を指定できる。例）3,5,10-12 3分，5分，10～12分

アスタリスクとバックスラッシュ (\*/)を使うと間隔で指定できる 例）/5\*\*\*\*\* ５分毎

@reboot Run once, at startup.

@yearly Run once a year, "0 0 1 1 \*".

@annually (same as @yearly)

@monthly Run once a month, "0 0 1 \* \*".

@weekly Run once a week, "0 0 \* \* 0".

@daily Run once a day, "0 0 \* \* \*".

@midnight (same as @daily)

@hourly Run once an hour, "0 \* \* \* \*".

具体例

@reboot /bin/sh /home/user1/shell.sh

# 曜日以降は自由に空白を使用して良い。つまり、引数を用いたコマンドも可能。

0 \*/1 \* \* \* node ~/example.js

# python で仮想環境を使用している場合は仮想環境内のpythonを使う

\*/5 \* \* \* ~/my\_proj/venv/bin/python3 ~/my\_proj/some\_script.py

編集がおわったら再起動

sudo systemctl restart cron

補足

・.serviceファイルと.timerファイルを併用してcronと同じような処理は可能。

・スクリプトを実行する場合、ファイルに実行権限が必要。cronを設定したユーザーのもので良いらしい。

・サーバー的に使いたいときでも一般ユーザーのcronで良い。

### プロセス

#### 基本

すべてのプロセスに.targetが付く？

#### システム系

一覧 - システム系

|  |  |  |
| --- | --- | --- |
| プロセス名 |  |  |
| init |  | 最初に起動されるプロセス |
| default | .target |  |
| acpid |  | 電顕管理 |
| dphys-swapfile |  | swap領域（仮想メモリ範囲）を管理する |
|  |  |  |
| kmod |  | カーネルモジュールをロード，挿入，削除 |
| udev |  | デバイス管理。ホットプラグデバイスが接続された際、動的にデバイスファイルを作成 |
| xserver-xorg |  | X Window System　つまりCUI版には無い？ 20.10.08 |

init 最初に起動されるプロセスで，PIDは必ず1。カーネルの次に起動される。

udev Linuxカーネル用のデバイス管理ツール

デーモン ：udevd

ライブラリ ：libudev コマンド ：udevadm

設定ファイル ：/etc/udev/rules.dまたは/usr/lib/udev/rules.d（ディレクトリ）に配置

環境変数 ：SYSTEMD\_LOG\_LEVEL

カーネルはデバイスを検知すると「/sys」以下（sysfsと呼ばれる）の情報を更新し，その更新をudevは検知し，/dev以下にデバイスファイルを動的に作成。デバイスファイル作成時の動作は/etc/udev/rules.d ディレクトリに配置された設定ファイル（拡張子「.rules」）に記述された情報に基づいて行われる。

sudo udevadm control --reload-rules

acpid コンピューターの電顕管理などに関連

Advanced Configuration and Power Interface

#### 通信系

|  |  |  |
| --- | --- | --- |
| プロセス名 |  |  |
| avahi-daemon |  | mDNS（ローカルDSN；≒Bonjour）を提供。  ※Androidは非対応2022-02-15現在 |
| bluetooth |  | 文字通り |
| dhcpcd |  | DHCPの提供。IPアドレスの固定。 |
| nfs-common |  | nfs（ネットワークを用いたファイル共有）。 |
| networking |  |  |
| network | .target |  |
| ssh |  |  |

#### 管理系

|  |  |  |
| --- | --- | --- |
|  |  |  |
| alsa-utils |  | オーディオ。サウンドカードのデバイスドライバ。(ALSA) |
| cron |  | 定時間経過毎にタスクを実行(crontab) ユーザー権限で実行 |
| dbus |  | アプリケーション間でメッセージ送信 |
| fake-hwclock |  | システムの終了時に現在時刻を記憶する。仮想RTC。 |
| lightdm |  | ディスプレイマネージャ（デスクトップモードのみかも？ 19.11.05）　/etc/lightdm.conf |
| paxctld |  | PaX（メモリの最小権限アクセス）のコントロールデーモン。 |
| plymouth |  | splash screen（起動時に表示される小さな画面）の制御 |
| procps |  | プロセス監視を行う。 |
| rpcbind |  |  |
| rsync |  | 高速なファイル同期を行う。 |
| rsyslog |  | ローカルおよびリモートサーバーのログ管理。 |
| triggerhappy |  | ホットキー（ショートカットキー）の管理 |

（参考）[Unit](#_Unit_1)

#### その他

一覧 – その他

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
| rng-tools | 真乱数と疑似乱数の間のブリッジとして動く |
| x11-common | X Window System（ウィンドウシステム）を管理 |

### 自作サービスの作り方

#### 概要

自動起動プログラム（サービス）を作成したい場合は /etc/systemd/system に .service ファイルを作成する。（UNIT定義ファイルと呼ぶ）iniファイルのような形式。

例えばmy\_serviced.service というファイルを作成すると，「my\_serviced」という名前のサービスが作成され，以下のように実行できる。

sudo systemctl daemon-reload # .service ファイルを書き換えたら実行する必要がある

sudo systemctl start my\_serviced # 今，開始する

sudo systemctl enable my\_serviced # OS起動時にサービスが自動起動するようにする

sudo systemctl list-unit-files --type=service | grep my\_service # サービスが認識されたかどうか

（参考）[systemd](#_systemd_2)

journalctl -b | less でログは確認する -b boot後のログ

#### 具体例

.serviceファイル

[Unit] # サービスの情報（メタデータ）

Description = Some Description # 　サービスの説明。空白文字も使える。

[Service] # メインの設定セクション

Type=simple # 　通常起動,，通常終了（デフォルト）

ExecStart= /opt/do\_something.sh # 　サービス起動時のコマンド

[Install] # サービスの有効化／無効化に関する設定

WantedBy = multi-user.target

.timerファイル

[Unit]

Description=Runs myscript

[Timer]

OnBootSec=1min # startして、この時間後にこのタイマーを開始

OnUnitActiveSec=10s # 実行する時間の間隔

Unit=my\_service.service # 対象サービス

[Install]

WantedBy=multi-user.target

※普通に/etc/systemd/system内にあるファイルを覗くと具体例が見られる。

#### 詳細

##### 環境変数が必要な場合

[Service]

Environment=TEST\_VALUE1=display1

##### 仮想環境が必要な場合

##### トラブルシューティング

・一般

まずは動作状況の確認

sudo systemctl status my\_serviced

・journalctl

systemdが動かないときはジャーナル（システムログ）を確認すると良い

例：

journalctl | tail

journalctl -u my\_service

・サービス専用のログファイルを作りたい場合

[Service]

StandardOutput=file:/tmp/discordbotd.log # 必要であれば，標準出力のログ

StandardError=append:/tmp/discordbotd\_error.log # 標準エラー出力。appendは

実行ユーザーを変える必要がある場合

補足：systemd通常rootで実行される

[Service]

User=your\_user\_name

・Failed with result 'protocol'.

Type=notifyだと発生する？とりあえずType=simpleでやりなおす

#### レファレンス

##### [Service]

メインの設定セクション

|  |  |  |
| --- | --- | --- |
| 項目 | 説明 | 詳細／例 |
| Type | 起動完了判定を指示する。  simple：メインプロセスとして起動し，  起動時点で終了とみなす　[デフォルト]  forking：自身を複製して子プロセスとして起動。  oneshot：文字通り，単発で終える  notify：simpleと類似。利用可能になった時  にデーモンが systemd に信号を送る。  dbus：メインプロセスが指定のD-Bus名を  取得（接続された）したタイミングで起動  idle：simpleと類似。すべてのジョブが  終了するまで遅延する。 |  |
| ExecStart | サービスの起動コマンド  当然，**対象ファイルには実行権限が必要**  WorkingDirectoryを指定しない場合は，絶対パス表現で指定する必要がある点に注意。 | ExecStart=/usr/sbin/nginx  （補足）ExecStart は純然たるコマンドとして文字列を指定する点に注意  例）ExecStart=/bin/bash echo ‘hoge’ |
| ExecStartPre | 本Unit実行に先立つコマンド。 | ExeStartに準ずる |
| ExecStartPost | 本Unit実行後に引き続くコマンド | ExeStartに準ずる |
| ExecReload | 本Unit再読み込み時に実行されるコマンド |  |
| RemainAfterExit | プロセスが終わっても処理を続けるか | true, false で指定。 |
| Restart | サービスプロセス停止時の再起動条件  always： 常に再起動を試みる  no ： 再起動しない  on-success： 終了コード0 の時に再起動  on-failure： 0以外の時に再起動 |  |
| RestartSec | 再起動するまでの待ち時間（秒） | =1s |
| SyslogIdentifier | syslogでログを見分ける時の目印 | journalctl SYSLOG\_IDENTIFIER=hoge  で対象ログが確認できる |
|  |  |  |
| WorkingDirectory | 本体，及び作業ディレクトリ  これを指定して置くと，ソースコード内で相対パス表現が使える。 |  |
| TimeoutStopSec | 停止完了までに待機する時間の設定、指定時間内に終了しなかった場合、強制終了。 | SIGTERM, SIGKILL が対象 |
| KillMode | ExecStopで停止せずに残ったプロセスの処理方法  mixed |  |
| KillSignal | サービスの終了にどのシグナルを使用するか |  |
|  |  |  |

[Service] のつづき

|  |  |  |
| --- | --- | --- |
| 項目 | 説明 | 詳細／例 |
| BusName |  |  |
| Environment | 環境変数を制御する。  例えばTEST1という環境変数を作って  このように使う事ができる  ExecStart=/bin/echo ${TEST1} | =PATH=/otp/myapp:$PATH  =TEST1=value1 |
| EnvironmentFile= | 環境変数を設定したテキストファイルを読み込む | /etc/sysconfig/にテキストファイルを配置する事のが通例らしい |
| Type | 起動完了判定を指示する。  simple終了するまで遅延する。 |  |
| StandardInput | ターミナルの入出力の制御？  journal  null  file: ファイルパスを指定。（追記されない）  append:指定ファイルに追記する。 | =file:/tmp/test.log  =append:/tmp/test.log |
| StandardOutput |
| StandardError |
| IgnoreSIGPIPE |  |  |
| EnvironmentFile | 環境変数を記述したファイルの事？ |  |
| KeyringMode |  | =shared |
| LimitNOFILE |  | =16384 |
| LimitNPROC |  | =1 |
| NotifyAccess |  | =main |
| CapabilityBoundingSet |  | =CAP\_NET\_ADMIN CAP\_NET\_BIND\_SERVICE |
| ProtectHome |  | =true |
| ProtectSystem |  | =full |
| PrivateTmp | このサービス専用の/tmpと/var/tmpを用意するかどうか |  |
| PIDFile | メインプロセスのPIDファイル |  |
| StartLimitInterval | 起動制限の基底となる時間間隔を設定 |  |
| StartLimitBurst | StartLimitIntervalで設定した時間の間に許容される起動回数を指定 |  |
| StartLimitAction | 起動制限に達したときのアクション指定。  none次の起動が禁止  reboot停止処理の実行後にreboo  reboot-immediate停止処理を行わず直ちにreboot  reboot-force | def.=none |
| User |  | def.=root |
| Group |  | def.=root |

##### [Unit]

説明や依存関係などを記述。動作には影響しないメタデータ。

|  |  |  |
| --- | --- | --- |
| 項目 | 説明 | 詳細／例 |
| Description | サービスの説明 |  |
| After | このUnitより先に起動するべきUnit  (opp) Before | After=apache2.service  複数の場合は空白で区切る |
| Before |  |  |
| Requires | サービスの有効に必要な他のユニット  (opp) Conflicts |  |
| Conflicts |  |  |
| Documentation |  |  |
| ConditionPathIsDirector |  |  |

##### [Timer]

起動時間に関する設定。※あまり記述してあるサービスが見られない。

* .timerファイル専用らしい。

|  |  |  |
| --- | --- | --- |
| 項目 | 説明 | 詳細／例 |
| Unit | 対象のサービスを指定する。 | =my\_serviced.service |
| OnCalendar |  | =\*-\*-\* 8:00 |
| OnUnitActivateSec | 実行する時間間隔 [秒] | =10s；=1min |
| OnBootSec |  |  |

##### [Socket]

※あまり記述してあるサービスが見られない。

|  |  |  |
| --- | --- | --- |
| 項目 | 説明 | 詳細／例 |
|  |  |  |

##### [Install]

サービス有効化／無効化に関する設定。このセクションがないとenableできない！！

|  |  |  |
| --- | --- | --- |
| 項目 | 説明 | 詳細／例 |
| WantedBy | ユーザーのターゲット  enable時にこのUnitの.wantedディレクトリにリンクを作成する。 | WantedBy=multi-user.target |
| Requiredby | enable時にこのUnitの.requiredディレクトリにリンクを作成する |  |
| Alias | enable時にこのUnitの別名を用意 |  |
| Also | 本ユニットと共にインストール，アンインストールされるユニットの一覧 |  |
| DefaultInstance | ？？？ |  |

#### 具体例

ある環境変数に値が入るまで待つ

ExecStartPre=- /bin/sh -c “while [! -f $XAUTHORITY ]; do sleep 1; done”

## 雑多情報

使用可能な開発言語はC，PHP，Java，Perl，Python，Rubyなど。

Ruby：Perlを基にしている。日本で開発。

### デスクトップ

LXDE C言語で開発。GTK+ツールキットを利用。

LXQt C++で開発。Qtツールキットを利用。

PIXEL Raspberry Pi 向きのデスクトップ環境。PIXEL for PCと言うPC用OSもある。

# bash（シェルスクリプト）

## 概要

### 基本

シェルの役割は，ユーザーにカーネルにアクセスできる窓口を提供する事。

シェルには様々な種類があるが，大きくsh（Bourne Shell）系とcsh（C Shell）系がある。

・sh系 ... bash（Bourne-Again Shell），ksh（Korn shell），zsh（Z Shell）

・csh系 ... tcsh

また，テキストファイルに一連のコマンドをまとめる事ができる。拡張子は.sh。

.shファイル内では関数の記述などもでき，少しだが構造化的な書き方もできる。

シェバンは

#!/bin/sh

#!/bin/bash # bashを使う場合

実行の仕方

#Terminal

source test.sh # 補足　sourceはbash特有のコマンド（らしい）

bash test.sh # こちらの呼び出し方の方が多いかも

test.sh # ファイル名から直接呼び出す場合はデフォルトのシェルで実行される。

. test.sh # source と同様

実行方法の違いによる特徴

|  |  |  |  |
| --- | --- | --- | --- |
|  | 実行権限の  無いファイル | シェル変数 | 備考 |
| ファイル名直接 | 実行できない | 破棄 |  |
| bash | 実行可 | 破棄 |  |
| source | 実行可 | 維持 | 実行中の状態を有効にしたまま，追加で実行  ※source はbash特有のコマンド |
| . （ドット） | 実行可 | 維持 | 結局のところsourceと効果は同じらしい？  2021-05-20 |

### bashとshの違い

違いに迷うようなら，shを使っておいた方が無難。但しそれは内部ではbashという矛盾，といった感じ。

|  |  |  |
| --- | --- | --- |
|  | bash | sh |
| 正式名 | bourne again shell | bourne shell |
| 特徴 | Linuxではデフォルトのシェル。  shの互換として開始された。  shより多機能である場合が多い。 | POSIX standardで記述されたプログラミング言語。  他のPOSIX言語に移植可能 |
| 補足 | 明示的にbashを使いたければ  !/bin/bashとしてbashを使えば良い。 | Linuxでは大体の場合，/bin/bashへのシンボリックリンク。但しPOSIXモードで動く。  ※BSDs，Solarisなどでは異なる（らしい） |

確認の仕方

echo $SHELL # デフォルトシェルの確認ができる

ls -l /bin/sh # シンボリックリンクのリンク先を確認できる

### 括弧

（）：丸かっこ

・サブシェルで括弧内のコマンド群を実行する。

・$() ⇒ `` とほぼ同等。つまり$() 内の文字列をコマンドと見做して実行する。

SEED=$(expr $(date +%M) + $(date +%S)) # bash 方式

echo $SEED

echo `date '+%Y/%m/%d'` # Sh 方式

・funcname(){} 関数を表す

・(()) 算術演算を表す。Cの形式の演算が使えたりする（らしい）

｛｝：波括弧

・変数展開。変数名と文字列を明示的に区別できる。

STR="user\_**${**NAME**}**\_logged\_in!"

・カレントシェルでカッコ内のコマンド群を実行する。

{

echo "PASSWD follows"

cat /etc/passwd

} >output.txt

※関数での func(){} のブラケットは正にこういった使い方。

［］：角括弧

・=test：式の結果を、0が真、1が偽の真偽値で返す。

if [ "x$1" = "x--help" -o "x$1" = "x-h" ]; then # 正にifでの使い方がそう。半角スペースが必要

### man

マニュアルを表示する

書式：man [options] [セクション番号] 調べる語句

man apt # apt のマニュアルが表示される q で閉じる

オプション

-k **部分一致**する単語のコマンド名を検索　(also)--apropos

kはkeyword から来ているらしい

aproposコマンドと同じ

-f **完全一致**する単語を検索　(also)--whatis;

fはfully matched keywordから来ていると予想

whatisコマンドと同じ

操作

検索 / キーに続いて検索したい文字列を入力してEnter nで次へ Nで戻る

セクション番号　※指定しない場合はセクション番号の小さい方のマニュアルが表示される

例：apt-key(8) を読め、と言われた場合

man apt-key 8

|  |  |  |
| --- | --- | --- |
| セクション | 備考 | 覚え方 |
| 1 ユーザーコマンド | だれもが実行できる | ユーザーファースト |
| 2システムコール | カーネルの関数 |  |
| 3ライブラリ呼び出し | ライブラリ関数 /lib？ |  |
| 4 特殊ファイル | 通常 /dev 配下 |  |
| 5 ファイルの書式と慣習 | 例 /etc/passwd | 書庫(5) の管理の慣習 |
| 6 ゲーム |  | 無(6)理ゲー |
| 7その他 | マクロパッケージなど |  |
| 8システム管理コマンド | root用 | rootのooを8と見立てる |
| 9カーネルルーチン | 非標準 |  |

この番号は man man で確認できる。

n 新しいドキュメンテーション：よりふさわしい場所に移動されるだろう

o 古いドキュメンテーション　猶予期間として保存されているもの

l 独自のシステムについてのローカルなドキュメンテーション

### シェルスクリプト用のコマンド

clear ターミナルの画面をまっさらにリセットする。

history bashのコマンド履歴を表示する

この履歴はそれぞれのユーザーのホームディレクトリに.bash\_history というファイルに記録されている。

その為，cat .bash\_historyでも表示が可能

書式：history [options] [ファイル名]

historyコマンド実行後に，!番号　する事で特定のコマンドを再実行できる。

$ !2 #２つ前のコマンドを再実行する。

!! 直前に実行したコマンド

!v 最近実行したvで始まるコマンド

time 処理にかかる時間を計測する

書式：tiem *command*

time xz mydata\* # 基本的な使い方

time (sleep1; echo hoge) # 複数コマンド 関数化しても良い。

(time sleep 1) > time.log # ログの記録

オプション

-p *command*

watch 指定したコマンドを一定時間ごとに繰り返し実行

デフォルトでは2秒ごとにコマンドを実行

watch date # 2秒ごとに時間を確認。このコマンドの効果が一番良く分かる。

watch cat /proc/meminfo # メモリの使用状況を確認

watch dd # 20.09.17 まだ試したことないが，有益かも？

-n 実行間隔の秒数を指定 --i**n**terval *sec* ※-sと考えがち。注意。　ってか-sにしてよ…

watch -n 10 ps # 10秒ごとにpsコマンドを繰り返す

alias コマンドのエイリアス（別名）を設定する。

書式：alias [options] [*name*[*=value*]]

alias ll='ls -alF' # ll というコマンド(alias)を新たに作成する。内容は ls -alF

# すでに同名のエイリアスがある場合は上書きされる

alias # 現在登録されているaliasの一覧を表示

unalias # ll というコマンド(alias)を削除する。

basename *fille\_path* ディレクトリのパスのみを取り出す。

ldd 共有ライブラリへの依存関係を表示する。 必要な共有ライブラリなどの情報が表示される。

List Dynamic Dependencies の意らしい。

書式：ldd [options].. *file\_name*

ldd /bin/bash

tmux 端末上で複数の仮想端末を開く

#### type

コマンドの「素性」を調べる

組み込み関数です シェル自身が実行する，最も基本的なコマンドである事を意味する。

エイリアスです 別名設定がされている事を意味する。通常 .bashrcで定義されている。

（参照）alias（コマンド），[環境変数：設定ファイル](#_設定ファイル)

ハッシュされています 要約関数：別の短い固定長の値を得る関数　？？？

予約語 for などの予約語

コマンドが実行される際，まずエイリアス，次にbuilt-in-commandに該当する名前を探し，最後にPATH内を先頭から探す。つまり，PATH 内にも存在しないコマンドは実行する事ができない。

書式：type [options] *command\_name*

type cp

オプション

-a 同名で実行可能なコマンドを全て表示する

紛らわしいコマンド名比較

file 　ファイルの情報を表示（種類など　例：ASCII text）

stat 　ファイル（など）の詳細情報を表示。（アクセス権，inode番号，タイムスタンプなど）

info 　manより詳しいマニュアルの表示

type 　コマンドの素性を調べる。（例　alias）

### その他のシェル

csh

ksh

tcsh

～シェルスクリプト用のコマンド　続き

tee 標準入力の内容を対象ファイルに保存。標準出力にも表示される。

書式：tee [options…] [*file\_name*…]

基本的に標準入力から受け取る（パイプを使用）。名前は水道管などのチーズから来ている。

ファイル名が指定されなかった場合は何も起こらない。（標準出力にのみ表示）

dmesg | tee log.txt # dmesg の結果を標準出力と，ディスプレイに表示。

オプション

-a 追記する 　> を>>に変えるオプションと捉える。--append

参考）[リダイレクト](#_リダイレクト)

xargs tee の逆で，標準入力からの文字列を，指定のコマンドの引数として使う

書式：xargs [options] [コマンド [initial-arguments]]

cat file.txt | xargs touch # file.txt の中身から新規ファイルを作成

ls |xargs file # カレントディレクトリ内にあるファイルのデータを見る

※「引数」と頭にインプットされがちだが、あくまでコマンドであると認識する事！

exit 現在のターミナルを終了する

logout ログインシェルをターミナルを終了

/bin/sh --version

もしくはecho $SHELLでも確認できる。（デフォルトシェルを格納する環境変数）

～シェルスクリプト用のコマンド　続き

info manより詳細なマニュアルを表示する

q　終了するにはqキーを押す

d　画面トップへ

l　１ページまえへ

?　正規表現でマニュアル内を検索

紛らわしいコマンド名比較

file 　ファイルの情報を表示（種類など　例：ASCII text）

stat 　ファイル（など）の詳細情報を表示。（アクセス権，inode番号，タイムスタンプなど）

info 　manより詳しいマニュアルの表示

type 　コマンドの素性を調べる。（例　alias）

whatis キーワードに**完全に一致**するマニュアルを表示 ≒man -f

(compare) apropos, whereis, [man](#_man)

apropos キーワードの**一部**に一致したマニュアルを表示 ≒man - k

(compare) whatis, [man](#_man)

which 指定したコマンドが格納されている場所を検索

環境変数PATHで設定されているディレクトリを検索する。(PATHに登録が無い場合は表示されない）

書式 which [パッケージ名]

which iptables # ⇒ もしも無い場合はno *package\_name* in ~~~ と表示される

which cp

> /bin/cp

whereis

指定したコマンドと，そのマニュアル，ソースコードが置かれている場所を絶対パスで表示する。

whereis ls

> ls: /bin/ls /usr/share/man/man1/ls.1.gz

紛らしいコマンド名比較

whereis コマンドの格納されるディレクトリから，コマンドのマニュアル，ソースの場所を表示。

which PATHの中から，コマンドが格納されている場所を検索する。

whatis キーワードに**完全一致**した名前をもつコマンドなどののmanページの一覧表示

which path? whereis

※whereとかwhichisとかいうコマンドは存在しない。

makewhatis # whatis データベースを生成，または更新する。catpath と言うファイルを基にしてマニュアルが作成される？

## プログラムlike

### 条件

if

if [ $hoge = $fuga ]; then # notは !=

echo "同じ文字列です"

else

echo

fi

注意点

[ の後ろ、] の前には空白が必要

コマンドの実行結果が空白かどうか判断する場合

originail\_dir="path/to/dir1"

target\_dir="path/to/dir2"

if diff -r "$originail\_dir" "$target\_dir" > /dev/null ; then

echo "ディレクトリ内のファイルは同一"

fi

文字列1 = 文字列2 文字列1と文字列2が等しければ真

文字列1 != 文字列2 文字列1と文字列2が等しくなければ真

-n 文字列 文字列の長さが1以上であれば真

-z 文字列 文字列の長さが0であれば真

### ループ

for I in a b c # python の書き方に近い

do # do ~ done でwhile ループができるらしい

echo $I # ループ内変数の使用

done #

for tx in \*.txt # ちょっと変わった使い方。カレントディレクトリの全ての.txtファイルが対象

do #

cp $tx $tx.cp # すべてコピーを作成

done #

for tx in ‘ls /home/ | grep hoge’ # こんな事もできるらしい

do

# 処理

done

<https://shellscript.sunone.me/input_output.html>

### 特殊変数

|  |  |
| --- | --- |
| 変数 | 内容 |
| $0 | スクリプトの名前 |
| $1, $2, ..., $9 | 第1引数, 第2引数, ..., 第9引数 |
| $# | 与えられた引数の数 |
| $\* | 与えられたすべての引数. 引数全体が"で囲まれている |
| $@ | 与えられたすべての引数. 引数一つ一つが"で囲まれている |
| $- | シェルに与えられたフラグ |
| $? | 直前に行ったコマンドの戻り値。  0=成功、0以外=失敗　と判断できる場合も多い |
| $$ | 現在のシェルのプロセス番号 |
| $! | 最後にバックグラウンドで行ったコマンドのプロセス番号 |

コマンドの結果

例えばlsの結果を格納する場合

files=$(ls)

### 特徴的な物

ブレース展開

チルダ展開

### ユーザー入力

#### read

read -p “input your name” name

### 構造化

#### 関数

function say\_hello(){ # function 関数名　パターン

echo hello

}

say\_something(){ # 関数名のみのパターン

echo $1 # 第一引数 $1を使う

echo ‘bye’

}

# 呼び出し方

say\_hello # 一番基本的な呼び出し方

say\_something hoge # 引数付き。空白で区切る（pythonと同じ）

ret1=`say\_something fuga` # 戻り値は基本的にない。**標準出力**の結果を受け取る。

ret2=$(say\_something fuga foo) # 別の方法。いづれの方法も =と変数名の間に**空白厳禁**

echo $ret1 # 戻り値の使い方。{$ret1} “$ret1” のような用法も見られる

引数

|  |  |  |
| --- | --- | --- |
|  |  |  |
| $0 | 自身のパス |  |
| $# | 引数の数 |  |
| $1 | 第一引数 |  |
| $? | 終了ステータス | 直前の関数の終了ステータス。正常なら0が返る。  異常終了時は1～255の値を返す |

## 起動時の自動プログラム

### systemdの場合

|  |  |  |
| --- | --- | --- |
| スクリプト／方法 | 権限 | 説明 |
| /etc/rc.local  【非推奨】 | root | Linux の起動時に自動的に root 権限で実行されるスクリプト。  当然書き込みにもroot権限が必要になる。  関連：/lib/systemd/system/rc-local.service |
| systemd | 選択可 | /etc/systemd/system に .service ファイルを作成する事で自作のサービス（デーモン）を作成する事ができる。 |
| cron | user | @rebootを使う事で似たようなことは可能。厳密にはuserログイン→cron起動のタイミングに実行されるらしい。（参照）cron |
| /etc/profile |  | ユーザがログインしたときに実行  ログインシェル用の環境変数の設定 |
| ~/.bashrc |  | bash を起動する度に毎回実行。/etc/profileと共に，環境変数の設定が主たる目的な為，Linux OS起動時に実行　という用途には向かない。 |
| autostart |  | 2017年前後のDebian系で使われていた。 |

補足

・rc.localの書き方

#!/bin/sh -e # このシェバンが無いと実行に失敗する事がある　-eは任意（意味？）

# ここに処理を記述

exit 0 # 必須ではないようだが付いている場合が多い

・rc.dはランレベルごとの起動スクリプトを格納するディレクトリ。例えば/etc/rc.d/rc5.dはランレベル5（GUI）での起動時に実行される起動スクリプトが格納される。

ここに格納されるのは /etc/init.d/ へのシンボリックリンクなので，SysVinitの起動時の方法（/etc/init.d/httpd start といった形）と実質的には変わっていない事になる？

・Ubuntu16.n 以降ではデフォルトでは/etc/rc.localが存在しない。**これはsystemdの使用する事が推奨されている**事が理由。但し，互換性の為にrc-local.serviceというデーモンが起動しており，root権限で/etc/rc.localを作成して実行権限(+x)を与えるとちゃんと機能する。

### その他

/etc/rc.d/rc.local systemdを採用したRedHat系(CentOS7など）に存在する。

2021-12-07 この情報の情報元は？？

foo.desktopなどとして「.desktopファイル」を作成する事でCUIでもデスクトップアプリを開始できる？2021-08-06

（参考例）

[Desktop Entry]

Type=Application

Name=PhotoFrame

Exec=/home/pi/PhotoFrame/photoframe /home/pi/PhotoFrame/

Terminal=false

Exec=/ 実行ファイルのフルパス 実行時のカレントディレクトリ という書き方

# コマンドレファレンス(bash)

## 基本中の基本

### 基本情報

シェルで使われるコマンドは/bin などに格納されており，以下の２つは同じ意味になる

cp text.txt copied.txt

/bin/cp text.txt copied.txt

#### コマンド

ファイルを開くときは，普通にフルパスを叩くと開く。もしくは [ アプリ名 path ]

cd 作業ディレクトリの移動。

※実行権限のあるディレクトリにしか移動できない

書式：cd *target\_directory*

引数なしでホームディレクトリに戻る。

または cd ~ でもホームディレクトリに戻れる

cd ~*user\_name* で指定したユーザーのホームディレクトリに戻る。※cd ~/user ではない！

pwd Print Working Directory カレントディレクトリを表示する

※同名の環境変数も存在する

sudo スーパーユーザー（rootユーザー）の権限でコマンドを実行

オプション

-E 実行ユーザーの環境変数を引き継ぐ

テキストエディタ

less １ページずつ表示。方向キーで上下スクロール。

more １ページずつ表示。エンターで１行ずつ表示する。

書式：more *text\_file\_name*

　1ページ進む：スペース

　1ページ戻る：b

　終了する：q

　先頭へ戻る：g（lessのみ）

　末尾へ進む：G（lessのみ）

nano 最小限機能のテキストエディタ。picoと呼ばれるエディタのクローンらしい。

emacs vimの次に人気のあるエディタ。ユーザー独自の拡張機能をカスタマイズできる。GUIもサポート。

開発環境として便利な機能を備えている。

vim Linuxでは一番人気だが，学習コストがかかる。サーバーではvimしか使えない事も多い。

書式：shutdown [options]　[時間]　[*”wall\_message”*]

※ウォールメッセージ(wall) ；ログイン中の全ユーザーにメッセージを送信する文字列

shutdown -h 23:00 # HH:MM 何時何分。-hはhaltの意。

shutdown -r +5 “reboot” # +数値　指定分後にシャットダウン。ウォールメッセージ付き。

shutdown now # 直ちにシャットダウン

shutdown # 時間指定なし，オプションなし → １分後に--poweroffが実行される。

man：システム管理コマンド(8)

※systemctlではpoweroff (もしくはhalt)な点に注意

オプション

-H --halt システムの終了後に電源ボタンを押して電気的に消す。

systemctl halt # これでも同じ結果が得られる

-P --poweroff　電源切断。

-h -H と -P ⇒ 終了して電源を切る。（デフォルト） **hourの意でない！**

systemctl poweroff # これでも同じ結果が得られる

-r 再起動

-k ログイン中の全ユーザーにメッセージを送る。シャットダウンはしない。

-c シャットダウンのキャンセル 参考）winでは shutdown **/a** なので注意

reboot 再起動

書式：reboot [options] [ 時間 ]

shutdown と異なり，reboot は**時間指定しないと即座に再起動処理に入る**ので注意。

rebootにはメッセージ指定が無い点に注意（参考：shutdown）

※ systemctl でも再起動は可能。

systemctl reboot # これでも同じ結果が得られる

systemctl start reboot.target # またはこのようにしても良い

（補足）rebootコマンドはhaltコマンドへのシンボリックリンクらしい

（補足）一定時間経過後に

オプション

-h 停止または再起動するまえに全てのハードディスクをstandbyモードにする。

lastコマンド？

echo $arg1 # 変数名の直前に$をつけて使う

(compare) export

export ：環境変数を設定する。

書式：export *arg* = *val* #

書式：export *arg\_name* #

export *var1* # シェル変数var1を環境変数に変える

export *var1*=hoge # これでもほぼ同じ意味

（参考）子プロセスが立ち上がる時に値渡しでコピー。子プロセスで値を変えても親プロセスには影響ない

（補足）この方法で設定した環境変数はプロセスの終了時（？）に無くなってしまう。永続化したい場合には~/.profileに記述することが推奨されている。（参照）[環境変数](#_環境変数_1)

set シェル変数と環境変数を全て**一覧表示**

※unset というコマンドが有ったりして，わりと紛らわしいコマンド名なので注意

(compare) env

unset シェル変数や環境変数を削除する。

書式：unset *arg\_name* 削除する際は**$が要らない**点に注意。

(opp) export（環境変数の場合）

env 全ての環境変数を表示。シェル変数は非対応。

printenv 一部，または全ての環境変数を表示。シェル変数は非対応。

(compare) set

#### ls

list segments の意。ファイルやディレクトリの情報を表示する。 ※dirでもいいらしい。

ls file?.txt # file1.txt, fileA.txt などが表示される

オプション

-l [path] # 指定path内のファイルをリスト形式で詳細表示する。ファイルの種類，アクセス権，

-rw-r--r-- 1 test staff 24 Jul 21 13:38 index.html

(1) (2) (3) (4) (5) (6) (7) (8)

(3)ハードリンク数，(4,5)所有者とグループ，(6)ファイルサイズ，(7)タイムスタンプ，(8)ファイル名　となる。

リンクの場合は hoge-link -> /etc/hosts などと表示される

タイムスタンプはオプションの無い場合，最終更新日(Modify)が表示。

タイムスタンプは6カ月以上前の場合は「日付と年」，それ以外は「日付と時刻」が表示

(1) の先頭の文字は

- ファイル，dディレクトリ，l シンボリックリンク，b ブロック？ c

(see also) stat, [ファイル権限](#_ファイル権限)

-F ファイル名の後ろにファイルタイプを表示 --classify

\* 実行可能ファイル

/ ディレクトリ

@ シンボリックリンク

--fulltime

-A [path] 隠しファイルを含むすべてのファイルを確認する。[.][..] は表示しない。

--almost-all

--a [path] 隠しファイルを含むすべてのファイル表示　よく -la で使用 --all

-A の結果に加え、カレントディレクトリ . と親ディレクトリ .. の情報も表示

（補足）ファイル名の先頭「.」がついたファイルは隠しファイル。このオプションで表示する。

d [path] 指定pathのディレクトリのみを表示する --directory

-h ファイルサイズの単位を読みやすい形式で表示

-i inode番号をつけて表示。 --inode

-u atime（アクセス時間）を表示する。

※ -uも-c も指定しないとmtime（ファイル内容を書き換えた日時）が表示される。

-c ctime（ファイル作成 ＝ inodeの変更日時）を表示する。

※ -uと-c を同時に使うと，cが優先されctimeが表示される。

-shal 現在のフォルダ以下

-R 子ディレクトリを再帰的に一覧表示する　--recursive

-r アルファベット逆順に表示する

-t 更新した時間順に表示

（補足）lsコマンドでタイムスタンプが変更される事はない。

(compare) tree, du

#### cat

conCATenate の意。ファイルの中身を全て表示する。ASCIIテキストが対象。

書式：cat [ *file\_name1* ] [ *file\_name2* ]…

ファイルを２つ以上選択すると，連結して表示される

引数やパイプを省略，または-（ハイフン）を指定すると，「ヒアリング」状態になる。

キーボード（標準入力）から入力した内容を，エンターを押すたびに標準出力に出す。

Ctrl-D が押されると終了。

最終行が「終了文字列」としてみなされる？

echo hoge | cat # パイプからcatの引数を得る

cat -n grub.cfg | grep 'uefi' # grepを併用してファイルの中身を検索（行番号付き）

オプション

-b 空白行を除いた行に行番号をつける (also) --number-non**b**lank　≒nl（オプション無し）

-n 空白行を含めた全ての行に行番号をつける (also) --**n**umber むしろnumber all と覚える

≒ nl -b a file

cat -n /etc/postgresql/11/main/postgresql.conf | grep autovacuum　# 特定項目の行番号を調べる

-e -vE と同じ

-E display $ at end of each line (also) --show-ends

-v 非表示文字と`^'や`^'を付けて表示 (LFDとTABは除く) (also) --show-nonprinting

(compare) less, more

（参考）zcat，xzcat，bzcat

補足情報

cat でsjis文字を読むと文字化けする事がある。Webに挙がっている解決方法は：

・vi を使う。

・icovやnkfを使う。

・python を使って文字コードを変える

cat text.txt |python -c 'from sys import stdin; print stdin.readline().decode("unicode-escape")' > test2.txt

#### grep

Global Regular Expression Print

ファイルの中身の文字列に対して正規表現(regex)を使って，**検索文字列を含む行**が抽出表示される

書式：grep [options] [-e] '*pattern*' [*target\_file*] [*target\_file*…]

grep 'hoge' test.txt # test.txtの内容の内，hoge を含む行のみ表示。最も基本的な使い方。

grep -c '^$' file.txt # 空白行の行数を調べる。 ^行頭と行末$を組み合わせてこんなことができる

grep PIN test.txt # ファイル名を指定できる（あまり使わないので忘れがち）

rsync --help | grep '\-n' # オプションの文字列を検索したい場合（エスケープシーケンスを使う）

ファイル名を省略すると，標準出力から受け付ける

オプション

-i 検索する際，アルファベットの大文字小文字を区別しない。 --ignore-case

-n 先頭にマッチした行の行番号も表示する。 --line-number

-c マッチした**行数のみ**を出力する。 --count

-v マッチしない行を出力する。 --invert-match

grep -v '[A-Z]' test.txt # text.txtの内，大文字アルファベットを含まない行を表示する

grep '[^A-Z]' test.txt # 大文字アルファベット以外の文字を1つでも使っている行を表示する

-f *file\_name* 検索パターンをファイルから読み込む。 --file= *file\_name*

grep -f pattern.txt test.txt

-[ *NUM*] マッチした行の前後[数値]行も出力する。 --content=*NUM*

-A [*NUM*] マッチした行の後[数値]行も出力する。

-B [*NUM*] マッチした行の前[数値]行も出力する。

-E 追加の正規表現が使える　≒egrep ※小文字eではだめ。

-F 検索パターンを正規表現ではなく，固定文字列とする　≒fgrep

egrep 拡張正規表現を使用できる

書式：egrep '検索内容' 検索対象

grep -E 'PINGT|pingt' test.txt # linux上では | がパイプとしてみなされるので’’が必要

grep 'PINGT|pingt' test.txt # この場合、’PINGT|pingt’ を検索してしまう。

拡張正規表現：|, +, ? など

fgrep 入力された文字列そのままの検索が出来る

fgrep 'ping-t|PING-T' test.txt # 'ping-t|PING-T' と言う文字列を検索する

Fixed Global Regular Expression Print の意らしい

##### 間違いやすい正規表現

^15$ # 行頭が1で行末が5である行 → 15 と書かれた行のみになってしまう。

^1.\*5$ # （比較）1で始まり5で終わる行

[0-59] # 0~5か9となる。あくまで一文字。

#.\* # #を含み，その後に何か文字がある行。注意点は # だけの行も対象，という点。

^[^a-z] # 先頭が小文字アルファベットではない行。

^t\* # 行頭がt，または行頭がtでない行 ⇒ 要するに全ての行になってしまう。

^$ # 空行。行頭と行末の間に何も文字がない事を示す。

a #

### メタキャラクタ

#### コマンド用

\* 0文字以上の文字列。ワイルドカード。（先頭のドットは対象外になる？）

例 \*[xyz].txt ⇒ 任意の文字列＋x,y,z の内の１文字.txt という意味。

正規表現では「0回以上の繰り返し」なので注意　wo\*w → ww, wow, woow...

? 任意の1文字 正規表現では「直前の文字の０個か１個」なので注意。

[] [] 内の１文字

$ 変数。また，一般ユーザーのプロンプトの末尾にも表示される。（rootは#）

\ バックスラッシュ。エスケープ文字。

メタキャラクタの意味を打ち消す。C言語と同じく[\\でバックスラッシュになる](smb://sshel93.local/usi-nas/data//でバックスラッシュになる)。

また，コマンド \ コマンド　とすると，コマンドを複数行で記述できる。

& コマンドの末尾に付ける事でバックグラウンドで動作する。

ping 192.168.1.1 &

- 標準入力からファイル名を受け取る事を意味する

gzip -cd test.tar.gz | tar ftv **-** # test.tar.gz を表示だけする。

; コマンドを続けて実行 例 command1; command2

空白 bashでは区切り文字として振る舞う。cat ‘my file’ だと my と file の２つを指定した事になる。

my＼␣file # この場合バックスラッシュによって空白がエスケープされ，「空白文字」として認識される。

! 番号 指定したhistory番号のコマンドの再実行。（参考）history

. ドット。シェル変数などを引き継いだままスクリプトを実行する

（参考）[bash:基本](#_基本)

クオーテーションの違いによる文字列の扱いについて

ダブル・クオーテーション「”」：　「$」，「`」，「\」，「"」は以外はエスケープして通常の文字として扱う。

シングル・クオーテーション「’」：　**全てエスケープ**され全て文字列になる。メタ文字除外すれば ’ ≒ ”

バック・クオーテーション「`」：　　囲まれた文字列をコマンドとみなす

#### ディレクトリ用

＼ （バックスラッシュ）文字列をエスケープする。

/ （スラッシュ）ディレクトリを表す。また，パスの頭に付ける事で絶対パスを示す。

~ （チルダ） ホームディレクトリを示す。＝ /home/*current\_user\_name*

注意点

~ は自身のホームディレクトリ，/home/*current\_user\_name* で、**ユーザーごとに異なる値**となる。

~*user1* などとしたの時だけ特別な挙動として =/home/*user1*といった感じになる

cd ~user1/test # cd /home/user1/test と同じ。（カレントユーザーが**誰であれ**そうなる）

cd ~/test # カレントユーザー名に影響を受ける。ユーザーが**user1 であれば**上の例と全く同じ挙動。

mkdir ~user1 # user1 が存在していれば /home/user1は既に存在しているので作成できない。

cd ~/user1 # 【注意！】 /home/current\_user\_name/user1 に移動しようとしてしまう。

./ カレントディレクトリ

../ 一階層上

cmd11 ; cmd2 cmd1 とcmd2を連続で実行する

cmd11 || cmd2 cmd1 が失敗した場合のみcmd2を実行する

cmd11 && cmd2 cmd1 が成功した場合のみcmd2を実行する

ちなみに，LinuxコマンドプロンプトであるbashはBourne-Again Shellの略。

#### 特殊文字

|  |  |
| --- | --- |
| 文字 | 効果 |
| \a | ASCIIのベル文字（07）を表示する（ビープ音を鳴らす） |
| \d | 「曜 日 月 日」の形式（例：Fri Jan 5）で日付を表示する |
| \e | ASCIIのエスケープ文字（033）を表示する |
| \h | ホスト名のうち最初の「.」までの部分を表示する |
| \H | ホスト名を表示する |
| \n | 改行する |
| \r | 復帰する |
| \s | シェルの名前を表示する |
| \t | 現在の時刻を24時間の「HH:MM:SS」形式で表示する |
| \T | 現在の時刻を12時間の「HH:MM:SS」形式で表示する |
| \@ | 現在の時刻を12時間の「am/pm」形式で表示する |
| \u | 現在のユーザー名を表示する |
| \v | bashのバージョンを表示する |
| \V | bashのリリースを表示する |
| \w | 現在の作業ディレクトリを、ユーザーのホームディレクトリからの絶対パスで表示する |
| \W | 現在の作業ディレクトリを表示する |
| \! | このコマンドの履歴番号を表示する |
| \# | このコマンドのコマンド番号（現在のシェルのセッション中に実行されたコマンドのシーケンスにおける位置）を表示する |
| \$ | 実効UIDが0の場合に#となり、それ以外の場合に$となる |
| \nnn | 8進数nnnに対応する文字を表示する |
| \\ | バックスラッシュを表示する |
| \[ | 非表示文字のシーケンスを開始する。これを使って、プロンプト中に端末の制御シーケンスを埋め込むことができる |
| \] | 非表示文字のシーケンスを終了する |

#### その他

$ LF Linuxの改行（コード）

echo “hoge$fuga” > result.txt # hoge までしか書き込まれない。

^M CR ウィンドウズで編集したファイルなどに付く。’\r’ でも良いらしいが？

-- 現在の行では，これ以降コマンドにオプションが登場しない事を示す。

touch -- -f # -f と言う名前のファイルが作成される

### リダイレクト

入力元や出力先を変更。標準入力の入力元はキーボード。標準出力の出力先（コマンドの実行結果など）はディスプレイだが，これをテキストファイルなどに変更（リダイレクト）できる。

先ずファイル・ディスクリプタ（数字の事）を理解する ※&をつけると送り先になる

0 標準入力

1 標準出力

2 標準エラー。シェル実行時に発生したエラーメッセージの出力先。

例えば ls \*.txt > output と結果を「output」にリダイレクトしているのに，

エラーはディスプレイに表示される。この出力先を変えたい場合に使う。

| パイプ。**コマンドの実行結果**などを，そのまま標準入力に渡す。

cat test.txt | grep PIN # 一番理解しやすいcat を使った例（意味はあまりないが）

> 出力先を指定。よくあるのが，結果をテキストファイルに保存する用途。(also) 1>

sh test.sh > result.txt # 結果をテキストファイルに保存 1> result.txtでも同じ。

1>&2 # 標準出力の出力先を，標準エラーの出力先と同じにする。>&2 など，>だけでも可。

2>&1 # 標準エラーの出力先を，標準出力の出力先と同じにする

sh test.sh >out.log 2>&1 # 標準出力と標準エラーの両方を out.logに保存

⇒ まず１の出力先を out.lognにする。その後，2の出力先を１と同じにする。

　⇒ つまり &1 = out.log のあとに &2 = &1 = out.log といった感じ。

# sh test.sh 2>&1 >result.txt # これでは2の出力を画面にしてから，1の出力先を変えているだけ。

# sh test.sh 1> 2> result.txt # こういった書き方はできない。まったくの間違い。

>> 出力先を指定し，出力先に**追記**。ファイルに追記する，などが可能。(also) 1>>

echo 127.0.0.1 >> hosts # hostsと言うファイルに ‘127.0.0.1’ と言う行を追記

< 入力元を指定 (also) 0<

cat <　test1.txt >> test.txt # これは cat test1.txt >> test.txt と同じ。

※この場合，左側から実行される　cat test1.txt が先，その結果を test.txt に追記。

<<　ヒアドキュメント **終了文字を指定**し，指定した入力元から入力　(also) 0<<

cat << world world と言う終了文字を指定。要するに，world が出てくるまでcatを使用。

<<<　ヒアストリング 指定した文字列をコマンドの標準入力に送る

cat <<< 'ping 192.168.1.15' # 標準出力に文字列が出るだけなので，何も起きない

### 操作方法

Tab を押すと文字補完機能を使える

### 環境変数，シェル変数

#### シェル変数

シェル変数は，変数を設定したシェル上でのみ有効な変数

arg1=1 # シェル変数を作成。変数を設定したシェル上でのみ有効な変数。

注意点として，変数，=，値の間に空白を作ってはいけない。

※export する事で環境変数にする事も可能。

　ちなみに，存在しないシェル変数を使っても特にエラーにはならない

（参考）[bash:基本](#_基本)

#### 環境変数

設定以降，ずっと有効で子プロセスに引き継がれる。

echo $PATH # 使うときは頭に$をつける。

HISTFILESIZE=0 # 代入する時には$はいらない

補足

・環境変数は**ユーザーごとに値の異なる，別々の設定**がされている。

・永続化したい場合にはスクリプトファイル ~/.profile にexportコマンドを記述する。（後述）

#### 代表的な環境変数

|  |  |
| --- | --- |
| 変数名 |  |
| PS1 | デフォルトのプロンプトを定義。  The primary prompt stringの意らしい。PS2も存在する。 |
| HISTFILE | コマンド履歴保存ファイルのパス。通常 ~/.bash\_history |
| HISTSIZE | **現在のシェル**（のみ）のコマンド履歴の保管数。通常=1000  参考）[シェルコマンド用のコマンド](#_シェルコマンド用のコマンド) |
| HISTFILESIZE | コマンド履歴数。~/.bash\_history （$HISTFILE）の保存（上限）数。  通常=2000　HISTFILESIZE=0 とすると履歴がクリアされる  ~/.bash\_history に保存されるタイミングは**ターミナルを終了した時**。 |
| HOSTNAME | ホスト名。winで言うところのコンピュータ名。  ※USERはUSERNAMEではないのに，紛らわしい。 |
| HOME | ログインユーザのホームディレクトリ |
| LANG | ロケール（言語設定）を格納 |
| PATH | コマンドやプログラムを検索するディレクトリのリスト  /binや/sbinなどが設定されている。/homeは基本的に含まれない。  export PATH=$PATH:/opt/local/bin　　いわゆる「パスを通す」例。  「：」は区切り文字。 |
| PWD | カレントディレクトリのパス Print Working Directory |
| USER | ログインしているユーザ  なお，ユーザー一覧は/etc/passwd で確認できる。 |
| LD\_LIBRARY\_PATH | 共有ライブラリの検索パス　Linker Loader の意らしい。 |
| EDITOR | デフォルトで起動するテキストエディタ |

その他

DISPLAY X Windowシステムの出力先。setenv DISPLAY localhost:0.0など

CPATH C言語のヘッダー置き場（？）

#### シェルの設定ファイル（スクリプト）

基本的に上から順番に実行されるらしい。詳細はman bash を参考にする。

|  |  |
| --- | --- |
| ファイル | 詳細 |
| /etc/profile | ユーザーログイン時に**全ユーザーが**読込み、実行する。  特に環境変数PATHをスクリプトで設定する。  ※これを編集する事はほとんどない。 |
| ~/.bash\_profile | ユーザーログイン時に各ユーザーが自身のhomeディレクトリから読み込み，実行する。特に環境変数をスクリプトで設定する。  デフォルトでは存在しない事もある |
| ~/.bash\_login | ユーザーログイン時に~/.bash\_profileが存在しない場合に読込み，実行される。  デフォルトでは存在しない事もある |
| ~/.profile | ~/.bash\_loginが存在しない場合に読込み，実行される。  Bourne互換シェルで（sh）で読み込まれる。  bash では .bash\_profile, .bash\_login, .profile の順番に読み込まれる。  （ .profile がパーツの一つとして組み込まれていることに注意） |
| ~/.bashrc | 対話モードの シェル（bash）を起動する度に毎回実行。 |
| /etc/bashrc | システム全体の関数とエイリアスに関する設定。  デフォルトのプロンプトを決める環境変数PS1などもある。 |
| ~/.bash\_logout |  |
|  |  |
| ~/.bash\_history | おまけ：前回ターミナルを閉じる直前までのコマンドが格納される。  これはコマンド履歴であって，スクリプトではない。 |
|  |  |

※これらのファイルはファイルブラウザやlsでも表示されないものもあるが，ちゃんと存在はしている。

⇒ ls -a であれば隠しファイルも表示される。

/etc/profile.d 昔の仕組み？本ディレクトリ以下にスクリプトファイルを格納。

## ファイル関連

Linuxで扱えるファイルの種類には以下のものがある。

・通常のファイル

・ディレクトリ

・シンボリックリンク

・特殊なファイル（ブロックデバイス，キャラクタデバイス，ソケット）

### ファイル情報

file ファイルの種類，情報を表示

（例）file hoge.txt

> hoge.txt: ASCII text

・fileコマンドを行うと，対象ファイルはアクセス時刻のみが変更される。

・ディレクトリを対象にする事も出来るが，*dir\_name* : directory と表示されるのみ。

stat 指定ファイルやディレクトリの詳細を表示する。

書式：stat [options] *file\_path*…

紛らわしいコマンド名比較

file 　ファイルの情報を表示（種類など　例：ASCII text）

stat 　ファイル（など）の詳細情報を表示。（アクセス権，inode番号，タイムスタンプなど）

info 　manより詳しいマニュアルの表示

type 　コマンドの素性を調べる。（例　alias）

(compare) [ls](#_ls_1) ⇒ -F(--full-time) オプションを使っても，アクセス権，所有者，ctimeしか表示されない。

#### find

ファイルを検索する。locateコマンドより複雑な条件での検索が可能。

書式：find [options] [*base\_path*] [検索式] [アクション]

base\_path に / (root) を指定すると全ディレクトリになる

オプション

-L 全てのシンボリックリンクをたどる

検索式

-attime n 最後にアクセスされた日にちで検索

find ./ -atime -3 # ここ２日以内にアクセスされたファイル

find ./ -atime 3 # ３日まえにアクセスされたファイル

find ./ -atime +3 # ここ３日より前にアクセスされたファイル

-mtime 最後に更新（修正）された日時　**m**odifyied time？

⇒ ctime では検索できないらしい。

-name *name* ファイル名／ディレクトリ名で検索

find subdir -type d -name test03 # subdirを起点にtest03を含むディレクトリを検索

find . -name \\*.txt # 拡張子が .txt のファイルを検索する

find . | grep ‘\.txt’ # これでも同じ事ができる

-perm 許可属性（パーミション）で検索する。

find . -perm a+rw # -perm -u+s；-perm -2000 など

-size ファイルサイズで検索

find ./ -size -100k # 100kより小さい容量のファイルを検索

find ./ -size 100k # 100kのファイルを検索

find ./ -size +100k # 100kより大きい容量のファイルを検索

-type 検索対象のファイル種別を指定。

このオプションを省略した時は，ファイルとディレクトリが両方表示される。（リンクは？

-type f 検索対象をファイルにする

**-type d** 検索対象をディレクトリにする

-type l 検索対象をシンボリックリンクにする

find /usr -type f | grep ‘pg\_ctl’

-user *user\_name* ファイルの所有者で検索

find ./ -user root

-maxdepth 指定した階層まで検索。

1にするとサブディレクトリはなし。

find . -maxdepth 1 -type f # -maxdepthはグローバルオプションなので、指定順序が先

-mindepth 何回層以降を検索するか指定。

～find続き

アクション

**-print** # 検索結果を改行で区切って表示（デフォルト）

-print0 # 検索結果をnull文字（\0）で区切って表示

-exec *command* # 検索結果のファイルに対してコマンドを実行

find test -exec echo foo {} \; # find の終端文字は ; かつエスケープが必要

find test -exec echo foo {} + # 14

find . -atime +13 -type f -exec rm {} \; #

末尾が　\；　別々にコマンドに渡す。引数が２つあったらコマンドを２回実行。

末尾が　+　　可能な限りまとめて渡す。引数２つを同時に投げる。

find . -atime +13 -type f -exec rm {} \; # 14日以上アクセスの無いファイルを削除

（補足）

・｛｝について；現在処理中のファイル名の事（らしい）

・xargs を覚えたが汎用性が高いし処理速度も速い。（特に-exec {}; と比較した場合）

!find # 最後に実施した検索条件でもう一度検索を行う。

### ファイル操作

#### Linuxのタイムスタンプ

|  |  |  |
| --- | --- | --- |
| atime | アクセス時間 | cat, moreなどで表示しただけでも変更される。 |
| mtime | 修正(mend)時間 | ファイル内容を書き換えた日時 |
| ctime | 作成(create)時間 | より具体的にいうなら，inode（※１）の変更日時 |

※１　inode：ファイルやフォルダに与えられる一意の整数値

タイプスタンプに関するTips

・atimeのみが変更されるコマンド例：cat, file

・touchでctimeを任意の時間には変更する事はできない。ctimeは現在時間以外での変更ができない。

・vi などを用いると，最初にatimeが変更され，編集が終了するとctime，mtimeが変更される。

・echo とリダイレクト（> / >>） した場合は ctimeとmtimeが変更される。

（関連）[touch](#_touch)

#### コマンド – ファイル操作

mkdir フォルダ作成

書式：mkdir [options] *directory1* [*directory2*]...

オプション

-m 権限指定で作成 --mode

mkdir -m 777 ./test # 指定の無い場合，ディレクトリなら777になる。

-p 親ディレクトリも一緒に作成。--parents

-v ディレクトリを作成後に，その結果メッセージを表示 --verbose

プチテクニック

mkdir dir\_name && cd $\_ # 作成と同時に移動

rmdir フォルダ削除。

但し原則**空のディレクトリしか削除できない**

（参考）rm -r　コマンドであれば空でなくても削除できる。

オプション

-p 対象ディレクトリの親ディレクトリも含めて削除。 --parents

rm の -r オプションと動作を混同しないように！

rm -r 自身より下位にいるディレクトリが削除される

rmdir -p 自身より上位にいるディレクトリごと削除する

（参考）rmdir -p で対象ディレクトリは空だが，親ディレクトリは空でない場合

⇒ コマンドは実行される。但し，親ディレクトリは削除されず，エラー出力もある。

-v (also) --verbose

rm ファイルの削除

書式：rm [options]... [file\_name1]

rm 2013100[6-9].log # sh で使えるメタキャラクタを使用し，複数のファイルを削除する例。

rm \*.data # 上に同じく。　※「\*」はsh上ではワイルドカード

ls ./target\_dir | xargs rm \*.data # フォルダ内ファイル全て削除。但しrm -r target\_dir の方が簡単。

-f 削除するかどうかの確認メッセージなしで削除する (opp) -i オプション

-i 削除して良いか確認する --interactive (opp) -fオプション

-r ディレクトリを（再帰的に）削除 ≒ rmdir　(also) -R, --recursive

オプション指定しないと**自身より下位にいるディレクトリが全て削除される**点に注意

-p 更新日時，所有ユーザなどのファイル情報を変更しない　--perserve[=ATTR\_LIST]

※rmdirの -pオプションと混同しないように！

strings

バイナリファイルの可視化部分を表示

書式：strings [option(s)] [file(s)]

#### cp

ファイルの複製

cp -r tmp/\* dir # tmpディレクトリにある全ファイルを./dirにコピー

# 対象をディレクトリ名だけにすると，現在のファイル名を保持する

-a 元ファイルの属性とディレクトリの構成を保持してコピーする。

-i 上書きする際には確認する --interactive

-f 確認メッセージなしで上書きする --force

-p ファイル属性（パーミッション，所有者，更新時間）を保持したままコピー。

※rootである必要がある　--perserve[=ATTR\_LIST]

mkdir やrmdir ではparentなので混同しないように！

-r ディレクトリを（再帰的に）コピー　(also) -R, --recursive

-v コピーの詳細情報を表示

関連コマンド：rsync

#### rsync

指定ディレクトリのファイルを同期させる。

複数マシンのデータを同期させる事も可能。

書式：rsync [オプション] コピー元 コピー先

コピー元の最後が / (スラッシュ) で終わるか終わらないかで挙動が異なる。

/ で終わる場合、コピー元のディレクトリ自体はコピーせずに、ディレクトリ内のファイルをすべてコピーするという挙動

オプション

-a なるべくコピー元のファイルと同一条件でコピーする。

-X 拡張属性(extended attribute)を保持 --xattrs

-s

-S nullの連続をsparse ブロックに変換 --sparse

補足：sparse　ブロックが空の場合にストレージ領域を効率よく使用する仕組みのこと

-n 結果のみを出力する。 --dry-run

-v 詳細を出力する。

-z 通信を圧縮。

-4 IPv4を使用

-6 IPv6を使用。

--delete

具体例

# ssh経由でリモートと同期させる

rsync –avz –e ssh user@192.168.1.1:/path/to/remote path/to/local

# 指定ディレクトリを除外してコピー　--exclueの引数は/path/to/copy/fromからの相対で指定

rsync --exclude ignore/dir/ /path/to/copy/from /path/to/copy/to

参考：<https://qiita.com/bezeklik/items/22e791df7187958d76c1>

#### mv

フォルダ/ファイル移動。又はファイル名変更

書式：mv (オプション)［元ファイル名］［新しいファイル名］

-r オプションがない事に注意

mv -f tmp/\* doc # tmpディレクトリにあるファイルやディレクトリをまとめてdocディレクトリに移動

mv file? subdir # メタキャラクタを使ってfileA, fileB...　などをまとめて移動する

mv file subdir/ # ディレクトリ移動の場合は最後に / をつける。移動先は subdir/file となる

mv dir/ subdir/ # ディレクトリ → ディレクトリの場合

-i 上書きする際には確認する --interactive

-f 同名のファイル，ディレクトリを確認メッセージなしで上書き --force

pbcopy クリップボードにコピー

locate ファイルディレクトリ名でファイルなどを検索する

※locateは専用のデータベースupdatedbを使用する為，findよりも高速

書式：locate [オプション]... キーワード

locate httpd.conf

(syn) find：詳細な検索条件，

(参考) info：manより詳しいマニュアルの表示

(参考) Ubuntuでは標準では入っていない。apt install mlocateが必要。

updatedb 同名のデータベースを更新。このデータベースはlocateコマンドで用いる。

設定ファイル：/etc/updatedb.conf

#### dd

データをブロック単位でコピーする。特にデバイス間（CDなど）のコピーも行える点が大きな特徴。

データのバックアップなどに非常に良く使われる。イメージファイルは.iso，.img，.raw の拡張子である事が多い。

dd if=”data.zip” of=”data2.zip”

dd if=/dev/zero of=/dev/sda bs=446 count=1 # /dev/zero は特殊なファイルで，ゼロクリアできる。

# 注意！bs, count のオプションが無いと，その**ファイルシステムの範囲内**をすべてゼロクリアする

if=*target\_file* 入力ファイル

of=*target\_file* 出力ファイル。省略すると標準出力に出力される（らしい）

bs=バイト数 １度にバッファに書き込むブロックサイズを指定 def.512

この値分メモリを占有するので，書込み速度は結構変わる。（足りない分をswapで補う為）

count=回数 入力ファイルから出力ファイルへブロックをコピーする回数を指定

status=

progress 進行状況を表示する

（参考）[/null](#_%2525252Fnull)

（補足）ddなどで使われるGiBやMiBというのは 1024で桁を変える数値の事。

※ちなみに、ギビバイト、メビバイトと呼ぶ。

例：1GiB = 1024MiB 1GB=1000MB

具体例（読込み）

sudo dd if=/dev/sdb | xz -v > hoge.img.xz # 読込み後に xz圧縮を行う　※気のせいか、すごく遅い気がする

具体例（書込み）

xzcat hoge.img.xz | sudo dd of=/dev/sdb # .xzファイルを解凍しながら書き込み

xz -dc hoge.img.xz | sudo dd of=/dev/sdb # 上の別バージョン1　※TABによる補完が使えない

unxz < hoge.img.xz -| sudo dd of=/dev/sdb # 上の別バージョン2（未確認 2021-03-30）

補足

・DVD/CDの.isoイメージの吸出しなどもできる。

・SDカード等のファイルシステムが修復不可能になった時、対象メディアが物理的に壊れていない限り、適当な.imgファイル（FAT32が望ましい）の上書きで復帰できる。

#### touch

空ファイルの作成，またはタイムスタンプの変更

書式：touch [options] *target\_file1*　*[target\_file2]…*

⇒ ファイルが存在する場合：オプションが無ければ**全てのタイムスタンプ**が**現在の日時**に変更される。

⇒ ファイルが存在しない場合：ファイルを新規作成。 (opp) -c(--no-create)

注意点

-aや-mオプションが省略されると， a,c,mすべてのタイムスタンプが変更される。

-t や-d オプションを省略した場合，**現在の日時**が使われる。

touchコマンドでは，ctimeは必ず**現在の日時**に変更される。（ctimeはinodeの作成日時）

タイムスタンプの変更比較：

cat atimeのみ変更

標準入力からの入力は受け付けない

# ×cat file.txt | touch # こういった事はできない！

-a atime，ctime のみを現在時間に変更。

(compare) -mオプション

-m mtime，ctimeのみを現在時間に変更する。

(compare) -a オプション

-c ファイルを作成しない　--no-create

⇒ ctimeを連想しがちだが，ファイルが存在しない場合の挙動を制御するだけ。

（参考） -mオプション，-aオプション

（関連）stat

～touch 続き

-d *datestring* 文字列を使ってタイムスタンプを変更する（a/mと併用）　--date= *datestring*

a/mの併用が無ければmtime, atiem両方が指定通りに変更される。

※ctimeも同時に現在の日時に変更される点に注意する。**ctimeを任意の値に変更する事は出来ない**。

touch -d "2019-01-01" file001 #

touch -d "1 Feb" file001 # 今年の２月１日に変更

touch -d "February 1" file001 # 上に同じ

touch -d "1:1" file001 # 今日の年月日の午前１時１分に変更 “01:01” でも同じ

touch -d "Sep 1 1927 23:58:59" # フルフォーマット

(compare) -tオプション，-rオプション

-t *timestamp* タイムスタンプを使って日時を変更する（a/mと併用）

書式：timestamp：[[CC]YY]MMDDhhmm[.ss]

※ -t からtimeを連想 → **時間のみの変更ではない**ので注意する！

touch -at 201201011122.33 file.txt # アクセス時間のみを2012年1月1日11:22:33に変更

(compare) -dオプション，-r オプション

（補足）-tや-dを使っても，ctimeだけはどうあがいても現在時間に変更される

-r=*filename* 対象ファイルのタイムスタンプを適用する（a/mと併用） --reference=*filename*

(compare) -dオプション，-r オプション

（関連）

stat → 指定したファイルの全てのタイムスタンプが表示される

[Linuxのタイムスタンプ](#_Linuxのタイムスタンプ)

### 圧縮／解凍

tar .tarアーカイブの作成／展開。**T**ape **AR**chives；元々は磁気テープ向きのアーカイブだった

Linuxでは**ディレクトリを直接圧縮する事はできない**。tarなどでアーカイブを作成してから圧縮する。

書式：tar -cf　*archive\_name*　*target\_path1* [*target\_path2* ]*…* -cxtのいずれかが必須

書式：tar -xf　*archived\_file*　[--directory=./extracted]

※xzなどは xz target だが，tarでは tar archive target\_args... なので注意

tar -cvf file.tar a.txt b.txt c.txt # ３つのファイルをアーカイブする

tar -cf subdir.tar subdir # subdirを.tarにまとめる。（デフォルトでは圧縮なし）

tar -xzf subdir.tar.gz # 展開

tar cjf subdir.tar.bz2 subdir # testdirを.tarアーカイブしてgz形式で圧縮

tar xJf test.tar.xz # 展開

オプション：オプションの – は省略できる。

-c 新しいアーカイブを作成。c--**c**reate

-x アーカイブからファイルを展開。慣習的に先。--e**x**tract --get

-t アーカイブの**内容を一覧表示**。慣習的に先。**T**est

-f *archived\_fn*　　アーカイブファイル名を指定。**ほぼ必須オプション**。　--**f**ile= *archived\_fn*

-f の注意点：-f の直後にはアーカイブファイル名が来るので，必然的に最後，もしくは単独オプションになる。

-v **処理の詳細情報**の表示 --verbose （冗長な）

-z gzipを通して圧縮／展開 .tar.gz又は.tgzができる。　g**Z**ip。(**G**NU zip)

-j bzip2(.bz2) を通して圧縮／展開（**j**ournaling systemはext3から対応）

-J xzを通して圧縮／展開。

bzip2より圧縮率が高い（大きい）から，大文字J（Journaling）と覚えておく。

（補足）xzファイルはwindowsの7zipで展開する事はできる。（圧縮はできない）

-C 展開先のディレクトリを指定。--directory=*dir\_path*

--exclude *path* 特定のディレクトリをアーカイブ対象から外す

応用

tar cvf - ~/test | xz > test.tar.xz # ファイル名を – とする事でパイプが可能

cpio .cpioアーカイブの作成／展開。**圧縮オプションはない**点に注意。

書式：cpio [options] oiptオプションの内のいづれかが必要

**基本的にパイプ**でファイルを受け取り、リダイレクトで保存ファイルを指定。

find . | cpio -o **>** ~/backup.cpio # findの検索結果を利用した例。（元ファイルは消えない）

> \*\*\*\*blocks # 正常終了するとこのような表示がされる

find . | cpio -o # 【 悪い例 】 この例だと標準出力に表示されるだけで意味が無い

-o # 新しいアーカイブを作成 copy Out

書式：cpio -o *target\_path.cpio* または cpio -o > *target\_path.cpio*

cpio -o < *name\_list* [**>** archive]

outは何かしら出力する，と覚えるといいかも。個人的にはmake out の方が覚えやすい。

-i アーカイブからファイルを展開 copy In

cpio -i **<** backup.cpio

gzip gzip 形式(.gz)でファイルを圧縮するコマンド。Windowsのzipファイルも展開（らしい）

書式：gzip [options] *file\_name1*...

オプション

-d 解凍（展開）　(also) --decompress ≒gunzip

-l 圧縮ファイルの情報を表示　--list

-c 元ファイルを残して標準出力へ。**c**atを連想して覚える　--stdout （STanDard OUTput；標準出力）

※tar コマンドでは -cオプションが Create なt点に注意！

gzip -cd test.tar.gz | tar ftv - # test.tar.gzの内容を展開せずに表示。-cdだと展開されないらしい。

※最後のハイフン(-) は標準入力からファイル名を受け取る事を意味する

-k 圧縮／展開の元ファイルを削除しない　--keep

-r ディレクトリ内を再帰的に圧縮 --recursive

※オプションの内容は xzコマンドとほとんど同じ。

gunzip gzip 形式(.gz)でファイルを展開するコマンド。gUNzip と言う事。

zip Windowsでも用いられるzip形式で圧縮するコマンド。厳密にはwindowsはpkzip形式。

書式：zip [options] アーカイブ ファイル名

標準では**入っていない事が多い。**aptやyumでインストールする必要がある。

sudo apt install zip

オプション

-e 暗号化する。(also) –enc

zip archived\_fn -r target\_dir

zip archived\_fn target\_file

unzip Windowsで用いられるzip形式を展開するコマンド。

オプション

-d *dir\_name* 指定ディレクトリに展開する

unzip -d hoge hoge.zip # ディレクトリ名を指定しなければならない

-p unxzなどの -c オプションに相当する。pipe の意だと思われる。

機器

bzip2 bzip 形式(.bz2)でファイルを圧縮／展開するコマンド。そこそこの圧縮率。

名前の由来はライセンス形式がBSD-styleな為。

複数ファイルを１つにまとめて圧縮する事はできない。tar を使う。

（補足）但し処理速度がいまいちで，gzipの完全互換には至らなかった。

-d 解凍（展開）　--decompress

-c 元ファイルを残して標準出力へ。**c**atを連想して覚える　--stdout （STanDard OUTput；標準出力）

bzip2 -c configure > configure.bz2 # 元ファイルを残して圧縮。

# 標準出力に出力されるだけ，のところをリダイレクトを使ってファイルに保存。

# ※ -k オプションは無い。

bunzip2 bzip 形式(.bz2)でファイルを展開するコマンド。

xz xz 形式(.xz)でファイルを圧縮。圧縮率が高い。

書式：xz [options] *target\_path*

7z の親戚。LZMA2アルゴリズムを使用する。2G超えのデータなどで

uni**X** **Z**ip?

-d 解凍（展開）　--decompress

-l 圧縮ファイルの情報を表示。（圧縮前のデータ量、圧縮率など）　--list

-c 元ファイルを残して標準出力へ。**c**atを連想して覚える　--stdout （STanDard OUTput；標準出力）

tar コマンドでは -cオプションが Create なので注意！ ≒xzcat

-k 圧縮／展開の元ファイルを削除しない　--keep

-r ディレクトリ内を再帰的に圧縮 --recursive

-v 進捗を表示 --verbose

unxz xz 形式(.xz)でファイルを展開。≒ xz -d

書式：unxz [options] *target\_filename*

（関連）xzcat

cat系コマンド

zcat でgzip 形式(.gz) のファイルの中身を標準出力へ送信。≒gzip -dc，gunzip -c

bzcat でbzip 形式(.bz2) のファイルの中身を標準出力へ送信。≒bz -dc，bunzip2 -c

xzcat でxz 形式(.xz) のファイルの中身を標準出力へ送信。≒xz -dc，unxz -c

xzcat hoge-rpi.img.xz | sudo dd of=/dev/sdb # .xzファイルを解凍しながら書き込み

unxz -c hoge-rpi.img.xz | sudo dd of=/dev/sdb # 上とまったく同じ　-c 標準出力へ　-d デコード

xz -dc hoge-rpi.img.xz | sudo dd of=/dev/sdb # 上とまったく同じ

unar 文字コード自動判別の解凍

※Windows で圧縮したファイルを Linux で解凍すると，文字化けが起こる。

ただし，unarは解凍のみで，ファイルの圧縮はできない。

標準のパッケージでは無いのでインストールが必要

sudo apt -y install unar

書式：unar [options] アーカイブ [files]...

オプション

-d アーカイブファイル名でディレクトリを作成

-p パスワードの指定

### リンク

Linuxでは，inodeというメタデータと，実体データと言う形でファイルが管理される。

inode番号はファイルシステム毎に管理される為，異なるファイルシステムへハードリンクを作成する事はできない。

inode

アイノード。unix系のファイルシステムで，ファイルを管理する為のID番号。

ハードリンク

・inodeを使った，実態データ（メモリ）へのアクセスファイル。C言語的に考えるならポインタ。

・ファイルの新規作成はハードリンクが１つある状態と考えると分かりやすい。

・ディレクトリへのハードリンクは作成できない。（循環を防ぐ）

・新しいハードリンクを作ると，以下のものが全く同じになる。

inode番号，ハードリンク数，ファイルサイズ，最終編集日付，パーミッション，所有ユーザー・グループ

所有ユーザー・グループは，最初に作ったユーザー・グループのものを維持するらしい 20.09.08

・シンボリックと異なり，ls -l の結果の先頭には何も表示されない。

・2つ以上のハードリンクが作成されている状態で片方をrmで削除すると，同じ中身のままファイルが1つだけ残される状態となる

シンボリックリンク

Windowsのショートカットのようなもの。元ファイルの場所を指し示すリンク。

ファイルの別名，分かりやすいpathの設定などに使われる。主にハードリンクへのリンクになるはず。

ls -l では一番左に「l（小文字エル）」が表示される。

**上書きできない**為，編集する際には**事前に削除**する必要がある。

まとめ：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | inode番号 | リンク先を編集 | 元ファイル削除 | ディレクトリ | 異なるﾌｧｲﾙｼｽﾃﾑ |
| ハードリンク | 元ファイルと同じ | 元ファイルも  変更 | ハードリンクが１つでもあれば存在し続ける | × | 作成できない |
| シンボリックリンク | 別　全くの別ファイルと考える |  | 無意味なシンボリックリンクが残される | 〇 | 作成できる |

次ページに続く

ln

ハードリンクを作成する

書式：ln [オプション] *target\_path*　*link\_name*

ln file1.txt link1 # file1.txt へのリンクlink1を作成。 file1.txt←link1 と言う方向

注意点 ：target\_path ← link 引数は対象が先、リンクが後

オプション

-s シンボリックリンクを作成する --**s**ymbolic ※省略するとハードリンクが作成される。

ln -s ../file1.txt sysbolic # 1つ上のディレクトリにアクセスする

windowsの**s**hortcutに似ているからs　と覚えても良さそう。

-f

外部ストレージ内にはリンクを作れない？2021-09-09

### ファイル権限

#### chmod

##### 基本

書式：chmod [options] [対象] [操作] [権限]　*file\_name/dir\_name* (abrr.) CHange MODe

対象 　u：Owner　g：Group　o：Other　a：すべて

操作 　+：権限を付与　-：権限を削除　=：指定した権限にする

権限 　r [4]：読み込み　w [2]：書き込み　x [1]：実行

※それぞれ，ビットフラグになっており，例えば7はrwx となる。

（例）

chmod o+x /var/www/html/ # 対象ディレクトリに，その他ユーザーの実行権限をつける

chmod a+x *file\_name* # 全てのユーザーに実行権限をつける　例：644→755

chmod a=rx,u+w *hoge* # 全てのユーザーに読み実行権限をつけて，所有者には書込み権限もつける

chmod u=rw, go=r *file\_name* # 所有者に読み書き，グループとその他に読み取り

find /src/dir -type f -exec chmod 644 {} ＼; # find を使った合わせ技

find /src/dir -type f -print0 | xargs -0 chmod 644 # これでも良い

オプション

-R 対象ディレクトリ以下のファイルのパーミッションも変更

備考：

・chmodが実行できるのは対象ファイルやディレクトリの**所有ユーザーかroot**のみ。

・デフォルト権限はファイル666，ディレクトリ777。

・マスク値を利用する事でデフォルト値は変えられる。（関連）umask

・各ファイルの所有権は ls -l などで確認できる（関連）[ls](#_ls)

・シンボリックリンクの場合，元ファイルのパーミションが変更される

許可された挙動のまとめ：

|  |  |  |
| --- | --- | --- |
|  | ファイル | ディレクトリ |
| 読み取り権限 | 内容の参照（cat, moreなど） | 参照（ls, findなど） |
| 書き込み権限 | 書き込み（vi, nanoなど） | ファイルの作成，削除 |
| 実行権限 | 実行 | 移動（cd） |

間違いやすい点

・ディレクトリに実行権限があっても，実行権限の無いファイルは実行できない。

（関連） stat, [ファイル権限](#_ファイル権限)

##### 特殊なビット

t：スティッキービット。ディレクトリの「その他ユーザー」につけるビット。

root以外のユーザーは，そのディレクトリ内のファイルの実行は出来るが削除ができない。

ls -l で確認すると，その他ユーザーの実行のところに xの代わりにtが付いている。

⇒ 対象はその他ユーザーだけではない。所有者であっても他のユーザーが作成したファイルは削除できない。

/tmpディレクトリが代表的。

chmod 1755 *dir* # 1000を加算した値を使う事とスティッキービットが付与

chmod o+t *dir* # 上と同じ効果。（常にo+t）

補足：chmod g+tや，ファイルにスティッキービットをつけるコマンドは実行自体はできる。

ただし，なんの変化も起こらない。

SUID，SGID

実行ファイルに設定される特別なビット

|  |  |  |
| --- | --- | --- |
|  | SUID | SGID |
| 意味 | 実行時に所有者の権限 | 実行時に所有グループの権限 |
| アルファベットで設定 | u+s | g+s |
| 数字を使った設定 | 4000  for(4) ユー（ザー） とでも覚える | 2000 |
| ls -l での表示 | ユーザー実行のところに  S（大文字エス） | グループ実行のところに  S（大文字エス） |

具体例

chmod u+s *file* # SUIDを付与（必ずu+sになる） SGIDは必ずg+s

chmod 4755 *file* # 4000を加算した値を使う

※セキュリティ管理の為に，新しくSUID，SGIDが設定されているファイルを定期的に監視すると良い。

chown ファイルの所有者やグループを変更する。**CH**ange **OWN**er

書式：chown *user\_name*　*target\_path* 所有ユーザーを変更

書式：chown [-R] ユーザ名:グループ名　*file\_name/dir\_name* 所属グループも同時に変更

書式：chown [-R] ユーザ名.グループ名　*file\_name/dir\_name*

オプション

-R *target\_path* 指定ディレクトリの子孫ディレクトリも再帰的に変更する (also)--recursive

※不思議と小文字オプション -r がない点に注意する。

chown -R :staffgroup dir # 所有グループ変更 ≒ chgrp -R staffgroup dir

-c 変更があるファイル毎に診断メッセージを表示

-f エラーメッセージを表示しない

具体例

chown staff:staffgroup file.txt # file.txtの所有ユーザをstaffにグループをstaffgroupに

chown :staffgroup file.txt # 所有グループのみ変更 ≒ chgrp staffgroup file.txt

chgrp staffgroup file.txt # 上に同じ

chgrp **CH**ange **GR**ou**P**

書式：chgrp *group\_name　target\_path*

オプション

-R *target\_path* 指定ディレクトリの子孫ディレクトリも再帰的に変更する --recursive, -r

umask マスク値を設定

通常マスク値はシェルの起動時にデフォルト値(777と666) に戻る。

マスク値もOwner Group Other になっている。

666 - マスク値 = ファイルのデフォルトパーミッション 例 666 – 066 = 600

777 - マスク値 = ディレクトリのデフォルトパーミッション 例 777 – 066 = 711

⇒ 700, 600にしたい時はマスク値を777にしないといけない（マイナスは考慮されない）

### その他

tree

書式：tree [ オプション ] [-H baseHREF] [-T title ]

オプション

-d ディレクトリのみ

-L level ディレクトリの深さ指定

tree -d -L 2 ./ # 現在のディレクトリから2階層したまでのディレクトリのみを表示

## テキスト操作／表示

### 制御コード

sedなどで使える（らしい）

コマンド入力の時はCtrl+ 該当文字

文字列入力の時はCtrl+V → 該当文字 となる

^@ NULL

^J LF(Line Feed)

^M CR(Carriage Return)

### コマンド

cut ファイルの任意のフィールドの文字列を抽出する

書式：cut [options][ *file\_name* ]

cut -d “ ” -f 2 file.txt # 空白で区切られた文字列の2つ目のフィールドをそれぞれ抽出

オプション

-c 文字数 抽出する**文字位置**を指定 --characters= 文字数

-d 区切り文字 区切り文字を指定（デフォルトはタブ） --**d**elimiter=区切り文字

-f *n*[,*m*] 抽出するフィールドを列番号で指定 --fields=

cut -d : -f 1,7 /etc/passwd # :を区切り文字として1番目のフィールドと7番目のフィールドを取り出す

cut -f 2 /etc/passwd # ２番目の文字を取り出す

od ファイルの内容を**8**/16進数で表示する **O**ctal **D**ump

書式：od [options][ *file\_name* ]

od -t x /bin/ls # lsコマンドのバイナリファイルの内容を16進数で表示。（デフォルトは8進数）

オプション

-t 出力フォーマットを指定 --format=フォーマット format **T**ype

c ASCII文字（またはバックスラッシュ付きのエスケープ文字）で表示

o[SIZE] 8進数で表示。SIZE未指定時4バイト区切り ※-t o2がデフォルト

x[SIZE] 16進数で表示。SIZE未指定時4バイト区切り

od -t x8 /etc/networks # 16進数8バイト区切りで表示

-o 8進数2バイト区切りで表示（-t o2 と同じ）**o**ct

-x 16進数2バイト区切りで表示（-t x2 と同じ）he**x**

-c ASCII表示（-t c と同じ）

head ファイルの先頭部分を指定して表示

書式：head [options] [ *filename* ]

オプション

-n 行数 指定した行数をファイルの先頭から表示 -行数　でも良い。 --lines=[+]**N**UM

このオプションを指定しない場合は先頭10行が表示

head -n 5 httpd.conf # 先頭から５行表示

head -5 httpd.conf # 全く同じ

※headとtailには **–l オプションが存在しない**

⇒ wc -l，split -l などと混同しない。本当に紛らわしいから統一して欲しい。

-c バイト数 指定したバイト数をファイルの先頭から表示　byte **C**ount --bytes=[+]NUM

tail ファイルの末尾部分を指定して表示

書式等はhead とほぼ同じ。headとは異なる部分のみ記述する。

オプション

-f Ctrl+Cなどで終了するまで対象行を表示し続ける。 --follow[={name|descriptor}]

※headやtailに-b オプションは存在しない！

pr ファイルを印刷用に整形する

### nl

ファイルに行番号をつけて表示する number the lines?

書式：nl [options]…　[ *filename* ]…

スタイル(style)

-b，-f，-h オプションで使う：

a 空行を含めた全ての行に行番号をつける　all nl -b a ≒ cat -n

**t** 空行を除いた行に行番号をつける（デフォルト）non-emp**T**y nl -b t≒ cat -b

n 空行に対して行番号を付ける　number **n**o lines

pBRE BRE（基本正規表現）に一致する行に対して行番号を付ける

オプション

-b 本文の行番号のスタイルを指定する --body-numbering=スタイル

（参考）[cat](#_cat_1)

-n 行番号の位置を指定する --number-format=書式

書式(format)

-nオプションでのみ使う：

ln 左揃え，0で埋めない **l**eft justified, **n**o leading zeros

rn 右揃え，0で埋めない **r**ight justified, **n**o leading zeros

rz 右揃え，0で埋める **r**ight justified, leading **z**eros

次ページに続く

～nl 続き

なお，オプションを何も指定しなかった場合「-v1 -i1 -l1 -sTAB -w6 -nrn -hn -bt -fn」

-v1 １行目を開始行とし

-i1 インクリメントは１行

-l1 空白行１行で１グループとする

-sTAB タブ区切り

-w6 行番号６桁

-nrn 行番号の位置：右揃え，0で埋めない -n rn としても良い。

-hn ヘッダ行：空行に対して行番号をつける

-bt ボディ行：空行を除いた行に行番号をつける

-fn フッタ行：空行に対して行番号をつける

fmt 1行あたりの最大文字数を指定してテキストを整形する

sort 行単位でファイルの内容をソートして**表示**（元ファイルは変化しない）

書式：sort [options] [ *filename* ]

オプション

-c ソートされているか確認

-b 行頭の空白を無視 --ignore-leading-**b**lanks

-f 大文字，小文字を区別しない --ignore-case

-r 降順 --**r**everse （デフォルトは昇順）

-t *sep\_str* 指定した文字を区切り文字としてフィールド認識する --field-separator=*sep\_str*

指定しないと**空白**が用いられる

sort -k 3 -t , test.csv # セパレータにコンマを指定し、3行目をキーにしてソート

-n 数字文字列を数値としてソート numeric?

-k *field\_num* ソート対象とするフィールドを1から始まる数値で指定 --key= *field\_num*

（関連）uniq

split 指定した行数でファイルを分割する

オプションを指定しないと1000行ごとに分割される

また，出力されるファイル名の末尾にはaa, ab,ac といった文字列が順番に付く。

書式：split [-行数] [[*in\_filename*　[ *out\_filename* ] ]

書式：split [-l 行数] [*in\_filename*　[*out\_filename*] ]

split -500 file.txt splited.txt # splitedaa, splitedab と言うファイルが出来上がる

オプション

-l 行数の指定

paste 複数のファイルを行単位で結合する。

file1の１行目とfile2の１行目を結合して新しい１行目を作るイメージ。

書式：paste [-d 区切り文字] [*filename1* *filename2*...]

cat file1

> Monday

> Tuesday

cat file2

> 月曜日

> 火曜日

paste -d : file1 file2

Monday : 月曜日

Tuesday : 火曜日

paste file1 file2 > unified # pasteは標準出力に出力されるだけ。保存したければこのようにする。

uniq ソートされているファイルにある連続して重複した行をまとめる

書式：uniq [options] [ *in\_filename* [*out\_filename* ] ]

sort file.txt > result.txt

uniq result.txt

・オプションを指定しない場合、重複した行を１つにまとめて表示

・out\_filenameを指定すると，標準出力に実行結果は表示されず，指定ファイルに出力される

ソートされていないファイルはcat の様にテキストの内容を表示する

オプション

-d 重複している行のみを表示 repeate**d** only --repeated

-u 重複していない行のみを表示 **u**nique only --unique

（関連）sort

wc ファイルの文字数や行数を数える **W**ord **C**ount

書式：wc [options] [*filename*]…

cat file.txt | wc -l

標準入力から読み込む事も可能。（ファイル名指定を省略する）

オプションを指定しないと，行数，単語数，文字（バイト）数を順に表示する。＝ -wcl

オプション

-c 文字数（バイト数）を表示

-l 行数を表示

-w 単語数を表示

diff ファイル差分を表示

書式：diff *filename1* *filename2*

nkf テキストファイルの文字コードを変換

書式：d

オプション

-j JISコード

-e EUC

-s SHIFT JIS

w8 Unicode(UTF-8)

### sed

ファイルや標準入力の内容を編集して**表示** Stream EDitor

「アドレス」と「コマンド」を組み合わせる。アドレスは行番号や正規表現で指定し，省略すると全ての行が処理対象になる。

※保存する場合はリダイレクトが必要

書式：sed [options] 編集コマンド [ファイル名]

書式：sed [options] -e *script1* [-e *script2* ...] [ファイル名]

書式：sed [options] -f スクリプト [ファイル名]

※sed に渡される引数は正規表現とみなされる為，ドット（ . ）などにエスケープ処理が必要 \. になる。

ファイル名を省くと標準入力から受け入れる。

（関連）tr

オプション

-e *script* 編集コマンドを指定 --expression= *script*

-f *script-file* 編集コマンドを記述したファイルを指定 --file=*script-file*

sed -f editfile.txt test.txt

-i ？？--in-place[=SUFFIX]

-n パターン空間の自動出力をオフにする

sed -n 5p *file\_name* # ファイルの５行目だけを表示する

sed -n 1,3p test.txt # test.txt の1～3行目を表示 ≒headとtailを組み合わせ

編集コマンド

y/文字列１文字列2…/文字列3文字列4…/ 文字列１を文字列３に，文字列２を文字列４に

[-e] s/*regexp*/*hoge*/ 各行の**最初に現れる**「regexp」を「hoge」に置換 s= start??

[-e] s/*regexp*/*hoge*/g 全ての文字列１を文字列２に置換g go next one (as well) とか適当に覚える事に

この時のgはフラグで，他には以下の様な物がある。

数字 行内でn番目に登場したものが対象になる

p 変更が合った行のみ表示。-n オプションと併用する。

[-e] /文字列/d **文字列が含まれる行**を削除

sed -e /^hoge/d test.txt #（補足）^は正規表現で「行頭にある文字列」を示す。

行番号*1*,行番号*3*d 行番号1から行番号3までの行を削除

sed 1,3d test.txt # test.txt の1～3行目を削除して表示

具体例：

sed -i -e 's/$/＼r/' source.txt #よく間違うのが's/＼n/＼r＼n/'では置換出来ないので注意

sed -e 's/Linux/リナックス/3' source.txt # 3番目に出現した'Linux'を'リナックス'に

sed -e s/pingt/hoge/g -e /^#/d test.txt # 置き換えして「#」で始まるコメント行を削除

sed /^#/d test.txt # text.txt から「#」で始まるコメント行を削除

### tr

ファイル内の文字を変換／削除するして**表示**　translateの略らしい

１文字の置換をする時は一番楽。（２文字以上の「単語」の変換ならsedの方が良い）

書式：tr [options] [文字列１] [文字列２]

tr a-z A-Z < /etc/hosts # /etc/hostsファイル内の英小文字を全て英大文字に変換して表示

cat /etc/hosts | tr [:lower:] [:upper:] # 一つ上のと全く同じ

tr -s [:space:] < file.txt # file.txt 内の連続したスペースをスペース１つに置き換えて表示

cat file | tr -s ‘ ‘ > re\_file # 〃 ＋ 新しいファイル名で保存する。

cat file | tr ‘＼t’ [:space:] # タブを空白文字に変換（表示するだけ）　※linuxでは / と＼は違う！

tr -d ac < file # 「file」ファイル内にある文字「a」と「c」を削除する

tr -d '^M' < file1.txt > file2.txt #　file1.txtの改行コードをCRLFからCRに変更

※注意点として，trコマンドは返還対象を**標準入力からのみ**受け付ける。つまりパイプなどを使う。

cat file.txt | tr PINGT pingt # file.txt 内のPINGT を pingt に変更して標準出力に表示

（関連）sed

オプション

-d 文字列１で指定した文字を削除 --**d**elete

-s 文字列１で指定した文字が連続した場合，１文字に置き換える --**s**queeze-repeats 押しつぶす

cat file.txt | tr -s “ “ > edited.txt # file.txt 内の連続する空白を1つにして edited.txtに保存

文字列の具体例

a-z 英小文字

A-Z 英大文字

文字クラス（character class）

※ 第２引数のみ，という使い方の場合uppder/lower以外は使えない。第１引数のみ，は可能

[:alpha:] 英字

[:lower:] 英小文字 ＝a-z

[:upper:] 英大文字

[:digit:] 数字 [1-9]

[:alnum:] 英数字

[:space:] スペース

(see also) sed

## パッケージ系

### 様々なパッケージ管理ソフト

|  |  |  |
| --- | --- | --- |
|  | 略称 |  |
| aptitude | apt | deb系のCUIフロントエンド。 |
| Red Hat Package Manager | rpm | yum系のCUIフロントエンド。 |
| Yum Extender |  | yumのGUI。当然rpm系。 |
| snappy | snap | １つのパッケージはsnap と呼ばれる。Ubuntuで標準。  Debian, Redhat問わず使える。仮想環境に対応。 |
| flatpak |  |  |
| Node Package Manager | npm | Node.jsのパッケージを管理する。 |
| Mintinstall |  | Linux Mintのパッケージ管理ソフト。deb系。GUIで管理するらしい。 |
| Synaptic |  | deb系とrpm系。GUIで管理するらしい。 |
| dnf | ？ | yumの後継。 |
| zipper |  | openSUSEのパッケージ管理ソフト。 |
| apkファイルマネージャ |  | Android専用のパッケージ管理ソフト。　類似：xapk |
| chocolatey |  | Windows環境でのパッケージ管理ソフト |

### AppImage

ディストリビューションに関係なくインストールできるようにした形式

#実行権限が必要

sudo chmod a+x sample.AppImage

# インストール

./sample.AppImage

### aptとyumの違い

|  |  |  |
| --- | --- | --- |
|  | apt | yum |
| ディストリビューション | UbuntuなどDebian系 | CentOSなどRedHat系 |
| 内部パッケージ管理 | dpkg | rpm |
| パッケージファイル形式 | .deb | .rpm |
| 設定ファイル |  | /etc/yum.conf |
| レポジトリのファイル | /etc/apt/sources.list | /etc/yum.repos.d （ディレクトリ） |
|  |  |  |

どちらもパッケージ管理ソフトで，自動ダウンロードやインストールを手軽にできるようになるという大まかな役割は同じ。

まぎらわしいオプション群

|  |  |  |
| --- | --- | --- |
|  | dpkg | rpm |
| インストールされているか | -l または --list | -q または --query　？ |
| アップグレード | -G または --refuse-downGrade  ⇒ 下位verは無視される | -Uまたは --upgrade  ⇒ 入ってなくてもインストール |
| 有無による挙動 | -Eまたは --skip-sameversion  ⇒ 同verは無視される | -F または --freshen  ⇒インストールされてない時は無視 |
| アンインストール | -r または --remove | -e まはた --erase |
| パッケージ詳細情報 | -s または --status | -qi または --query --info |
| パッケージが所有するファイル | -Lまたは --listfiles | -ql または --query --list |
| ファイルを所有するパッケージ | -Sまたは --search | -qf または --query --file |
| パッケージの依存関係 | -p ？？  またはapt **depends**（インストール後） | -qRpまたは  --query **--requires** --package |
| インストール中の詳細情報 | ない（標準で表示される） | -ihまたは  -I --hash |
| グループインストール | -Riでディレクトリを指定 | yumでgroupinstall  （rpmでは非対応） |

|  |  |  |
| --- | --- | --- |
|  | apt | yum |
| パッケージ詳細情報 | show パッケージの情報  showpkg パッケージの非依存関係を含めた**詳細情報** | info |
| アップデート可能なパッケージを表示 | list --upgradable  ※listは完全一致を検索 | check-update　全パッケージ  list updates　パッケージ一覧 |
| パッケージを検索 | apt(-get) search  ※aptのsearchは部分一致を検索 | search |

（補足）aptもyum もハイフン２つ型、省略型のサブコマンドはないらしい

### apt

#### 概要

aptツールはdpkgツールを拡張したツール群で，Debian系パッケージを管理する。

パッケージとは，ソフトウェアの実行ファイル，設定ファイル，ライブラリ，マニュアルなどが含まれているファイルの事。≒ソフトウェアだと思えばよい。

なお，現在ではapt-cacheとapt-get ながaptコマンドに統合されている。

ファイル形式は：<パッケージ名>\_<バージョン番号>-<リリース番号>\_<アーキテクチャ名>.deb

（関連） [dpkg（コマンド）](#_dpkg)，[/dpkg（フォルダ）](#_%2525252Fdpkg)

（補足）Red Hat系では rpmやyum（高水準）が使われる。

（補足）aptitudeの略らしい。

※実は .debファイルもインストールできる

sudo apt install ./zoom\_amd64.deb # apt を使う場合は ./が必要。無いとパッケージ名だと思われる。

sudo dpkg -i zoom\_amd64.deb # dpkg の場合はそのままで大丈夫

apt install firmware デバイスが認識されない場合。firmware-linux などを含む（らしい）

rpi-update ファームウェアとカーネルモジュールの最新版。但しβ版を含む為，非推奨

#### リポジトリの追加／削除

追加

追加したリポジトリは以下の場所に格納される

Ubuntu /etc/apt/sources.list.d/

削除

sudo apt remove --purge package\_name # まずアンインストール

sudo rm -f /etc/apt/sources.list.d/package\_name.list\* # このように関連ファイルを削除しても良い

#### apt-cache

パッケージ情報の検索・参照などを行う

サブコマンド

search *keywords* 正規表現パターンによってパッケージ一覧を検索する

show *package\_name* パッケージの詳細情報を表示する

showpkg *package\_name* パッケージのバージョンや非依存関係（※）などの**詳細情報**を表示

※**非依存**(reverse depends)するパッケージの表示（比較）apt-cache depends

（関連）showsrc （類似）rdepends

showsrc *bin\_package\_name* ソースレコードを表示する

（関連）showpkg

depends *package\_name* **引数のパッケージが依存**するパッケージ一覧 （関連）rdepends 非依存

apt-cache depends vlc

> 依存：vlc-bin… 推奨：vlc-plugin-notify…

pkgnames システム内のすべてのパッケージ名一覧を表示する（あまり使われないらしい）

apt-cache pkgnames # aptだとダメで，apt-cacheである必要がある

policy ポリシー設定情報を表示する（あまり使われないらしい）

statts 指定したパッケージの統計情報

#### apt-get

ネットワーク経由でパッケージを取得インストール・アンインストール

※プログラム名はメニューからプログラムを右クリック→Properties→コマンド　で確認できる

書式：apt-get [options] サブコマンド

オプション

-m ローカル環境のインデックスファイルが、サーバ上のインデックスファイルより古くても、無視して最後まで処理を続ける。 --fix-missing；--ignore-missing,

サブコマンド：

install *package\_name* パッケージをインストール／アップグレード

remove *package\_name* パッケージをアンインストール

remove --purge *package\_name* 設定ファイルも含め完全削除。なお apt purge *package\_name* でも可

update パッケージのデータベースを最新版に

upgrade 今インストールされているパッケージを最新版に

dist-upgrade ディストリビューション（≒OS）の更新に伴う更新

show *package\_name* パッケージの詳細情報を表示

search *package\_name* パッケージを検索(部分一致) (compare) dpkg -l

list *package\_name* パッケージを検索(完全一致)

[ listだけで使えるオプション ]

--installed インストール済のパッケージのみ

--upgradable

--all-versions

sudo apt list "git\*" --installed

clean 過去に取得したパッケージファイルを削除。

具体的には/var/cache/apt/archives にキャッシュされている「.deb」ファイルが全て削除

apt-get install などでインストールする際，パッケージファイルはまず同フォルダにダウンロードされる。放置しておくとディスク容量が圧迫されるので定期的に削除する。

#### apt-mark

パッケージを手動、または自動でインストールされたものとしてマークする。

書式：apt-get [options] {auto|manual} サブコマンド　パッケージ名１　[パッケージ名２...]

サブコマンド：

hold *package\_name* パッケージを保留としてマークする (opp) unhold

showauto auto指定のパッケージ一覧の表示 (syn) showmanual, showhold

（補足）apt-markはaptに統合されていない。

#### apt-key

GPG公開鍵を管理するコマンド。Ubuntu公式以外からパッケージを取得するには，このコマンドで認証する鍵情報を取り込んでおく必要がある。

GPG：GnuPG OpenPGP(Pretty Good Privacy) のGnu実装

補足：

apt-keyは2022年半ばに廃止となった。

wget -q -O - https://packages.something.com/gpg.key | sudo apt-key add -

Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).

以下は代わりの手法

（編集中）

apt-key　adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 3FA7E0328081BFF6A14DA29AA6A19B38D3D831EF

cd /tmp # 適当な場所に移動。この場合/tmp

wget https://packages.something.com/gpg.key # 適当な場所にgpg.keyをダウンロード

dest="/etc/apt/trusted.gpg.d/$2.gpg"

gpg --no-default-keyring --keyring temp-keyring.gpg --import hoge.key

chmod 644 "$dest"

"/etc/apt/trusted.gpg.d/$2.gpg"

（参考；スクリプト版）<https://hnakamur.github.io/blog/2020/12/10/alternative-script-for-apt-key-add/>

#### トラブルシューティング（apt）

##### could not get lock /var/lib/apt/lists/lock

sudo rm /var/lib/apt/lists/lock

sudo rm /var/lib/dpkg/lock

### dpkg

パッケージファイル「deb」ファイルを取り扱う(Debian PacKaGe)

パッケージのインストール先などを表示

設定ファイル：/etc/dpkg/dpkg.cfg

書式：dpkg [options] アクション

オプションは省略できるが，**アクションは省略できない**点に注意！

オプション

-E 同バージョンが既にインストールされていればインストールしない --skip-sam**e**version

-G 新バージョンが既にインストールされていればインストールしない --refuse-down**g**rade

sudo dpkg -Gi package\_file.deb

-R （オプション）ディレクトリを再帰的に処理

sudo dpkg -R --install packages\_dir # ディレクトリに格納されているパッケージをまとめてインストール

アクション

-i *package\_file.deb* ファイルのインストール　--install

-r *package\_name* 設定ファイルを残してパッケージをアンインストール --remove　（関連）-P

※apt-get には -r が無い事に注意する。

-P *package\_name* 設定ファイルも含め完全に削除。注意！大文字のP　--purge

-p *package\_name* パッケージの情報を表示

-l [*keywords*] インストール済パッケージの内，条件にあてはまるものを表示。 --list

dpkg -l git\* # ワイルドカードを用いた検索 → インストール済かどうか確認できる

dpkg -l | grep git # パイプとgrepを使っても良い

(compare) apt search

-L *package\_name* パッケージが所有するファイルの一覧　--listfiles オプション長いから大文字

(opp) -S; --search

-s *package\_name* インストール済のパッケージの詳細情報の表示

--status　 細かく（小文字）status

-S *pattern*… 指定したファイルがどのパッケージからインストールされたかを表示

--search 目を大きく（大文字）serch (opp) -L: --listfiles

-C インストール不完全なパッケージの表示 --audit

なぜauditがC？　監査は大きなcheck と覚えようか。。。

--configure

dpkg-reconfigure インストール済みのパッケージを初期状態に戻す

書式：dpkg-reconfigure *package\_name*

dpkg-reconfigure slapd # 使用例

（関連）[パッケージ（アプリ）](#_パッケージ（アプリ）)

ldconfig

特徴：ルート権限

共有ライブラリのキャッシュ /etc/ld.so.cacheを更新する為のコマンド。

/etc/ld.so.confを参照し，バイナリファイルである /etc/ld.so.cacheを更新する。

共有ライブラリを格納する為のディレクトリを新たに作成 → ldconfigでキャッシュを更新　と言った流れになる。

（関連）/etc/ld.so.conf [ファイル]

### yumとrpm

#### rpm

Red Hat系のパッケージ管理ツール　**R**ed Hat **P**ackage **M**anager

ファイル形式は：<パッケージ名>\_<バージョン番号>-<リリース番号>.<アーキテクチャ名>.rpm

設定ファイル：/usr/lib/rpm/rpmrc

オプション（グループ１）

-i *package\_filename* (also) --install

-e *package\_filename* (also) --erase なぜかremoveでは無いので注意

-U *package\_filename* アップグレード（なければインストール） --upgrade

rpm -U procmail-3.22-9.i386.rpm

-F *package\_filename* アップグレード（なければインストールしない） --freshen

併用オプション

-v **詳細情報**の表示　--verbose

-h **進行状況**の表示。#マークでプログレスバーの様に表示する。 --hash

※ほかのコマンドのように--verboseではない点に注意　-vはverifyで使われている

--test 実際には実行せずにテストを実施 なぜか-tは存在しない

rpm --install --test procmail-3.22-9.i386.rpm

--nodeps 依存関係を無視

rpm -e –nodeps postfix # 依存関係を無視してアンインストール

オプション（グループ２）

-q *package\_name* 併用オプションが無い場合は，指定したパッケージがインストールされているか

照会する。 --query

情報が欲しい時は基本的にこのオプションを常につける，と覚える

-V *package\_name* **インストール済**パッケージの検査。 --verify

併用オプションは -a と --nomd5のみ

※ファイルのサイズ，MD5チェックサム，所有ユーザ，所有グループ，タイムスタンプが検査される

※ -v （スモールV）はオプショングループ１にのみ存在し，詳細情報を表示する

rpm -V -all # インストールされている全パッケージの検査

併用オプション（-q, -V）

-a インストール済の全パッケージの表示 –all

--changelog 指定したパッケージの変更履歴を表示

※ --log というのは存在しない。省略オプションもない。

rpm -qp --changelog *package\_name*.rpm

-c **設定ファイル**の一覧表示 --configfiles ※-lオプションと混同しない

rpm -qc *package\_name* # インストールされた設定ファイルの表示

-l **パッケージファイル**に含まれるファイルの表示 --list　(compare) -R

併用オプションのpをつけるか付けないかでパッケージファイル／パッケージを変える。

-i 指定したパッケージ（ファイル）の詳細情報の表示 --info

-f *file\_name* 指定したファイルがどのパッケージからインストールされたかを表示 –file

rpm -qf /etc/yum.conf

（補足）dpkgでの同等オプションは -S, --search

-p *package\_file\_name* インストール前の**パッケージファイル**を照会対象とする　--package

--nomd5 MD5によるファイル改ざんを**検査しない no**md5

-R 指定した**パッケージ**が依存するファイルの表示　--requires　(compare) -l

rpm -qRp emacs-23.1-28.el6.x86\_64.rpm

その他

--version yumのバージョンを表示。-vオプションはグループ１のverboseなので注意。

rpm2cpio RPM系パッケージをcpio形式のアーカイブに変換する

#### yum

Yellowdog Updater Modified

Yellowdog Updater (YUP)が前身であり，名前もそれに由来。

パッケージ名などでインストールできたりする，高水準のパッケージ管理ツール。

インターネット上ないしはメディア上のパッケージ情報のデータベースを使い，依存関係を調整しながら，パッケージのインストール・アンインストール・アップグレードを行うことができる

内部ではrpmが実行されている。

設定ファイル：/etc/yum.conf

レポジトリ ：/etc/yum.repos.d ※ディレクトリ　拡張子.repoのファイルを配置する

.repoファイルの例

[*my\_repo\_id*]

name=*my\_repo\_name*

baseurl=http://myuser:mypasswd@hoge.zpp.jp/repo/

（関連）dnf

install *package\_name* パッケージのインストール

update [*package\_name*] パッケージ（毎）のアップデート。

パッケージ名を省略すると全パッケージ指定。

それでは困る場合もあるので，--exclude=php\* で除外パッケージを指定する事ができる。

remove *package\_name* アンインストール

info [*package\_name*] パッケージの**詳細**情報 ※infoはbashコマンドでも詳細情報

yum info php # php の詳細情報を見る。ダウンロードのリポジトリなど。

rpm -qip postfix-1.1.12-1.i386.rpm # 【比較】-pをつけて**パッケージファイル**の詳細情報みる

list [installed|updates|available|extras] [*full\_package\_name*]

パッケージの一覧，インストール済かどうか，バージョンなどを表示する。

インストール可能なパッケージは通常オンライン通信が必要。

yum list # インストール可能なパッケージとインストール済のパッケージが表示される

yum list *full\_pkg* # インストール済かどうかの確認。但し，**フルパッケージ名が必要**なので現実的でない

yum list | grep ‘ssh’ # こちらの方が現実的

yum list installed # インストール済のパッケージのリスト

yum list updates # アップデート可能なパッケージのリスト

yum list available # 利用可能なパッケージ

yum list extras # 利用できないパッケージ ⇒ なぜ表示する必要が？

serch *keywords* パッケージをキーワードで検索。

check-update [*package\_name*] アップデート可能なパッケージの表示

zypperではcheck-update**s**なので，紛らわしい事この上ない。

grouplist パッケージグループの一覧表示

groupinstall *group\_name* パッケージグループのインストール

※show はaptのサブコマンド。yumには無い。

#### 設定ファイル

##### /etc/yum.conf

proxy プロキシサーバーの設定

##### .specファイル

rpmbuildコマンドで.specファイルを指定する事でrpmパッケージを作成する事がｄ駅る

#### リポジトリの追加

/etc/yum.repos.dディレクトリに .repo と言うテキストファイルを作成すれば良い

[リポジトリID]

name=リポジトリ名

baseurl=リポジトリの場所

### flatpak

aptでインストール

apt -y install flatpak

# ひつようなパッケージ

sudo apt install gnome-software-plugin-flatpak

# リポジトリ情報の追加

flatpak remote-add --if-not-exists flathub https://dl.flathub.org/repo/flathub.flatpakrepo

flatpak list # インストールされたパッケージの一覧

flatpak update

flatpak install spotify.Client

パッケージのインストール

flatpak install same-package.flatpak # ダウンロードした.flatpakファイルから

flatpak install flathub org.same.package # https://flathub.org/apps　でアプリを探す

公式：<https://flatpak.org/setup/Ubuntu>

### zypper

openSUSEでパッケージ管理をするコマンド。**--を使わない**。省略サブコマンドはアルファベット２文字。

zypper in emacs\* # ワイルドカードを使用したインストール方法があるらしい

サブコマンド

install *package\_name* (also) in

update [*package\_name*](also) up

remove *package\_name* (also) rm

info *package\_name* 詳細情報の表示

search *keywords* パッケージの検索　(also) se

list-update**s** [options] アップデート可能なパッケージの表示　(also) lu

repos リポジトリの一覧表示 。(also) lr *省略はlrなのに，listrepとかではないのは何故*

refresh リポジトリの更新 ※updateではない

パッケージ名には任意の文字を表すワイルドカードが使えます。

### dnf

YUMの後継となるパッケージ管理ツールでFedora 22 からデフォルトのパッケージ管理ツール。

### snappy

#### 概要

Ubuntu生まれのパッケージマネージャー

公式サイト：https://snapcraft.io/

詳細

・Snap Storeで管理されているパッケージがダウンロードされる。

・その為、apt updateのようなコマンドは不要

・Snap Storeで管理しているパッケージは最新版で無いこともあるらしいので注意する。

インストール

# インストール

sudo apt install snapd

# インストール確認として、パッケージ検索してみる

snap find java

仮想環境の作成方法

（調査中）

# 仮想環境を作成するソフトsnapcraftをインストール

sudo snap install snapcraft

# aptでインストールしても良いようだが、versionが変わるらしい

sudo apt install snapcraft

#### サブコマンド

list インストール済のパッケージリストの表示

remove アンインストール

snap remove パッケージ名

find パッケージの検索（インストール済、未を問わず）

snap find {パッケージ名}

info パッケージの情報を表示。バージョンの確認などが行える

snap info {パッケージ名}

## 管理系

### コマンド

su ユーザー変更substitute user,

書式：su [-] [ユーザー名] [args]

ユーザー名を省略するとrootへの切り換えとなる。

sudo su # 間違いやすいのが，**su [root] にもルート権限が必要**な点

# rootへの切り替え時にパスワードを入れるのに，sudoが必要なのはなぜ？？

su user1 # 元のユーザーに戻る

exit # これでも元のユーザーに戻る

オプション

* あり 切り替えるユーザーの環境変数を読み込む
* なし 切り替え前のユーザーの環境変数を引きつぐ

su substitute user

locale 現在のロケールと使用可能なロケールの情報を表示

### Grub起動プロンプト

#### 概要

過去の産物ではあるがGRUB起動プロンプトを使う機会が稀にある。下のような画面。

特にブートパーティションに何かしらの問題が起こった場合などにに利用する事になる。

GNU GRUB version 2.04

grub>

（補足）

Web上でも，Grubプロンプトに関する情報はほとんど見つけられない。

唯一、有力な情報：<https://urotasm.hatenablog.com/entry/2018/01/12/232111>

#### チュートリアル

grub> ls # マシンに存在するパーティションを全て表示

(proc) (hd0) (hd1) (hd1, gpt3) (hd1, gpt2) (hd1, gpt1) (hd2) (hd2, gpt3) (hd2, gpt2) (hd2, gpt1)...

grub> ls (hd2, gpt2) # /boot/があるパーティションを探す。この場合はsdb2 ※１

lost+found/ var/ **boot/** etc/ media/ bin dev/ home/ lib lib32 lib64...

grub> cat (hd2, gpt2)/etc/lsb-release # 念のためディストリビューションを確認（任意）

DISTRIB\_ID=Ubuntu # 補足：これはUbuntuの場合。

DISTRIB\_RELEASE=20.04 # バージョン確認用のファイルは

DISTRIB\_CODENAME=focal # ディストリビューションにより異なるので注意。

DISTRIB\_DESCRIPTION="Ubuntu 20.04.3 LTS"

grub> ls (hd2, gpt2)/boot/ # カーネルイメージなどが存在するか確認

vmlinuz-5.11.0-36-generic vmlinuz-5.11.0-37-generic ...

initrd.img-5.11.0-36-generic initrd.img-5.11.0-37-generic ...

grub> set root=(hd2, gpt2) # ルートの指定（環境変数）

grub> linux /boot/vmlinuz-5.11.0-37-generic root=/dev/sdb2 # カーネルイメージを読み込む（？）

grub> initrd initrd.img-5.11.0-37-generic # 初期RAMを読み込む（？）

grub> boot # 起動

※１ grub2 ではディスク番号は0から，パーティション番号は1から始まる。その為、(hd2, gpt2) は3番目のディスクの2番目のパーティションとなり，sdc2と判明する。

⇒ 但し，実際には /dev/sdc2とした時はエラーとなり，**/dev/sdb2**で起動できた。原因不明。2021-10-17

（補足）2021-10-17 の時，boot後しばらくして (initramfs) と表示された場合は起動に失敗なのでリセットしてやり直す。

起動後のterminal：

sudo grub-install /dev/sdc # ブートローダーをインストール

（補足）パーティションでは無く，ディスクに入れるらしい。/dev/sdc1では無く，/dev/sdcという事。

（補足）この処理，ひょっとしたら不要かもしれない。2021-10-18

（補足）

これは2021-10-17に実際に不具合から復帰した時の記録でもある。Grub起動プロンプトは滅多に触る事が無いため，実践的な情報を集めにくい。

#### コマンドレファレンス（GRUB 起動プロンプト）

##### set

引数なしで現在の設定が見られる。

⇒ locale\_dir をみるとルートディレクトリがどれか分かる。

##### ls

通常のls(list of segment) と同じ。

ls -l (hd0, gp1)/efi # 使用例 パーティションはgpt1という情報もある。

##### kernel(linux)

書式：kernel *kernel\_img* [options]

grub> kernel /boot/vmlinuz-2.6.35 root=/dev/sda2 # ルートパーティションの指定

grub> kernel /boot/vmlinuz-2.6.35 1 # シングルユーザーモード

# grub> linux /boot/vmlinuz-2.6.35 root=/dev/sda2 # Grub2の場合

（補足）

・vmlinuz：内部にLinuxカーネル本体を包含する静的リンクされた実行ファイル。ブートローダが呼び出すので、カーネルでは無く、ブートローダが対応している形式でないといけない。

・コマンド名はGrub legacyの場合kernel，Grub2の場合はlinux 。

・Virtual Memory UNIXをlinuxも関連＋zipped→ vmlinux　と言う説がある。

オプション

init initの代わりのコマンドを指定

通常GRUBは/sbin/initを呼び出すが，その代わりに実行されるコマンドを指定できる。

例）init=*/bin/bash*

root ルートパーティションの変更

例）root=*/dev/sda2*

指定したランレベルで起動

数字(0-6) 1に関してはsでも可（下の行を参照）

s ランレベル１＝シングルユーザーモード (also)single

quiet 起動中のカーネルからの情報出力を抑制する

##### initrd

Linuxカーネルのブートの為に，初期RAMディスク（initial ramdisk）に一時的なルートファイルシステムをメモリに読み込む。

##### initramfs

≒initrd　使用するOSディストリビューションにより異なる？

#### 関連コマンド

##### grub-install

GRUB2 またはGRUB Legacy のインストール

OS起動中のterminal：

grub-install /dev/had

##### boot-repair

boot-repairというツールがある、と聞いた。

sudo add-apt-repository ppa:yannubuntu/boot-repair

sudo apt install -y boot-repair

2021-10-17にこれを用いてブートローダーの不具合を解消しようと試みたが、上手くいかなかった。

#### 用語

vmlinuz

gzip圧縮されたLinuxカーネルイメージ。

圧縮されていないイメージのファイル名は「vmlinux」。

Virtual Memory UNIX → Linuxも関連 → vmlinux　と言う説がある。

initrd.img

Linuxカーネルのブートの為に読み込まれる一時的なルートファイルシステム。

初期RAMディスク（Initial Ramdisk）と呼ばれる。

### GRUB

#### GRUB Legacy

##### 設定ファイル

boot/grub/menu.lst GRUB Legacy(GRUB1) の設定ファイル。

boot/grub/grub.conf ディストリビューションによってはこちら

root (hd1,2) # ルートパーティションを２番目のディスクの３番目のパーティションに指定

#### GRUB2

設定ファイルboot/grub/grub.cfgを直接編集する事は無く /etc/default/grub および /boot/grub.dディレクトリ内のファイルに記述して，grub-mkconfigコマンドで編集を反映する。

grub-mkconfig # 設定内容を/boot/grub/grub.cfg ファイルに反映させる。

sudo update-grub # コマンドはこうだ，という情報があるが？

設定ファイルの例：

menuentry ‘CentOS7, 3.10.0-957.12.2.el7.x86\_64’ –class centos{

insmod gzio

insmod part\_mdsos # **ins**ert **mod**ule の意らしい。ロードするモジュール。

set root=(hd0, 1)

linux /vmlinuz-3.10.0-957.12.2.el7.x86\_64 root=/dev/mapper/centos-root

initrd /initramfs-3.10.0-957.12.2.el7.x86\_64.img

}

設定ファイルの項目

menuentry メニューに表示するエントリ名

insmod ロードするモジュールの指定 **ins**ert **mod**uleの意。

set root ルートパーティションの指定

（例）set root=(hd0,4)

ルートパーティションを1番目のディスクの4番目のパーティションに指定

**ディスク番号は0**から，パーティション番号は1から数える。

※GRUB Legacy の場合は両方とも0から数える

linux

書式：linux *kernel\_img* [options]

オプション：

quiet 起動中のカーネルからの情報出力を抑制する

initrd 初期RAMディスクファイルの指定

（参考）[ディスク操作：デバイスファイル名](#_デバイスファイル名)

##### GRUBメニュー

先に：

etc/default/grubのGRUB\_HIDDEN\_TIMEOUTを以下の様にする必要があるらしい

※boot/grub/grub.cfg　ではないので注意。

GRUB\_TIMEOUT\_STYLE=menu # hidden →

#GRUB\_TIMEOUT=0

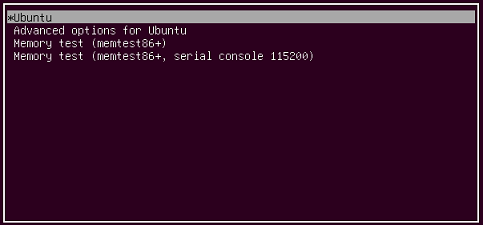
GRUB\_HIDDEN\_TIMEOUT\_QUIET=false # 追加

その後：terminal

update-grub # 必要ないかも？

コンピューターの起動時にShiftを押しながら起動するとGrubメニューが表示される。

こんな画面の事

選択してエンター ⇒

選んだブートローダーが立ち上がる

e　⇒ 設定ファイル編集画面

##### ランレベルの変更

grub設定ファイルの下の方にlinuxで始まる行があるので，

linux /boot/vmlinuz-...　root=UUID=23d7a25a- ~略~ -a358cf6be843 【ここ】 ro quiet...

↓

systemd.unit=rescue.target # これを追加（これはシングル［レスキュー］の場合）

（参考）[ランレベル](#_ランレベル)

##### grub関連コマンド

grub2-reboot

grub2-set-default

grub2-mkconfig

### バージョン管理・カーネル

lsb\_release ディストリビューション（OS）のバージョンの確認

-a すべてを表示

-s 値だけを表示 --short

-d 説明の表示 --description

-i ディストリビューターIDの表示 --id

-r リリースの表示 --release

uname 現在使用しているシステム情報を表示 (Unix Name)

オプション

-a 詳細な情報を表示

-s

-r

-m OSのbit数の表示

lsmod ロードされているカーネルモジュールの情報を表示する

（関連）/proc/modules ※まったく同じ内容

dmesg

カーネルのリングバッファ（※１）を表示。システムのデバイスの認識の仕方を確認できる。

（※１）システムが残すログ，システム起動時のメッセージの事

dmsgではなく、dmesgなので注意！

（補足）リングバッファの上限を超えた場合、古い情報は表示されない。

書式 dmesg [options]

e.g. dmesg | tail # tail は最終行から数行（def.=10）表示する。

オプション

-c リングバッファを消去 (also) --clear

### デバイス

#### 概要：

先に少し分かり難い事を説明

##### loop back device

一般的なファイルをあたかもハード・ディスクなどのブロック型デバイスであるかのように扱う機能の事。

例）

dd if=/dev/zero of=dummy.img bs=1M count=100 # ディスクイメージを作成

fallocate -l 1G dummy.img # こんな方法もあるとか？

sudo losetup /dev/loop0 dummy.img # 作成したイメージを/dev/loop0と関連付ける

sudo losetup -l # 特にlosetup -f で作成した場合、これでデバイスを確認

sudo ???? # ディスクラベルを作る

sudo mkfs.vfat -v -F32 /dev/loop0 # ファイルシステムを作る

sudo parted /dev/loop0 -s mklabel msdos -s mkpart primary fat32 0% 100% # 同時に両方作る方法

sudo mount -t ext4 /dev/loop0 /mnt/lp0 # マウント

##### デバイスファイル名

ドライブを表すデバイスファイル名は，ケーブル規格などによって変わり，１台目からa,b,cとなる。

また，そのあとにパーティション番号が続く。

注意点としてはLinux上と，Grub上でファイル名が異なる点。

|  |  |  |  |
| --- | --- | --- | --- |
| Linux上 | GRUB上 | 意味 | 備考 |
| /dev/sda | (hd0) | SCSI接続した１つ目のドライブ | sd\*となる |
| /dev/sda1 | (hd0,1) | sdaの１つ目のパーティション | sda\*となる  ※Legacyの場合は(hd0,0)となるので注意！ |
| /dev/sdb | (hd1) | SCSI接続した２つ目のドライブ |  |
| /dev/hda | ？ | IDEの接続した１つ目のドライブ | hd\*となる |
|  |  |  |  |

#### コマンド – デバイス

lshw ハードウェア全体の情報を確認する

オプション

-short 短縮した情報を表示

lscpu CPU情報を確認する

lsusb USBデバイスについての情報を表示する

cat /proc/bus/usb/devices でも良い。（Raspbian, Ubuntuには無かった）

lsblk ブロックデバイスをツリー状に一覧表示

書式：lsblk [options] *dev/sda1*

オプション

-f ファイルシステムに関する情報の表示。UUIDも含む。 (also) -fs

--output 出力する列を指定

lsblk --output NAME,UUID,FSTYPE # 列名に空白はダメらしい

blkid ブロックデバイスの詳細情報を一覧表示する

書式：sudo blkid *dev/sda1* ※ルート権限がないと何も表示されない。

ls -l /dev/disk/by-uuid　これでも似たような情報が確認できる

（関連）lsblk

ブロックデバイス：HDDやCD-ROMなどのようにブロック単位でデータを転送するデバイス

fstab ファイルシステムのチェック，および問題を修復

自動マウントも可能。

（関連）

[/etc/fstab](#_fstab_自動マウントの設定) マウント設定が記載されたファイル

/etc/mtab 現在マウントされているファイルシステムの情報が格納

losetup ループバックデバイスの作成，削除

書式：losetup [options] *loop\_dev*

losetup -f /root/disk1.img

losetup /dev/loop0 ./*disk\_image.img*

オプション

-f [*loop\_dev*] 使っていない最初のデバイスを検索。--find デバイス番号省略で、デバイス名だけ返す。

-d 対象のデバイスを切り離す。--detach

-l

fallocate

任意のサイズの空ファイルが作成される

書式：fallocate -l <ファイルサイズ> <ファイル名>

補足：このコマンドでswapファイルを作るのは非推奨。ddを使うべき。

#### lspci

全ての PCI デバイスを表示。LiSt of PCI (pc interfaces)の意。

書式 lspci [options]

結果の一例：

00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 02)

(1) (2) (3) (4)

Subsystem: Intel Corporation PRO/1000 MT Desktop Adapter

Flags: bus master, 66MHz, medium devsel, latency 64, IRQ 19　(5) (6)

I/O ports at d020 [size=8]　(7)

...

・(1)PCI識別番号　00:03.0

・(2)PCIデバイスの種類　Ethernet controller

・(3)ベンダー名（ベンダーID）　Intel Corporation

・(4)デバイス名　82540EM Gigabit Ethernet Controller (rev 02)

・(5)バスの速度 66MHz 【 -vで表示 】

・(6)IRQ番号 　 19 【 -vで表示 】

・(7)I/Oポートアドレス d020 【 -vで表示 】

オプション

-v 詳細表示

-vv 更に詳細表示

小技

lspci | grep VGA VGAを確認する

lspci | grep Ethernet ネットワークカードを確認する

lspci | grep Audio オーディオデバイスを確認する

#### mount

デバイスのマウント

書式：mount　[*target\_device\_name*]　[*mount\_point* ]　[options]

対象については，ln （リンク作成）と同じと覚える

(opp) umount ※unmount ではない事に注意

※引数なしで実行した場合，現在マウントされている**ファイルシステムの一覧を表示**する。

その場合は/etc/mtab ファイルの情報を元に表示される。

関連ファイル：

/etc/mtab カーネルが提供する全マウントデバイス

※近年では/proc/self/mountsへのシンボリックリンクになっているらしい

/proc/mounts /proc/self/mountsへのシンボリックリンク

/proc/self/mounts 現在実行中のプロセスでマウントしているデバイス

オプション：

-o コンマ区切りでマウントオプションを指定する。 --options

（参考）[etc/fstab](#_%2525252Ffstab_自動マウントの設定)

mount -t ext3 -o ro /dev/sda1 /export # 使用例：読み取り専用にする

# サブツリーを別の場所にマウント（/etc/fstabには無いオプション）

-o bind

参考：-w（-o rw相当[def.]）；-r（-o ro相当）

-a etc/fstab でautoになっているデバイスをマウントする --all

-t ファイルシステムの種類を指定 --type

umount -at xfs # xfs形式のファイルシステムを全てアンマウント

-B サブツリーを別の場所にマウント--bind

書式：mount --bind <dir1> <dir2>

特定のユーザーに特定のディレクトリにだけアクセスさせる時などに使用する。

mount --bind /var/log /root/systemlogs

（参考）bindfsコマンド

umount

デバイスのマウント

※オプションなどは、ほとんどmountと同じなので省略。特殊、独特なもののみ記載する。

findmnt マウントポイントをツリー状に表示する

### 認証系

PAM認証

PAM とは Pluggable Authentication Modules

特定のソフトウェアに依存しない、独立した認証の仕組み。

### システム系

#### コマンド

arch

現在のマシンのアーキテクチャを表示

lsinitrd

初期RAMディスク(initramfs) の中身を参照する

setenforce SELinuxの動作モードを変更する。現状の取得は getenforce

1 Enforcing 0 Permissive

dmidecode

sudo dmidecode -t baseboard # マザーボードの情報を確認する

#### 日時系

##### timedatectl

書式：timedatectl [options] subcommand

※普通ならあまり使われないコマンドだが，デュアルブートや仮想マシンなどを使うと使う事もある。

サブコマンド

list-timezones 使用できるタイムゾーンを一覧表示

# 利用できるタイムゾーンを調べる

timedatectl list-timezones | grep Asia

set-timezone *timezone\_name*

set-local-rtc RTCを使用するかどうか。1/0。（yes/no，true/falseも可能）

# デュアルブート時の時間の差を解消する

sudo timedatectl set-local-rtc 1 --adjust-system-clock

set-ntp NTPを使用するかどうか。1/0。（yes/no，true/falseも可能）

sudo timedatectl set-ntp true

status 現在の時刻と設定を表示

オプション

--adjust-system-clock RTCの設定を変更したとき、それに合わせてシステムクロックを調整する

（参考）<https://www.atmarkit.co.jp/ait/articles/1906/13/news007.html>

##### date

現在の日付を表示

##### hwclock

概要

ハードウェアクロック（RTCともいう？）の情報を出力する

sudo hwclock --verbose --systohc --localtime

オプション

--verbose ハードウェアクロックの情報を出力する

hwclock --verbose

--systohc OSの時刻（NTPで取得した時間）をハードウェアクロックに適用する

--hctosys ハードウェアクロックの時間をOSの時刻に適用する

modprobe 依存関係を考慮してカーネルモジュールのロードとアンロードを行う。

書式：modprobe [options] *modulename*

modprobe cdrom # cdromモジュール（CD-ROMドライバ）をロード

modprobe -r cdrom # アンロード

設定ファイル：/etc/modprobe.d/*myconfig*.conf

（ **/etc/modprobe.d/配下の.confファイル**であれば設定ファイルとして扱われる ）

options 各カーネルモジュールのデフォルトパラメータを指定する

alias カーネルモジュールに別名をつける

install 特定のカーネルモジュールのロード時に実行されるコマンドを指定する

remove 特定のカーネルモジュールのアンロード時に実行されるコマンドを指定する (also) -r

remove *modulename* command

blacklist ロードしたくないカーネルモジュールを指定する

設定例

alias uhci uhci-hcd

install snd-emu10k1 /sbin/modprobe --ignore-install snd-emu10k1 && /sbin/modprobe snd-emu10k1-synth

remove ov518\_decomp /sbin/modprobe -r --first-time --ignore-remove v518\_decomp

options b43 nohwcrypt=1 qos=0

blacklist dccp

efibootmgr

efibootmgr uefiブートローダの設定を管理する

オプション

-o ブートオーダーを変更する --bootorder

（参考）<https://kledgeb.blogspot.com/2015/11/efibootmgr-1-uefiuefi.html>

BIOS（Basic I/O System）

UEFI（Unified Extensible Firmware Interface）

BIOSよりも高機能で、GUIなどを用いて設定ができる。

BIOS/UEFIで出来る事

・日付と時刻

・電源管理

・起動デバイスの優先順位

・組み込みデバイスの有効/無効化

quiet 起動中のカーネルからの情報出力を抑制する

initctl　Upstartの「ジョブ」を制御する

Upstartの「ジョブ」を起動する為に必要なイベントを発行する。

（参照）[Upstart](#_Upstart)

cat /proc/cpuinfo # cpu

sudo dmidecode -t baseboard # マザーボード

#### sysctl カーネルパラメータの変更

#### init　ランレベルを変更する。

書式：init [options] [command]

※ランレベルを確認したい場合は init --help で見れる。

（関連）systemd

オプション： --help と --no-wall のみ

コマンド

0～6 ランレベルを指定

s (S) ランレベル１。レスキューモード。

q (Q) 【 SysVinit限定 】システムを再起動せずに /etc/inittab を再度読み込む

注意 init -q ではない！init q

telinit ランレベルの変更

initとの違い：

init 子孫プロセスが死ぬまで待機

telinit ランレベル変更のシグナルが送られるまで待機

telinit 6 # ランレベル 6に→再起動。つまりreboot と同じ事が起こる。

（参考）[ランレベル](#_ランレベル)

runlevel 現在および１つ前のランレベルを調べる

>N 5 # この様な結果が表示される

（関連）init

### ディスク操作（パーティション）

#### 基本情報

ディスク容量が200GB，メモリが512MBのサーバを数百人のユーザで利用する場合の一例

・/ 1GB

・swap 1GB

・/boot 100MB

・/usr 8GB

・/var 8GB

・/home およそ180GB

#### swap領域

##### 概要

別名：swap area

**仮想メモリ**の事。swapユニットによって設定される。

swap領域の推奨値は以下の通り。（Red Hat Enterprise Linux 8）

RAMが2GB未満の場合：RAMの2倍

RAMが2GB～8GBの場合：RAMと同量

RAMが8GB～64GBの場合：4GBからRAMの0.5倍

RAMが64GB以上：負荷に依存するが，少なくとも4GB

⇒ 条件が複雑だが、2~8GBだと考えておけばOK？

swapファイルを作成する方法と、swap用パーティションを用意する方法がある

|  |  |  |
| --- | --- | --- |
|  | swapファイル | 専用パーティション |
| 概要 | ファイルを作成してあるパーティションの一部をswap領域として専有する | swap領域専用のパーティションを作成する |
| 特徴 | ・後から領域の変更が可能 |  |
| 備考 | ・どちらか迷ったらこちらで作成すると良い | ・クラウドのVMなどではコスト高になる（らしい） |

補足

ファイルシステムの助けを借りず、独自で機能するように設定する必要がある。

##### swapファイル

特に後からswap領域を増やす必要がある場合に使われる？

# 現在のswap領域の設定を確認する

swapon -a

# swapファイルを作成する　以下の例では2GB

# 補足 ddでは　bs×count でファイルをコピーする

sudo dd if=/dev/zero of=/swapfile bs=1M count=2048 # /(root) に直接作成

chmod 600 /swapfile # 権限の変更

# フォーマットしてをswapを有効化

sudo mkswap /swapfile

sudo swapon /swapfile # 取り消す時はswapoff /swapfile

次回ログイン時にも有効になるように下記の設定ファイルを変更

[ /etc/fstab ]

/swapfile swap swap sw 0 0 0

補足：fallocateを使った方法を示すWebサイトがあるが、fallocateは非推奨

無事作成されたかどうかを確認する

[ terminal ]

free -mh

# 「Used」値が0である場合は、スワップ領域はまだ使用されていない。

# これでも確認可能

sudo swapon -s

・配置先はどこでも良い？

・パーティションを用意する場合とどう違う？

補足

そのため、fallocateでswapファイルを作成するのは非推奨。ddを使う。

fstabの他の設定方法

/swapfile none swap sw 0 0 0 # Ubuntuのデフォルトがこうなっていた

/swapfile swap swap sw defaults 0 0 # optionをdefaultに

##### 専用パーティション

sudo fdisk /dev/sdb # fdisk の対話モードへ入る。

# fdisk の対話式コマンドへ

# 補足：対話モードの時はCtrl+Cで対話コマンドをキャンセルできる

# パーティションを作成する

コマンド: d → n → p # 既にあるパーティションを削除してから新規作成する場合

最初ののセクタ： # これは空白（=def.)で良い（2048など）

最後のセクタ：+2G # 2Gバイト数でも指定できるらしい

# 先にパーティションタイプの一覧表示で「Linux swap」の値を確認。82とかのはず

コマンド: l

# タイプをswapに変更する

コマンド: t # 既にあるパーティションを変更する

# 確認と書き込み

コマンド: p # 編集内容の確認

コマンド: w # 上記変更の適用（対話モードが終了する）

コマンド: q # 適用せず終了する場合

# 念の為OSのコマンドでもパーティションが作成されているか確認

lsblk

# フォーマットしてをswapを有効化

mkswap /dev/sdb1

swapon /dev/sdb1

# 補足

# 「最後のセクタ」をセクタで指定する場合：例として2GB

# 補足：k→Mなどは1024を掛ける　セクタサイズは通常 512b

　(2 × 1024 × 1024 × 1024 / 512) + 2048(最初のセクタ) = 4,196,352

参考：https://kazmax.zpp.jp/linux\_beginner/mkswap.html

補足

fdisk -l /dev/sda1 | grep 'Sector size' # fdiskでセクタサイズ、ブロックサイズを確認

fdisk -l /dev/sda1 | grep 'セクタサイズ' # 日本語環境の場合

#### dphys-swapfile

設定ファイル

/etc/dphys-swapfile

#### zram

もともとはRAM 内に圧縮ブロックデバイスを作成するためのカーネル

zramctl

設定スクリプト：/etc/systemd/nvzramconfig.sh

#### fdisk

MBR形式のパーティション操作。（GPT形式ではgdiskを用いる）

書式：fdisk [options] *device\_name*

特徴：ルート権限。通常対話型で用いる。

関連：gdisk，parted

オプション

-l [*device\_name*] RAM を含め，各パーティションの情報を表示する

デバイス名省略すると，すべてのデバイスの情報が表示される。

サブコマンド

l パーティションタイプの一覧表示

（抜粋）b：FAT32，82：Linux swap，83：Linux，ef：EFI

n パーティションの作成

d パーティションの削除

p パーティションテーブルの表示 ※直接実行時の-lオプションと混同しないように！

t パーティションタイプ（システムID）の変更。続いてLを入力するとHEX値とシステムの対応表が表示される。

w パーティションテーブルの変更を保存して終了

q パーティションテーブルの変更を保存せずに終了

m サブコマンドのメニューの表示

x エキスパートモードへ

h ヘッダ数を変更。def.=

s セクタサイズを変更。def.= 512(b) ストレージで定義されている値な為、通常変更不要

なお，fdsik, gdisk, parted, gparted ではディスクをアンマウントしないとパーティション変更ができない。

gdisk

GPT形式用のfdisk

MBRにも対応しているらしいが？

#### parted

##### 概要

パーティション操作を行う。MBR形式，GPT形式のパーティションテーブルをサポート

bsd，loop (raw disk access)，gpt，mac，msdos，pc98，sun　に対応。

書式：parted *device\_name* [サブコマンド1] [サブコマンド2]…

（通常はfdiskと同じく，対話型で用いる。）

特徴：ルート権限

オプション

-l すべてのデバイスのパーティションテーブルを表示

parted -l /dev/sda

-s 対話形式に入ることなくサブコマンドを指定 (also) --script

parted /dev/sdb -s mklabel gpt

サブコマンド

[Parted対話形式のコマンド](#_Parted対話形式のコマンド)を参考

（オンラインマニュアル：英語）[https://www.gnu.org/software/parted/manual/html\_node/index.html#Top](https://www.gnu.org/software/parted/manual/html_node/index.html" \l "Top)

補足：

バージョンアップなどにより，あまり使いやすいコマンドではなくなった印象。

fdiskやgdiskの方が良いかもしれない。GUI環境なら迷わずgpartedを使う事！

GParted

parted のGUIバージョン。 UbuntuのLive USBなどでは最初から入っている。

##### Parted対話形式のコマンド

mklabel ディスクラベル（パーティションテーブルの方式）の変更。(also) mktable

※意味が分からない人は実行してはいけない。通常実行しない。

書式：mklabel *partition\_table*

sudo parted /dev/sdc mklabel gpt

ディスクラベル

mbr （デフォルト） msdosでも良いらしい。master boot record

gpt Linux標準。 macでも良いらしい。guid partition table

mkpart 新しいパーティションを作成する

書式：mkpart　[パーティションタイプ]　[ファイルシステム]　[パーティション名]　開始位置　終了位置

パーティションタイプ：primary（基本），extended（拡張），logical（論理）

ファイルシステム：ext2, fat, fat32, ntfs，linux-swap など

開始位置/1M 終了位置 s（セクタ；大体=512byte），M（メガ）や%などで指定できる。

MiB, GiB指定も可能。 -1s とすると最後の1セクタ未使用で残す　の意になる。

※どこかで相対指定が可能と見たが、+は使えないらしい。かつ、あくまで**ディスクの位置を指定**する点に注意。

mkpart primary linux-swap 20GiB 24GiB # swap領域を4GiB分作成

mkpart primary 532480s 5.92G # セクタと

sudo parted /dev/sdc -s mkpart primary 0% 100% # コマンドで一括。％指定。

対話：

(parted) mkpart

パラメータを省略すると，対話形式で決定をしていく

p パーティションテーブルの情報を表示する　(also) print

rm パーティションを削除する e.g. rm 2

resizepart パーティションサイズの変更【ver3.2より導入】

書式：resizepart　パーティション番号　 [終了位置] ※このパーティション番号を minor\_number と呼ぶ

注意点

・ドライブのアンマウントが必要。（？）

・ver4.2では対話モードでしか使えないコマンド

・相対指定ができないらしい

help *command* コマンドのヘルプが見られる。e.g. help mkpart

set boot，rootなど各種フラグを付与する。

unit 表示単位を変更

(parted) unit s # セクタサイズに変更　※-sで直接実行しても何も起きない

s：セクタ（=512b）

q 対話モードを終了する (also) quit

※partedはfdiskの様に処理を保留→wで実行は無く、１つ１つのコマンドが即適用される。注意。

～Parted対話形式のコマンド　続き　【ver 2.4以降，無くなったコマンド】

注意点：バージョン2.4以降[ check, cp, mkfs, mkpartfs, move, resize. ]コマンドは無くなった

resize パーティションのサイズの変更

書式：resize　パーティション番号　 開始位置　 終了位置

resize 1 10000 20402 # もともとの終了位置は変えずに，開始位置だけを変えるとmoveの様になる

cp パーティションの複写

move パーティションの移動

mkpartfs パーティション＋ファイルシステムの作成。

書式：mkpartfs　パーティションタイプ　ファイルシステム　 開始位置　 終了位置

（参考）partedチュートリアル：<http://nobusan.jp/computer/partition/parted/index.html>

#### コマンド：LVM関連

##### 概要

基本的な流れ

pvcreate → vgcreate → lvcreate 物理→論理→ボリュームグループ

→ 論理ボリュームにファイルシステムを作成 → ファイルシステムのマウント

補足：

・個人のコンピュータでLVMを採用するメリットはほとんどない。

⇒ LVMが介入するのでややこしい。ディスクの信頼性が下がる。　など

・ループバックデバイスを使ってLVM作成を試す事ができる。

sudo apt install vlm2 # vlm管理ツールのインストール

sudo dd=/dev/zero of=disk1.img bs=1024 count=512000 # ダミーイメージの作成(500Mb)

sudo dd=/dev/zero of=disk2.img bs=1024 count=512000 # ２つ目

sudo losetup -f ./disk1.img # ループバックデバイスの作成

sudo losetup -f ./disk2.img #

sudo losetup -l # ループバックデバイスの確認

> NAME SIZELIMIT OFFSET AUTOCLEAR RO BACK-FILE

> /dev/loop0 0 0 0 0 /root/disk1.img

> /dev/loop1 0 0 0 0 /root/disk2.img

sudo pvcreate /dev/loop0 # 物理ボリュームの作成

sudo pvcreate /dev/loop1 # ※作成したデバイスを指定する事。ここでは/loop0と/loop1

sudo pvscan # 作成した物理ボリュームの確認

vgcreate vg01 /dev/loop0 /dev/loop1 # ボリュームグループの作成

lvcreate -L 800M -n lv01 vg01 # vg01にlv01と言う論理ボリューム(800M)を作成

sudo lvdisplay # 作成した論理ボリュームの確認。 pvscanでも良い。

（以下略

（参考）<https://qiita.com/hana_shin/items/300f2ed35ea9e9fbd85a>

##### 用語：LVM

LVM；Logical Volume Manager

複数の物理ボリュームから１つの「論理ボリューム」を構成し，ハード的なパーティションの圧迫問題の対策となる仕組み。

各論理ボリュームのサイズを後から変更，単体のディスクを超えるサイズの論理ボリュームの作成，新たな物理ボリュームの追加，特定のボリュームだけをボリュームグループから削除　などが可能

なお，論理ボリューム(LV)のデバイスファイルは /dev/VG名/LV名 となる。

ジャーナリングファイルシステム

新履歴をログに記録しておく機能。ジャーナルに記録されている情報を使用することで，障害時のデータ復旧を短時間で行うことができます。ext3以降のファイルシステムに付属。

ボリュームグループ(VG; Volume Group)

物理ボリュームを集めて構成された仮想的な領域。≒RAIDなどで使う？仮想的なハードディスク。

（参考）vgcreate

物理ボリューム

PV：Physical Volume　物理的なハードディスクやパーティション

（参考）pvcreate

倫理ボリューム(LV; Logical Volume)

LVM(Logical Volume Manager) によって作成される，物理ボリューム内の仮想的なパーティションの事で，サイズを後から変更したりとかなり柔軟な管理が可能。

（参考）lvcreate

スナップショット

倫理ボリュームへの変更前の情報を保持する機能。任意のバックアップ対象を含める事で，運用を停止する事なく，安全にバックアップを行う事ができる。

##### コマンド

pvcreate 物理ボリュームの作成

vgcreate ボリュームグループの作成

lvcreate 倫理ボリュームの作成

vgdisplay ボリュームグループの詳細情報を表示

pvdisplay 物理ボリュームの詳細な情報を表示

lvmdiskscan 利用可能なブロックストレージデバイスを表示

#### ディスク関連のその他のコマンド

df ディスクの空き容量の確認　Disc Free

同時にマウントパスとデバイス名（sda1など）が確認できる

書式：df *directory\_name*

オプション

-m 単位をメガバイトにして表示する -k キロバイト単位で表示

-h 読みやすい単位で表示する

-i inodeの使用状況を表示

-T ファイルシステムを表示？？

（関連）/etc/mtab

du

ディスクの使用量を表示 Disk Usage

書式：du [options]... *file\_path*... または　du [options]... --files0-from=*file\_path*...

du -h -d 1 ./

パスの指定が無い場合は，カレントディレクトリとサブディレクトリが占めている容量が表示される。

オプション

-a ディレクトリ以外にファイルについても表示

-h 分かりやすい単位で表示 --human-readable

-S ディレクトリの容量にサブディレクトリの容量を含めずに表示 --separate-dirs

du -h

> 12K ./dir #

> 40K . # この容量は，dirの容量も含まれている。=28K+12K （下の結果より）

↓

du -hS

> 12K ./dir #

> 28K . # dirの容量を除いた，指定（カレント）の純粋な容量。

-s 指定したファイルやディレクトリの合計容量が１つだけ表示 --summarize

du -hs

> 40K . # 上の例と同じ環境でコマンドを打った結果

-c 全体の合計（-sオプションで出る結果と同じもの）も同時に表示　 --total

du -hs

> 12K ./dir #

> 40K . # dirの容量も含んだ容量

> 40K 合計 # 合計容量も表示。つまり-scは意味がない。やるなら-Sc とか。

-k キロバイト単位で表示 -m メガバイト単位

-d *num* 対象ディレクトリの深さを指定する --max-depth=*num*

dump

特にサーバーのバックアップファイルを作成する。

但し古い情報が多い。⇒ いまはあまり使われていない？20.09.22

（関連）restore

#### 用語（パーティション・ディスク管理関係）

基本(primary)パーティション

ストレージ内に作成する，ファイルシステムを作成する為の枠。

起動ドライブとして指定できる。

拡張(extended)パーティション

論理パーティションを作成する為だけのパーティション。１つのディスクに付き１つしか作成できない。

論理パーティションは数に制限なく作成できるので，５つ以上のパーティションが必要な場合などに利用。

論理パーティション

拡張パーティションに作られる特殊なパーティション。論理ドライブとも呼ばれる？

ストレージのパーティション数に関わらず，最初の論理パーティションは５番目（/dev/sda5）からとなる。

GPT；GUID Partition Table

MBRに置き換わる，新しいパーティションテーブル。

基本パーティション数128個。最大ハードディスク容量9.4ZB（ゼビバイト）Zゼビ= 1000E（エクサ）。

GPT形式のストレージを起動ディスクとする場合，ファームウェアがBIOSでは無くUEFIである必要がある。（そもそも2.2TBを超えるストレージはファームウェアがUEFIである必要がある。）

拡張パーティション，論理パーティションという考え方は無い。

fdiskの代わりにgdisk コマンドを用いる。

（関連）MBR

MBR；Master Boot Record

HDDなどのストレージの最も先頭にある，起動に必要なプログラムや情報を記録した小さな領域。 コンピュータの起動時に最初に読み込まれる。MBRの場合はパーティションを１つのストレージ（≒HDD）に最大４つ作成する事ができ，うち１つを拡張パーティションにする事ができる。（詳細）[拡張パーティション](#_拡張パーティション)

HHDの最初のセクタ(512バイト)にMBRは有り，使用領域が厳密に決められている。

**最初の446バイトがブートローダ，次の16バイトにパーティション情報が続き（第１～第４），最後の２バイトがブートシグニチャ**になっている。

具体例１）MBRの復元を行う

dd if=mbr\_backup.img of=/dev/sda bs=446 count=1 # MBRを書き換える

PBR：Partition Boot Record　との違いは？

ルートパーティション

最上位のディレクトリパスであるルートディレクトリ(/) を持つパーティション

Btrfs

バターエフエス。次世代のLinux標準ファイルシステム。（開発中）

btrfs オラクル製ファイルシステム

特徴

・マルチデバイス（複数のデバイスにまたがったファイルシステムの作成）

・サブボリューム単位でのスナップショップ機能

※サブボリューム：ディレクトリの一部をサブボリュームとして個別のファイルしスナップショット（倫理ボリュームへのバックアップ）などを取得できる。

・自動でデータを圧縮する。

LBA

ESP；EFI System Partition

WindowsなどUEFIに準拠するコンピュータで使われ，ブートローダや起動に必要なドライバが置かれており，ここの情報を基にOSが起動される。

電源が投入され，ファームウェアが読み込まれた後の起動シーケンスで最初にアクセスされる。

Journaling file system

書き換え処理要求（トランザクション）単位で内包するファイル構成情報を含むメタデータを管理および保持できる機能を持ったファイルシステム。ログの記録などが可能。Linuxにおいてはext2のみが非対象。

##### MBR：拡張パーティション

基本パーティションは４個までしか作成できず，それ以上のパーティションが欲しい場合は基本パーティションの内の１つを拡張パーティションとして割り当て，その中に倫理パーティションを作成する。

また，これは旧来のMBRディスクでのみ使用されるパーティションでGPTでは使用されない。

fdisk では e を選ぶとextedned（拡張）パーティションとなる。

### ファイルシステム

#### 概要

各パーティションにはファイルシステムが必要。

Linux用ファイルシステム一覧

|  |  |
| --- | --- |
|  | 備考 |
| ext2 | 以前のLinuxで標準的に使用されていた規格。inodeの数が制限される。 |
| ext3 | ex2の後継（root可能）inodeの数が制限される。  ジャーナリングファイルシステム |
| ext4 | ex3の後継（root可能）inodeの数が制限される。  ジャーナリングファイルシステム |
| XFS | ジャーナリングファイルシステム（root可能）  動的inode　　　　IRIX 5.3において初めて搭載。ほとんどのLinuxで使用可能。  swap領域でもちいられる？ |
| JFS | IBMが開発  ジャーナリングファイルシステム  動的inode |
| exFAT | マイクロソフトが開発。FAT16，FAT32との互換性は無い。  フラッシュドライブ向けのファイルシステム。倫理上では16EiBまで可能。  Linuxカーネルではデフォルトではサポートしていないが，使用する事自体は可能。  Win/Linux/mac間で利用できるため，外付けフラッシュドライブ向き。 |
| Btrfs | 次世代のLinux標準ファイルシステムとなるべく開発中。 |
| aufs | 複数のディレクトリを透過的に重ねることができる技術で、仮想ファイルシステムに向く。過去にdockerで長い使われてきた実績がある。 |

・ext2/ext3/ext4ファイルシステムを作成した時，ファイルシステム全体の**5%**がrootユーザ用の領域として予約される。

#### 関連コマンド

コマンド概要

|  |  |  |
| --- | --- | --- |
| コマンド | 概要 | 補足 |
| tune2fs | ext2/ext3のパラメータを設定 | ext4 も一応？ |
| resize2fs | ファイルシステムのサイズを変更する |  |
| mkfs | ファイルシステムを作成 | デフォルトではext2  -t オプションでシステムを選択 |
| mke2fs | ext2/ext3ファイルシステムの作成 | デフォルトではext2  -t, 又は -j などでシステムを選択 |
| fsck | ファイルシステムの整合性を確認 |  |
| e2fsck | ext2/ext3ファイルシステムの整合性を確認 | ext4 も一応？ |
| dumpe2fs | ext2/ext3システムをダンプ |  |
| debugfs | ファイルシステムの調査、デバッグ |  |

xfs\_info XFSファイルシステムの情報を表示

xfs\_db XFSファイルシステムのデバッグを行う

xfs\_check XFSファイルシステムのチェック

xfs\_fsck ではない点に注意（fsckやe2fsckというコマンドがある）

xfs\_fsr XFSファイルシステムのデフラグを行う XFS **F**ile**S**ystem **R**eorganizer

（例）xfs\_fsr /dev/sda1

xfs\_repair XFSファイルシステムの検査・修復を行う

（例）xfs\_repair /dev/sda1

※xfs\_mkfs は存在しない。mkfs.xfsで作成を行う。

##### tune2fs

ext2/ext3ファイルシステムのパラメータを設定

このコマンドで**ext4に変更する事はできない**。mkfs などを使うしかない。設定の変更はできるらしい。

オプション

-c *f* 次にファイルシステムチェックを行うまでの最大マウント回数を指定（**c**ount）

tune2fs -c 10 /dev/sda2 # ファイルシステムをチェックせずマウントできる回数を10回にする

-i *time* [**d**/m/w] ファイルシステムのチェックを行うまでの最大時間間隔を指定。（**i**nterval）

tune2fs -i 2w /dev/sda2 # 数字のみだとデフォルトである日付(d)が指定される

-j ext2をext3ファイルシステムに変換。（**j**ournaling systemはext3から対応）

-J *jaournal-options* ジャーナルパラメータを変更。**ext4への変更ではない**

-L *label\_name* ファイルシステムのボリュームラベルを設定 (also) LABEL= *label\_name*

tune2fs -L /dev/sda1 filesystem #

（関連）tunefs

※mkfs, mke2fs と違って-t オプションは無い。

hdparm ハードディスクのパラメータを確認、設定

sdparm

sudo sdparm /dev/sdb #

badblocks デバイスの不良セクタを検索するツール

sudo badblocks -v -s /dev/sdb1 | tee /tmp/badblocks.txt

オプション

-s スキャンの進捗を表示します。

-v 詳細な表示を行います。

-w 書き込みモードでテストします。

-o *file\_name* 指定されたファイルに不良ブロックのリストを書き出します。

fsck ファイルシステムの整合性を確認 **f**ile **s**ystem **ch**ecker

注意点：対象のファイルシステムはアンマウントする必用がある

書式：fsck [options] [*device\_name*]

オプション

-t *file\_system* 　ファイルシステムの種類を指定

-A /etc/fstab に記述されている全ファイルシステムに対して実行 w**A**lk through

-N 実際には実行せず，実行する内容を表示 do **N**ot execute

-a 問題を自動的に修復 **a**utomatically repair

-r 問題を対話的に修復 **r**epair

（関連）

/etc/fstab マウント設定が記載されたファイル

/etc/mtab 現在マウントされているファイルシステムの情報が格納

e2fsck ext2/ext3ファイルシステムの整合性を確認

オプション

-f forceファイルシステム状態フラッグがcleanのときも強制的にPass1~5のチェックを行う。

-p preen軽度なエラーは自動修正

-y 全ての問い合わせに対し，自動的にyes

-n 全ての問い合わせに対し，自動的にno

-b superblock 指定したスーパーブロックバックアップを用いる

-v 詳細を表示

（関連）fsck

具体例

e2fsck -f -y -v -C 0 ‘/dev/sda2’ # GPartedより

e2label

rootfs-expand ルートファイルシステムの自動拡張。

CentOSでのみ使うコマンドらしい。resize2fsとほぼ同等のコマンド？

##### mkfs

mkfs.xfs，mke2fsなどのフロントエンドコマンド。（実際にはmkfs.ext2などが実行される）

※mkefsでは無い点に注意（mke2fsと言うコマンドがある）

書式：mkfs [options] *device\_name*

※-t指定なしだとext2システムが作成される

オプション（共通）

-t *file\_system* ファイルシステムを指定。省略するとext2になる。

-c ファイルシステムの整合性を検査（けっこう時間がかかる）

-v 詳細な進捗を表示

-l 不良ブロックのリストを filename から読み込む。

-L ラベルを指定する

-j ext3ファイルシステムの作成

関連コマンド

mkfs.xfs XFSファイルシステムを作成する

mkfs.ext4 ext4ファイルシステムを作成する

mkfs.fat FATファイルシステムを作成する。

mkfs.vfat FAT/FAT32ファイルシステムを作成する。

オプション

-F [*num*] FATの数字を指定。このオプションを省略をするとFAT16になる。

sudo mkfs.vfat -v -F32 ‘/dev/sdb1’ # FAT32にする

（参考）ファイル形式別：<https://z1000s.hatenablog.com/entry/2017/05/31/134333>

（補足）ext2/ext3/ext4ファイルシステムを作成すると，全体の**5%** がrootユーザー用に割り当てられる。

mke2fs ext2ファイルシステムを新規に作成　**m**a**ke** ext**2** **f**ile **s**ystem

*片方のコマンドはmkfsなのに，mk****e****2fsなので注意！統一しろ！*

倫理ボリューム内にファイルシステムを作成

書式：mke2fs [options] *device\_name*

* -t 指定なしだとext2システムが作成される
* FAT系はこれでは作成できない。mkfs.vfat を用いる

オプション

-j　 ext3ファイルシステムを新規に作成（**j**ournaling systemはext3から対応）

-t　*file\_system* 新規に作成するファイルシステムの種類を指定

mke2fs -t ext3 /dev/sda2

-c ファイルシステムを作成する前に不良ブロックを検査

（参考）mkfs , e2fsck

～mkfs 続き

具体例：

# gparted で実行されるコマンド

sudo mkfs.fat -F32 -v -l ’/dev/sdb1’ # sdb1 にfat32 のファイルシステムを作成 ⇒ -lなしなら成功

# 以下はターミナルでは未確認 20.10.06

##### resize2fs

ファイルシステムのサイズを拡張する。（アンマウントなしで可能）

事前にe2fsckで整合性を確認しておく必要がある。

書式：resize2fs [options] *device\_name* [size]

size指定がないとパーティションの最大サイズまで拡張される。

size はs（セクタ）, M（メガバイト）, K（キロバイト）, M（メガ）, G（ギガ）,が使えるが、小数点が使えない点に注意する

オプション

-p （小文字）作業完了分のパーセントを表すバーを表示

-P （大文字）ファイルシステムの大きさを見積もる

mkswap swap領域を作成する

fdiskでswapパーティションを作成 → mkswapでswap領域を作成 → swaponで有効化。

（関連）swapon

swapon (see also) mkswap：swap領域の作成

sudo swapon /swapfile

swapoff swap領域を無効にする。

sudo swapoff /swapfile

mkfs.xfs XFSファイルシステムを作成する

mkfs.fat

#### その他のファイルシステム

##### Lustre

大規模のクラスターコンピューティング・スーパーコンピュータで使用されている分散ファイルシステム

公式：<https://www.lustre.org/>

### アカウント系

#### まとめ

アカウント情報の確認コマンドのまとめ

|  |  |
| --- | --- |
| コマンド | 機能 |
| id | 自身の名前とUID，所属グループとGID |
| groups | 所属グループのみ。GIDは出ない。 |
| whoami | ログインしているユーザー**自身の名前**を表示 |
| who | ログインユーザーの**一覧**を表示。ログイン時刻も表示。 |
| w | ログインユーザーと，実行中の**プロセス**を表示 |
| last | 最近ログインしたユーザーの一覧をリスト表示する |

・プライマリ（基本）グループ ... デフォルトのグループ

・セカンダリ（サブ）グループ ... プライマリグループのほかに所属しているグループ

#### コマンド：アカウント操作

usermod ユーザーの設定を変更

-a 補助グループを追加する --append

-d ホームディレクトリを再設定 --home

-u UID(User ID) の変更。 --uid

-g GID(Group ID) の変更。プライマリグループを変更。 --gid

-l *new\_name* ログイン名の変更。e.g. -l newpi pi --login

-G 補助グループを変更する　e.g. usermod -G project1,project2 pi --groups

useradd 新規ユーザーを作成する

書式：useradd *user\_name*

オプション

-b *base\_dir* ホームディレクトリのベースパスを指定 --base-dir

-D ユーザー作成時のデフォルトの設定を表示 --defaults

（補足）害のないオプションなので打ってみると良い

-g プライマリグループを指定 (also) -G セカンダリグループを指定 --gid

-m ユーザーのホームディレクトリが存在しない場合，作成する --create-home

-M ホームディレクトリを作らない --no-create-home

（補足）追加するユーザーの所属グループに，現在存在しないものを入力すると，/etc/groupファイルも更新される。

（補足）<https://ja.wikipedia.org/wiki/ユーザー識別子>

adduser 新規ユーザーを作成する

Debian系の場合，対話形式でpassword などの設定を行う。（RedHat系ではuseraddと同じ挙動）

userdel ユーザー削除

書式：userdel [options] *user\_name*

オプション

-r 対象ユーザーのhomeとメールを削除する (also) --remove

補足）ユーザーを失ったファイルなどの所有者はUID（1000など）に置き換わる。

groupmod グループ名変更

オプション

-n グループ名の変更 -n new-name old-name

-d ユーザーをグループのメンバーから外す

passwd ユーザーのパスワード変更

書式：passwd [*user\_name*]

実行ファイル：/usr/bin/passwd

対象ファイル：/etc/shadow

オプション

-l 指定ユーザーをロック（ログインできないように）する【ルート権限】

-u ロックを解除する【ルート権限】 ラージUは存在しない

gpasswd　 グループのパスワードを変更

-a ユーザーをグループに所属させる

sudo gpasswd -a tmp sudo # tmpユーザをsudoグループに追加。管理者権限をもつのとは意味が異なる

#### コマンド：アカウント情報確認

id 自身の名前とUID，所属グループ名とGIDの表示

書式：id [*username* ] [options]

-u ユーザーIDのみ表示

-n ユーザーやグループ名だけを表示（-u, -g, -Gと共に使用）

groups 自身の所属グループ名のみ。GIDは表示されない。

書式：groups [*username* ]

whoami ログインしているユーザー**自身の名前**を表示

who ログインしているユーザーの**一覧**を表示

root pts/0 2019-10-14 11:29 (192.168.56.1)

# ユーザー名，端末，ログイン時刻が表示される

w システムにログインしているユーザーと，実行中の**プロセス**を表示

USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT

root pts/0 192.168.56.1 Mon11 29:43m 0.37s 0.10s ss centos72

# ユーザー名，端末，ログインしたホスト，ログイン時刻，？，？，？，実行中のプロセス

last 最近ログインしたユーザーの一覧をリスト表示する

/var/log/wtmp　にログが残っており，それを表示している。

-数値 数値で指定した行数分の最近のログイン情報を表示する

last -5 # 過去５行分のログイン情報を表示

> root pts/0 192.168.56.1 Mon Oct 14 11:29 still looged in

> root pts/5 192.168.56.103 Mon Oct 10 20:17 - 14:04 (4+17:47)

# ユーザー名，端末，ログインしたホスト，ログイン時刻，ログアウト時刻

### プロセス

#### 概要

最初に起動されるinit（またはsystemd）にPID 1が割り当てられ，以降は2,3.. と割り当てられる。

上限まで使い切ると，再び数の小さい方から使用する。

#### プロセスとジョブの違い

プロセス

・プログラムの実行単位

・プロセスIDと言う一意のIDで管理する。

・実行ユーザーが設定されており，

ジョブ

・シェルを実行する作業単位

・複数のプログラム（コマンド）をまとめたものなので，複数のプロセスをまとめたもの，とも言える。

#### コマンド

top システムモニタの様にCPU負荷の高いプロセス順にプロセスを表示。5秒更新。

プロセス数，CPU使用時間，メモリ量，swap領域　などが上段に表示される。

その下にCPU使用率の高い順にプロセスが表示される。

操作

Shift+m 目盛り使用率の多い順に並び替える

q topコマンドを終了

uptime システムの稼働時間、平均負荷などを調べる

現在時間，up [稼働日数，] 稼働時間，接続ユーザー　負荷平均は直近1分，直近5分，直近15分

※topコマンドの１行目に表示される内容と全く同じ

uptime of 100 hoursみたいな意味合らしい。

free メモリやswap領域の使用状況を表示

topコマンドのヘッダ部分に表示されている情報とほぼ同じ

書式：free [options]

オプション

-h 人間に読みやすい単位で表示する --human

「memuse」や「mem」というコマンドは無い。

参考

・メモリ総量、使用量を知るには以下のような方法もある

cat /proc/meminfo | grep Mem

関連コマンド：df

#### nice

実行優先度を指定してプロセスを実行

まず，優先番号はNiceness value （nice値）と呼ばれている。

nice 値は「-20」が最優先で「19」が最も優先度が低い。-1～-20までの負の値は，rootのみが設定可能。

※注意点としては標準のnice値は0であるのに，パラメータ省略してniceコマンドを実行すると，nice値は10になる点。

書式：nice [-数値] [options] [command…]

nice -n -20 sh test.sh # test.sh を最優先で実行する

-数字 指定したnice値にする。**負の値の時は --20 などとなり，紛らわしい。**

後述のrenice では数値そのまま　つまり-20の場合 renice -20 となる。

内部ではnice値は0～39で扱われているらしい。nice値 = (-1×内部値 ) + 20 ???

CPUの待ち時間と考えると優先度が分かりやすいかも

renice 既に起動しているプロセスのnice値を変更

書式：renice [-n] nice値 [*PID*] [options]

数字 指定したnice値にする。

niceの時と異なり，負の値の時は **-20となる，紛らわしい。**

数字２つの場合は一つ目がnice値，二つ目がPIDとなる。

オプション

-n nice値の指定。省略可能。nice値の指定は**必ず第１オプションになる。**

# renice -p 500 -n -20 # これはエラーになる

renice -n 20 -p 500 # これはOK

-u *user\_name* 指定したユーザーが所有する全てのプロセスのnice値を変更

-p *PID* 指定したPIDをもつプロセスのnice値を変更。

renice -20 -p 500 # renice -20 500 と全く同じこと

#### ps

現在実行中のプロセスとそのPIDの確認 **P**rocess **S**tatus

ps -ef # UNIXオプション版

ps aux | tail -n 10 # よく用いられるオプション。BSDオプション版。（パイプ以降は任意）

オプション

※「-」を付けないBSDオプションと，「-」で始まるUNIXオプションがある。まぎらわしい。

BSDオプションは１単語の頭文字が多い。

a 他のユーザーのプロセスも表示　all

よくxと併用する。ax = -e 全てのプロセスを表示。

x 制御端末の無いデーモン等のプロセスも表示

f プロセス の親子関係をツリーで表示　forestの意。←これめちゃ紛らわしい

l ロングフォーマット。親プロセスのPID，nice値，プロセスの状態なども表示。

u プロセスの実行ユーザー名も表示　user

ps uf -C プロセス名 #

UNIXオプションは2単語やcompoundな単語

-e 全てのプロセスを表示。every processの意。

-f 完全なフォーマットでプロセスを表示　full-formatの意。

なお表示される，PPIDとはparent pidの事。親プロセスのPID。

よく-efの形で使われる。

-p 指定したPIDのプロセス情報を表示 (also) p *PID*

-l 親プロセスのPIDやnice値も併せて表示(also) l long-format の意。BSDのlと同じ。

（関連）psgrep，[killコマンドで使うシグナル名](#_killコマンドで使うシグナル名)

ps aux（上）とps -ef（下）の比較

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

root 221 0.0 0.4 95236 16320 ? S<s 9月27 0:00 /lib/systemd/

UID PID PPID C STIME TTY TIME CMD

root 221 1 0 9月27 ? 00:00:00 /lib/systemd/systemd-journald

STAT：R+,Ss+,Ss,S,の順番？

pgrep　特定のプロセス名の**プロセスID**を検索。

書式：pgrep [options] 検索パターン（プロセス名など）

オプション

-l 詳細リストの表示

-u 実行(effective)ユーザーで検索。指定方法はユーザー名またはuid (also) --euid

pgrep -u test, hoge # testまたはhogeユーザーの権限で動作しているプロセスを表示

pgrep -u 0 # root 権限で動作しているプロセスを表示

-U 実ユーザーIDで検索。指定方法は-uに同じ　(also) --uid

pstree 動作中のプロセスの親子関係をツリー状で表示する

≒ps axf

（参考）

#### kill

PIDの一致するプロセスを**１つだけ**終了させる

書式：kill [-*signal\_name*/-*signal\_number*]　PID | jobspec

PIDはps -a などで確認できる。

注意点

-20などハイフン付きはシグナル番号とみなされる。ハイフン無しはPIDとみなされる。

signal\_nameなどを指定しなかった場合は「クリーンナップして終了(15)」が送られる。（下記参考）

signal\_nameは２種類ある（e.g. TERMとSIGTERM）番号も含めると指定方法が3種類ある。

また，指定しなかった場合は -SIGTERM(15) が送られる。

オプション

-s シグナルの指定 kill -s 15 と kill -15 は全く同じ。

具体例

kill 3382 # PID3382のプロセスを終了させる。シグナル指定が無いのでTERM(15)が送られる。

killall 名前が一致するプロセスを**全て**終了させる

PIDは全て一意だが，１つのプロセス名で複数のPIDを持つプロセスを動かす事は十分ありうる。

そういったプロセス群をすべて終了させる事ができる。

書式：killall [-*signal\_name*/-*signal\_number*] *process\_name*

sudo killall -HUP httpd # httpd の全てのプロセスをハングアップする

killall -SIGTERM command # 「command」を実行している全てのプロセスを終了する

pkill **正規表現**に一致したプロセスを**全て**終了させる　signals to **p**rocess

書式：pkill [-*signal\_name*/*signal\_number*] *process\_name/command*

pkillでもシグナルを指定しなかった場合は -SIGTERM(15) が送られる。

オプション

--signal [*signal\_name*/*signal\_number* ]

##### killコマンドで使うシグナル名

これは kill -l でもみられる。

|  |  |  |  |
| --- | --- | --- | --- |
| シグナル名 | 番号 | 動作 | 詳細 |
| HUP(SIGHUP) | 1 | ハングアップ（応答なし） | 端末の切断時にプロセスを終了 |
| INT(SIGINT) | 2 | 割込みによる終了 | Ctrl+Cと同様 Cancel？ |
| QUIT(SIGQUIT) | 3 |  | Ctrl+\と同様 |
| KILL(SIGKILL) | **9** | 強制終了 | 9（急）に終わり(kill) にする |
| TERM(SIGTERM) | **15** | クリーンアップして終了  （デフォルト） | 指定なしの場合これが選ばれる  termネーターの掃除(clean up)⇒人1殺5し |
| CONT(SIGCONT) | 18 | 一時停止のプロセス再開 | （環境依存） |
| STOP(SIGSTOP) | 19 | 一時停止 | （環境依存） |
| TSTP(SIGTSTP) | 20 | 端末から一時停止 | （環境依存）　Ctrl+Zと同様  Terminal SToP？ |

[https://ja.wikipedia.org/wiki/%E3%82%B7%E3%82%B0%E3%83%8A%E3%83%AB\_(Unix)](https://ja.wikipedia.org/wiki/シグナル_(Unix))

例）

ping 192.168.1.1

ps x | grep 'ping' # 別のTerminalで実行

> 6781 pts/0 S+ 0:00 ping 192.168.1.1

kill -s TSTP 6781 # もしくはpingを実行したTerminalでCtrl+Z

kill -s CONT 6781 # 再開される。　※この後Ctrl+Zなどで停止できなくなるのは何故？

kill -s TERM 6781 # 再開される

STATで表示される主な記号

|  |  |
| --- | --- |
| 記　　号 | 内　　　　　　容 |
| D | 割り込み不可のスリープ状態 |
| R | 実行可能 |
| S | スリープ中 |
| T | 停止中またはトレース中 |
| W | swapアウトしている |
| Z | ゾンビプロセス |
| N | nice値を持つ |

（関連）[ps](#_ps)

### ジョブ関連

ジョブ：Upstartを用いたシステムで，コマンドやプログラムをグループ化して管理する為の単位。

つまりUpstart専用のコマンドと言う事？？20.10.02

次ページに続く

ジョブ　つづき

jobs バックグラウンド，一時停止中のジョブを表示

書式：jobs [options] [jobspec]

書式：jobs -x コマンド [args]

ジョブ番号 状態 コマンド と表示される

sleep 60 # 時間のかかる事が分かっている処理があるとする

> Ctrl + Z を押す → ジョブを停止

jobs # 停止中のjobとして、sleep60が表示される ※最後に&が付いていればバックグラウンド

sleep 60 & # 初めからバックグラウンドで動作させるには、コマンドの最後に「&」を付ける

オプション

-s 停止中のジョブだけを表示する

-l PID（Process ID）を含めて表示 (compare) -p PIDのみが表示される

jobs -l # PID含めて表示。PIDが14470 だったとする

kill -CONT *14470* # PIDが分かればkill コマンドが使える。（この例では再開）

bg 一時停止中のジョブをバックグラウンド動作に切り替える

書式：bg [ジョブ番号]

bg 1 #

fg バックグラウンドで動作中，または一時停止中のジョブをフォアグラウンドに切り替える。

書式：fg [ジョブ番号] 番号指定がない場合，カレントジョブ(※)がフォアグラウンドに変更

※1直前にバックグラウンドで実行されたジョブ，または直前に一時停止されたジョブ。

fg 1 # 停止中のジョブなら再開、バックグラウンドならフォアグラウンドに移る

（参考）[kill](#_kill), jobs

nohup

Terminalを終了したり，ログアウトしても処理を続ける。≒バックグラウンド処理に切り替える。

書式：nohup *commands* [parameters]

nohup ping localhost > ping.txt & # 最後の & はバックグラウンドで処理を行う事を示す。

### ログ

#### syslog

概要

ログの確認方法

関連する設定

rsyslog.confで設定

関連するコマンド

#### journald

概要

systemd 環境におけるログ管理。

ログの確認方法

関連する設定

/etc/systemd/journal.conf

volatile /run/log/journal に記録する

persistent /var/log/journal に記録する

auto[def.] /var/log/journal が存在すればそちらに，なければ /run/log/journalに記録する

none ログを記録せずに破棄する(syslog等への転送は実施する)

関連するコマンド

journalctl

### その他

wall wall message の送信；同セグメントの全端末に向けたブロードキャスト電文。

書式：wall [options] [<file>|<message>]

hostname

ホスト名を管理。少し古いコマンド。

hostnamectl

ホスト名を管理。Ubuntu 16.04以降で有効。

（関連）/etc/hosts、/etc/hostname

## ネットワーク系

### DNSのリソースレコード

DNSサーバーは，IPアドレスとドメイン名とを紐づけるためのデータベースを持っています。データ一つ一つを「リソースレコード」といい，リソースレコードは「ゾーンファイル」と呼ばれるファイルへ保存されています。

|  |  |
| --- | --- |
| SOA  Start of Authority | ゾーンに関する情報  ※ゾーン：１つのDNSサーバーが管理しているドメインの範囲 |
| NS  Name Server | ゾーンの情報を管理するDNSサーバー名 |
| MX  Mail Exchange | ゾーンで管理するドメイン宛のメールサーバー |
| A  Addres | FQDN（名前）に対応するアドレス |
| CNAME  Canonical Name | FQDN（名前）の別名 |
| PTR  Pointer | IPアドレスに対応するFQDN（名前）  in-addr.arpa 形式 |
|  |  |

### コマンド：ネットワーク系

arp-scan IP/MACアドレスの一覧。win でいうarp -a。

但しインストールが必要　（sudo apt install arp-scan）

arp-scan -I インターフェイス名 -l　# e.g. sudo arp-scan -I eth0 -l

arp

オプション

-d arpエントリの削除

arp –d 192.168.1.1

-s arpエントリの追加

arp –s 192.168.1.1 00:40:8c:18:10:00 temp

-n arpキャッシュの表示

arp –n # 全体のarpキャッシュの表示

arp –n –i eth0 # 特定のインターフェイスに関するarpキャッシュの表示

cloud-init クラウドインスタンスの構成やカスタマイズを行う。

ifup

ネットワークのインターフェイスをOnにする。

(opp) ifdown

オプション

-v ：詳細表示

iwconfig

無線LANインターフェース設定

iwconfig wlan0 power off # PowerManagementをoffにする

ping

ICMPプロトコルを用いて、相手機器の通信状態を確認する

オプション

-4 IPv4を使用

-6 IPv6を使用

traceroute

対象までの経路確認

関連：tcptraceroute、Winの tracert

ss ソケットに関する情報を表示。

どのポートが開いているか，などを確認できる。sockets statisticsの意らしい。旧netstat

書式：ss [options] [フィルター]

オプション

-n --numeric

-r --resolv

-l

-t

-u

-nr ルーティング確認

host DNSの正引き・逆引きを行う

host ping-t.com # 正引きの例

host 18.178.121.6 # 逆引きの例

※dns というコマンドは無い。

nslookup DNSの正引き・逆引きを行う

nslookup www.yahoo.co.jp

dig 詳細な名前解決を行う

### コマンド：SSH系

#### ssh

OpenSSHを利用して遠隔地のコンピューターのコンソールへログインする。

書式：ssh *user*@*ip\_addr*

書式：ssh *ssh\_config\_name*

オプション

-i 秘密鍵の指定。省略すると，~/.ssh/id\_rsa などが選択される。

ssh user@192.168.1.100 -i ~/ssh/keys/id\_rsa # -i オプションではフルパスを指定する。

リモートのファイル

dd if=~/local\_data | ssh remotehost dd of =/tmp/remote\_data # ローカルからリモート

ssh remotehost dd if=/tmp/remote\_data | dd of=~/local\_data # リモートからローカル

ssh-add 鍵の登録？

#### ssh-keygen

sshの公開鍵，秘密鍵のセットを作成する

書式：ssh-keygen

オプション

-b ビット数を指定

-C

-f 鍵ファイルの指定。

-t 暗号タイプを選択。rsa/ecdsaなど。

-N 新しいパスフレーズを指定

-R 指定されたホストの鍵をknown\_hostsから抹消

configファイル

Linux：[~/.ssh/config] windows：[C:\ Users\tuser\_name\.ssh\config ]

Host test

HostName srv\_url\_or\_ip # リモートの場所。URLやIPアドレスなど

User srv\_user\_name # リモートOSでのユーザー名

IdentityFile ~/.ssh/test/gehogeho\_a.key # 任意：パスワード認証も可能

Port 2222 # 任意：

IdentitiesOnly yes # 任意：yes or no[def.]

[ terminal ]

Host test # これで接続できるようになる

参考：https://nxmnpg.lemoda.net/ja/5/ssh\_config

ssh でGUIが有効なOSにログインする場合、クライアント側でサーバー側のGUIを扱うことはできない。

以下の方法の内いずれかを選択する事で対応可能

なお、これをX11 Forwarding と呼ぶらしい。

・~/.ssh/configを編集

ForwardX11 yes

・sshd\_configを編集

X11Forwarding yes X11転送の許可。（default：yes）

X11DisplayOffset 10 X11が転送するディスプレイ番号。（default：10）

X11UseLocalhost no X11 をlocalhost のみ許可。（default：yes）

・bash で実行時にオプションを指定

ssh usi@192.168.11.94 -X # -X オプションを指定

#### wget

Web経由でファイルをダウンロード

書式：wget [options] url1　[url2]…

オプション

-O *filename* 複数のテキストデータを一つに連結して書き込む。 --output-document=

-q 処理結果の表示を行わない。--quiet

### curl

様々なプロトコル，特にhtmlメソッドでファイルをダウンロード。

curl -X GET 127.0.0.1:8080/gyoumu/api/personnel/get/?id\_number=13093

オプション

-o ファイル名を指定 --remote-name

-O [ファイル名] 転送元と同じ名前で保存 --output ファイル名

-i HTTP ヘッダを出力に含めます。--include

-I HTTP ヘッダのみを取得します。--head

-d POSTデータの指定　--data

curl -X POST localhost:3000/items -d '{"name": "hoge", "price": 200}'

-X コマンド 使用するコマンドの明示　--request コマンド

-H 任意のヘッダーを送信する　--header

-f

-S

-L

-F http postリクエストを送信する。--form

-F ‘passwd=@/etc/passwd;type=text/plain’

補足：類似ツールにPostmanというのがあるが、そちらはアプリケーション。（Win/Mac/Linux）

### ftp

put ftpサーバーからファイルを取得

get ftpサーバーにファイルを転送

### sftp

オプション

-b バッチファイルの指定

-B 1 つの要求の最大バッファサイズを設定します。有効な値は 1024 ～ 32768 。

-i 鍵認証の場合の鍵ファイルを指定

### scp

sshを用いたファイル転送

書式：scp [options] 転送元　転送先

scp *user\_name*@ホスト名:*sorce\_path*　*target\_path* # リモートto ローカル

scp *user\_name*@*HOST\_A*:*sorce\_path*　*user\_name*@*HOST\_B*:*target\_path* # リモート to リモート

scp *usr1*@192.168.1.1:/home/*usr1*/*file.txt* ./*copied.txt* # 例：リモートtoローカル

scp ./*file.txt* *usr1*@192.168.1.1: # 例：ローカル to リモートのhome　「：」を忘れない！

オプション

-i identity\_file？

-P ssh のポートを指定

-r 下位ディレクトリも再帰的にコピー。（単純にディレクトリを転送するときに使う）

オーディオ・ビジュアル

### ip

ip route show

ルーティングテーブルの状態表示

ゲートウェイの確認もできる

sudo ip route add 192.168.1.0/24 via 192.168.1.254 dev eth0

sudo ip route delete 192.168.1.0/24

ip route show

補足

routeという古いコマンドが存在する。出力文字列のフォーマットが異なる。

ip addr show 現在のIPアドレスなど，ネットワーク設定の確認。

旧 ifconfig

Winで言うipconfig

オプション：-a

ip addr show enp0s3 # ネットワークインターフェース enp0s3 に割り当てられたIPアドレスを確認

ip addr show

（参考資料）<https://engineers-life.com/linux_command/network_linux/linux_ip-route/>

### route

書式：route [option] [target]

オプション無しで，現在のルートテーブルが表示される

インストール

sudo apt install net-tools # ほぼ間違いなく、最初から入っているが

補足

比較的古いコマンド？ip route というのが新しいコマンドらしい。（出力結果が異なる）

日本語環境だと「受信元サイト」と表示されるが，これはDestinationの事。

defaultはデフォルトルートの意らしいE

オプション

metric 数値 メトリック値を指定する

gw gw\_addr ゲートウェイを指定する

サブコマンド

#route add 宛先IPアドレス gw ゲートウェイのIPアドレス インタフェース

route add -net 172.31.0.0 gw 10.13.0.145 netmask 255.255.255.0 eth0　# -netが必要

route add 172.31.0.0 gw 10.13.0.145 eth0 # デフォルトのmaskは255.255.255.255なので注意

route delete あて先IPアドレス netmask サブネット・マスク [gw ゲートウエイIPアドレス] LANインタフェース名

DNS名でないときは -net が必要？ 2021-10-08

### mtr

My Trace Rout の略らしい。

## プログラム系

### gcc

小さなサイズのプログラムのコンパイラ。GNU Compiler Collection

※近年，直接使う事は少ない。makefileの中で使われる。

書式：gcc [options] *source\_file.c*…

gcc main.c hello.c -o main # 複数ファイルをコンパイルする場合

オプション

-g デバッグ情報を付加する。dbx,gdbなどのデバッガ を使用するときに必要。--debug

-o *filename* 指定したファイル名で出力を行う。省略すると a.outというファイルが作成される。

-c コンパイルのみを行う。filename.o のみが作成される。（ニーモニックみたいな状態？）

-L *linker* ライブラリ探査の対象ファイルを追加する

-Wall コンパイル時に警告を出す。（通常はエラーメッセージしか出ない）

拡張子

.c （小文字）C言語。プリプロセッサ、コンパイラ、アセンブラにかけられる。

.cxx C++。プリプロセッサ、コンパイラ、アセンブラにかけられる。.C（大文字），.ccでも良い。

.m Objective-C 言語ソース。プリプロセッサ、コンパイラ、アセンブラにかけられる。

.i プリプロセッサにかけられたC言語ソース。コンパイラ、アセンブラにかけられる。

.ii プリプロセッサにかけられたC++言語ソース。コンパイラ、アセンブラにかけられる。

.s アセンブリ言語ソース。アセンブラにかけられる。

.S アセンブリ言語ソース。プリプロセッサ、アセンブラにかけられる。

.h プリプロセッサファイル。通常はコマンドラインには現れない。

g++ C++用のコンパイラ。デフォルトでリンクするライブラリなどに違いがある。

### make

プログラムのコンパイルなどを行う。

書式：make [options] *target\_file*…

make install アプリをインストールする。開発時には使わない。

make -j4 CPUを並列で使う？

詳細は別ドキュメント　C／C++.docx に記載

### ldconfig

動的リンカによる実行時の結合関係を設定する

### python

pip

　pip list インストールされているパッケージの表示

参考)

arp-scan インストールが必要　（sudo apt install arp-scan）win でいうarp -a

arp-scan -I インターフェイス名 -l　# e.g. sudo arp-scan -I eth0 -l

## オーディオ・ビジュアル

設定ファイル/proc/asound/modules

alsa共通オプション

-c チャンネル数の指定

-f 周波数の指定

-f cd (16 ビット、リトルエンディアン、44100、ステレオ)

-f cdr (16 ビット、ビッグエンディアン、44100、ステレオ)

-f dat (16 ビット、リトルエンディアン、48000、ステレオ)

-D PCMを名前から選択。--device=NAME

(e.g.) -D plughw:[card, device]

aplay オーディオデータの再生

オプション

(aplay オプション）<https://vimvimvim.blogspot.com/2012/10/aplay.html>

arecord 音声の録音

オプション

-l マイクデバイスのリストを表示する

(e.g.) -f cd CD音質（44.1kHz)にする

amixer

(e.g.) amixer sset Mic 50 -c 1

オプション

（参考）<http://www.yam-web.net/raspberry-pi/music.html>

（参考）<https://qiita.com/yozawa/items/0695664ad89dad56e62f>

## その他

cal カレンダーを表示する

date 日時の表示，設定

オプション

-s 日時の設定。時刻省略時は 00:00:00になる。--set=

date -s "01/30 13:00 2015"

sudo date --set=" 2018-06-29 18:40:30"

ntpdate NTPサーバーを指定する。

rfkill ブルートゥースなどの電波を出さない設定

rfkill list リストの表示

screen 仮想端末制御プログラム；terminal上で複数の仮想端末を開く

よくUSB接続で他のデバイスに接続する際に（通称シリアルコンソール）で用いられる

例：

Screen /dev/ttyUSB0 115200 # ボーレート115200で接続

/dev/ttyUSB0 > console\_log.txt

tmux 仮想端末制御プログラム

md5sum MD5メッセージダイジェストを計算・チェックする

sha256sum SHA256メッセージダイジェストを計算・チェックする

sha512sum SHA512メッセージダイジェストを計算・チェックする

オプション（共通）

-c オプションで読み込んだファイルの値と照合できる　--check 　　※ちゃんとした書式があるらしい

md5sum -c checksum.txt

（補足）diffコマンドとの併用で違いを確認する方法もある

sha256sum some\_image.img > result.txt # 結果を保存

diff sha256\_download.txt sha256\_result.txt # 違いを比較

# 比較.txtファイルは、sha256の出力結果と同じ　hash\_value file\_name と言う書式で作成する。

実践：ラズパイイメージのダウンロード

[ダウンロード先](https://www.raspberrypi.org/downloads/raspberry-pi-os/) にはSHA-256というハッシュ値が記述されている。ダウンロード後に以下を実行

sha256sum 20\*\*-\*\*-\*\*-raspbian-\*\*\*.zip > raspbian-\*\*\*.sha256 # ラズパイイメージのチェック

（参考）<https://hombre-nuevo.com/microcomputer/raspberrypi/raspberrypi0025/>

convert 画像ファイルのフォーマットやサイズを変更する

at

crontab 定期的にジョブを実行するcron の設定を行う。定期的なログの取得などに用いる。

特徴としては，ユーザー権限でプログラムが実行される点。※rc.localはルート権限。

書式：at [options] *source MMDDYY hh;mm*

オプション

-f ファイル名の指定

(see also) atq キュー（予約）を表示 atrm　取り消し（remove)

(see also) crontab（コマンド）　rc.local　※こちらはスクリプトファイル

pdftk

書式：pdftk *file1,pdf file2.pdf* cat output *merged.pdf*

書式：pdftk *file1.pdf cat 1 3 7 output extructed.pdf*

インストール

sudo snap install pdftk # version 2.02-4, or

sudo apt install pdftk-java # version 3.0.9-1

## 補足情報

### 類似コマンド

#### lsシリーズ

lsblk ブロックデバイス

lscpu CPU情報

lshw ハードウェア全体

lsinitrd 初期RAMディスク

lsmod ロードされているカーネルモジュール

lspci

lsusb USBデバイス

# レファレンス（C言語）

Linux特有のライブラリをまとめる。

それ以外は別ドキュメント，C／C++.docにまとめる

なお，LinuxではC言語用ヘッダーファイルは通常 /usr/include/ に格納されている。

資料

<https://www.gnu.org/software/libc/manual/>

## <sys/ioctl.h>

### ioctl

書式：int ioctl(int fd, unsigned long request, ...);

デバイスのパラメータを制御する。

（参考）<https://man7.org/linux/man-pages/man2/ioctl.2.html>

## <unistd.h>

UNIx STanDard Header file　の意らしい

useconds\_

（参考）<https://linuxjm.osdn.jp/html/LDP_man-pages/man3/usleep.3.html>

# ディレクトリ／ファイル

## 主なディレクトリまとめ

※「分割」はルートパーティションから分割できるかどうかを意味する。

|  |  |  |  |
| --- | --- | --- | --- |
| ディレクトリ | 分割 | 傾向 | 概要／備考 |
| /home | 〇 | 書き多  肥大化 | ホームディレクトリ。各ユーザーが自由に使える。 |
| /usr | 〇 | 読み多 | 一般的に使用される共有ライブラリ。（カーネルの起動には不要  (opp) /boot  元々homeとして使われていた。後付けでUser System Resourceなどと言われるようになったらしい。 |
| /var | 〇 | 書き多 | ログやメール。 |
| /opt | 〇 | 肥大化 | 追加コマンド群。ルートとは**別パーティションを推奨**。  アドオン？？？ |
| /boot | △ |  | カーネルの起動に必要なコマンド群。(opp) /usr  意外にも，ルートパーティションとは**別のパーティションを割り当てる事が可能**。（考えてみると，USBからのブート（Live USB）がある） |
| /tmp | △ |  | メモリ上にのみ存在する一時ファイル。  一般ユーザによる書込みが可能。 |
| /bin | × |  | 一般ユーザ用コマンド群。 |
| /sbin | × |  | 管理者用コマンド群。 |
| /etc | × |  | 設定ファイル。**ルートパーティションと同じでないといけない**。 |
| /lib | × |  | 共有ライブラリ。 |
| /dev | × |  | デバイス。 |

その他

/proc 仮想ディレクトリ。起動中のプロセス，ハードウェア，システムリソースなどの詳細な情報。

/sys 仮想ディレクトリ。デバイスやドライバの情報

/run　 というのもあるが

## /home

厳密には/home/username

特徴：

ホームディレクトリ。Windowsで言うところのMy Documentのようなもの。

ルートパーティションとは別に割り当てが可能

書き込みが頻繁に発生

サーバー用途で大量の人が使う場合は，膨大な広さのパーティションを割り当てる必要がある。

### 直下ファイル

.bash\_profile .bash\_profile はログイン時にのみ実行されるshスクリプト。

.profile ログイン時に実行されるスクリプト（関連）/etc/profile

.bashrc 対話モードの bash を起動する時に毎回実行されるshスクリプト。

.bash\_history historyで表示される，過去のコマンド履歴

（参照）[シェルの設定ファイル（スクリプト）](#_シェルの設定ファイル（スクリプト）)

## /usr

特徴：

Unix System Resourcesの意。User，つまりホームディレクトリに相当、などと勘違いしがちなので注意。

起動には不要なプログラムやライブラリなどを格納。

読み込みが頻繁に発生。

ルートパーティションとは別に割り当てが可能。

### /bin：一般ユーザー用コマンド。

システムの起動には必要ない。

### /sbin：管理者用コマンド。

システムの起動には必要ない。

### /local：ホスト機固有

/usr/local/bin プログラム

/usr/local/etc 設定ファイル

補足

類似ディレクトリは/opt/ ；違いは

|  |  |  |
| --- | --- | --- |
|  | 元来の意味 | 実際の運用 |
| /usr/local | ローカル機固有のファイル | 管理者がアプリをインストールする場合など。 |
| /opt/ | 追加の拡張アドオン | 特定のアプリの拡張アドオンなど。 |

手動で配置したバイナリのsudo対応？未確認

ln -s /usr/bin/nodejs /usr/bin/node # 注意点 ：target\_path ← link 引数は対象が先、リンクが後

### /lib：ユーザー固有の共有ライブラリ

/usr/binや/usr/sbinに格納されるコマンドが必要とする共有ライブラリ

/usr/lib/rpm/rpmrc rpmツールの設定ファイル

### /include：C言語ライブラリ

C言語で使うライブラリなどが格納されている

※どこかでC言語用ライブラリは /usr/lib という話を聞いたが，実際には/usr/includeだった。(Debian)

### /src：Linuxカーネルのソース

### /shared/ ハードウェア構造に依存しないファイル

アーキテクチャ（ハードウェア構造）に依存しないファイル。

主にマニュアルなどのドキュメント類が置かれる。

|  |  |
| --- | --- |
| usr/shared/doc | 各種ドキュメント  ソフトウェアに関する更新履歴，説明書 |
| usr/shared/info | infoマニュアル |
| usr/shared/man | オンラインマニュアル |
| usr/shared/man/man1 | セクション１のマニュアル（？） |
| usr/shared/man/man2 | 同様 |

（参考）[シェルコマンド用のコマンド](#_シェルコマンド用のコマンド_1)

## /var

特徴：

ログやメールなど動的に変化する(variable)ファイルを扱う

書き込みが頻繁に発生。高速に書き込みできるディスクに割り当てるのが望ましい。

ルートパーティションとは別に割り当てが可能。

### /cache アプリケーションの一時ファイル

### /log ログ

/var/log/messages journaldが出力するLinuxのメインログ。とくにRedHat系？

/var/log/syslog 古いシステムでの標準のログ。今はjournaldに置き換わられている。

/var/log/secure セキュリティに関するログ

/var/log/maillog メールに関するログ

/var/log/wtmp ログイン関連。バイナリなので，whoコマンドでないと見れない。

### /tmp　一時的なファイルの配置場所

/tmpと/var/tmpの違いは？

## /opt

特徴：

追加でインストールしたパッケージを扱う。アドオン（拡張）アプリケーションなど。

ルートパーティションとは別に割り当てが可能（むしろ別パーティションを推奨）

参考）

<https://gigazine.net/news/20080104_linux_directory_structure/>

## /boot

特徴：

ブートローダが使用するファイル群と捉えて良い？20.10.13

Linuxカーネルや，起動に必要なファイルが格納されている。

ルートパーティションと**は別のパーティションを割り当てる事が可能**

（推奨されてはいないが，デュアルブートとかもあるので納得はできる。）

RAID上にシステムを組む場合でも，/boot だけはRAIDに含めない構成をとる場合がある。

### 直下ファイル

config.txt

ラズベリーパイでBIOSの代わりに使う設定ファイル。

### /overlays

### /grub ブートローダの設定ファイル

システムによるが，以下のどちらか１つだけが存在する。

boot/grub/menu.lst GRUB Legacy(GRUB1) の設定ファイル。

boot/grub/grub.conf ディストリビューションによってはこちら

boot/grub/grub.cfg （参考）GRUB2の設定ファイル

**但しこのファイルを直接編集する事は無い。** /etc/default/grubか/etc/grub.dディレクトリ内のファイルを編集してgrub-mkconfigコマンドを使用する。

（参考）[GRUB](#_GRUB2)

## /tmp

特徴：

**再起動すると中身が削除される**仮想データの集まり

・ルートパーティションとは別に割り当てが可能

一時ファイルが増える事が容量オーバーして他のディレクトリに障害が発生する可能性がある。

その場合は/tmp は専用のパーティションなどに割り当てる。

・スティッキービットが設定されている。具体的にはrwx rwx rwt になっている。

（参考）/var/tmp ⇒このディレクトリのあつかいはディストリビューションによって異なる。

## /bin

特徴：

カーネルが使用するコマンド群と捉えて良い？20.10.13

一般ユーザ用の基本コマンドを格納。

ディストリビューションによる違いがほとんどなく，同じようなコマンドが格納される。

ルートパーティションである必要がある

## /sbin

特徴：

管理用コマンド(Superuser Binaries) が格納される

ルートパーティションである必要がある。

rootユーザー以外は通常パスが通っていない。

### 直下ファイル

init SysVinitでは，これが最初のプロセスとなる。

nologin /etc/passwdでユーザログインを拒否するための記述 (syn) /bin/false

## /etc

特徴：

システムの設定ファイル。ルートパーティションである必要がある。

### 直下ファイル – 一般

ld.so.conf

Linker Loader の意らしい。

ライブラリの検索時に，/lib や /usrlib以外も検索させたい場合，ここに記述して置く。

毎回このファイルを参照するのは効率的でない為に，キャッシュファイルが存在する（/etc/ld.so.cache）

（関連）ldconfig，ld.so.cache

ld.so.cache

プログラムの実行時に，共有ライブラリの場所を検索するために使用されるファイル

ldconfigコマンドが実行された時に，/etc/ld.so.confファイルを参照してこのファイルは更新される。

（関連）ldconfig，ld.so.conf

os-release

os-release

rc.local

Linuxの起動時にroot権限で実行されるスクリプト。

profile

ログイン時に実行されるスクリプト

environment

デフォルトの環境変数を設定する

/etc 直下ファイル　続き

updatedb.conf locateコマンドで使うデータベースupdatedbの設定ファイル。

profile 全ユーザーに適用される環境変数

bashrc 全ユーザーに適用されるシェル変数？　読み込まれる順番は一番後。

mtab 現在マウントされているデバイスの一覧。近年では/proc/self/mountsへのシンボリックリンク。

(compare) etc/fstab ファイルシステムのマウント設定を記載するファイル

network

### 直下ファイル – ネットワーク系

hosts

IPアドレスとホスト名が羅列されたリスト。

簡易DNSの様にも使えて，DNSより先に参照されるのでテスト段階のサーバーなどにも用いられる。

resolv.conf 利用するDNSサーバー情報

/run/system/resolve/stub-resolv.conf へのシンボリックリンクらしい。

nameserver

search

### 直下ファイル – パッケージ

yum.conf yumツールの設定ファイル

yum.repos.d yumツールのリポジトリ設定

### 直下ファイル – 管理

group システムに存在する全てのグループ情報が格納。GID，グループパスワードなど。

hosts mDNSで使うホスト名とIPアドレスのペアのリストが書かれたテキストファイル。

hostname ホスト名を格納するテキストファイル

shadow 暗号化されたユーザーパスワードやパスワードの有効期限が含まれる。

ここには/etc/groupのものは含まれない。

#### passwd

システムに存在する全てのユーザー情報が格納。システムユーザーの情報も含む。

ファイル所有者はroot。パーミッションは644。

cat /etc/passwd

grep user\_name /etc/passwd # grepを使って特定のユーザーを調べる場合

結果の一例

> user01:x:1004:1005::/home/user01:/bin/bash

第1フィールド：ユーザ名

第2フィールド：パスワード

x と言う表記がある場合は/etc/shadow へ暗号化されて格納されている事を示す。

第3フィールド：ユーザID

第4フィールド：グループID

第5フィールド：コメント欄

第6フィールド：ホームディレクトリ

第7フィールド：ログインシェル（デフォルトシェル）

/sbin/nologin の場合，ログインできない事を示す。

### 直下ファイル – その他

#### fstab 自動マウントの設定

利用するファイルシステムの自動マウント設定ができる。コマンドはmount -a

ここにUUIDの記述のあるデバイスは一般ユーザーでもマウントできるようになる。

（比較）/etc/mtab 現在マウントされているファイルシステムの情報が格納

詳細：

|  |  |  |
| --- | --- | --- |
| 列名 |  | 詳細／具体例 |
| <device> | デバイス名 | UUID=XXXX-XXXX  /dev/sda1 ；LABEL=/boot |
| <dir> | マウントポイント | /media/usb/naspi |
| <type> | フォーマット | vfat : vfat:FAT exfat-fuse: exFAT  ※cdrom の場合iso9660 などもある |
| <options> | async:　I/O は非同期で行われる (opp) sync  auto : 　起動時に自動でマウント (opp) noauto (関連) mount -a  dev : ファイルシステム上の特別なデバイスを解釈する (nodev)  exec : バイナリの実行を許可する (opp) noexec  rw : 読み書き可能 (opp) ro = read only  umask : 777 - umask がディレクトリのアクセス権になる  users :  users グループ全員 user: 一般ユーザーのマウント許可，本人のみアンマウント可。  owner: デバイスの所有者 **nouser**: rootだけマウント，アンマウント可  uid, gid  ※ user とともに使う。[ uid=1000,gid=1000, username=username,password=password ]  ※0はroot，65534 はnobody(nogroup) を示す  suid: SUIDとSGID を有効に　※アクセス権限のビットの事 (opp) nosuid  noatime, nodiratime, relatime ドライブのパフォーマンスが改善  defaults: デフォルト設定(async, auto, dev, exec, nouser, rw, suid) | |
| <dump> | dump コマンドのバックアップ対象 | 0:対象外 1:バックアップ対象。　ほとんどの場合は 0 |
| <fsck> | fsckのチェック順序  （起動時に行われる） | 0: fsckでチェックを行わない  1: fsckを行う（/　のみ）  2: fsck必要（/　以外で指定する場合）  数字が小さいものが優先度が高い  近年のものは列名が<pass>になっている事がある。その他は同じ。 |

# 具体的にはこんな形になる

UUID=XXXX-XXXX /media/usb/name vfat async,auto,dev,rw,users,umask=000 0 0

※noauto があっても，オプションなしのmountオプションではマウントされる。

<options> 追記

errors=remount-ro という記述を見た

UUIDの確認の仕方

名前 確認の仕方 詳細

UUID ls -l /dev/disk/by-uuid/ ファイルシステムレベル。

PARTUUIDパーティションテーブルレベル

PTUUID sudo blkid /dev/sda1

sw とは何？20.11.09

関連）df

#### resolv.conf

DNSへのアクセスを提供する。

nameserver ネームサーバーのIPアドレス。複数行指定可能。

nameserver 192.168.0.0

domain 自身が所属するドメイン名を指定する。

InterNIC (日本では JPNIC) に登録されている名前を書く（らしい）

domain doc.com

search ホスト名ルックアップのための検索リスト。複数の指定可能。

search dm1.example.org dm2.example.org

sortlist

options

#### inittab

起動の際の情報を格納

SysVinitを採用したシステムにおいてinitやtelinit で起動する時に必要な起動に関する情報を格納している。なお，各ランレベルに応じたスクリプトは /etc/rc[0-6].d に入っている。

K01alsa-utils のようにKから始まるものはサービスを停止させるファイル。

但しUbuntuなどsystemdを用いたシステムでは存在しない。代わりに存在するのが/etc/init/rc-sysinit.conf

systemdはランレベルの代わりに「ターゲット」を用いる。

SysVinitを採用したシステムではCtrl+Alt+Delを押した時の動作をこの設定ファイルで指定する。

ctrlaltdel 「ctrl+alt+del」キーを押下した際のシステムの動作を設定

なぜ/etcに格納されるのか？あくまで設定ファイルだから？

### apt/

/etc/apt/sources.list aptが取得するパッケージの取得元（リポジトリ）を設定する。

### default/

/etc/default/grub GRUB2の間接的設定ファイル。

このファイルを編集して，grub-mkconfig コマンドで設定内容を適用する。

本体のファイルは/boot/grub/grub.cfg だが，このファイルを直接編集する事はない。

### netplan/

補足：ラズパイには存在しなかった。Ubuntuでは存在を確認

### gdm3/

/etc/gdm3/custom.conf

sudo systemctl enable gdm.service

比較：KDM，Lightdm

### /lightdm

/etc/lightdm/lightdm.conf

デスクトップマネージャー。自動ログインの設定など？

Ubuntu20

### /network

/etc/network/if-up.d/ 上りのルートに関するスクリプトを保持

/etc/network/if-down.d/ 下りのルートに関するスクリプトを保持

#### interfaces

/etc/network/interfaces

IPアドレスの設定に使われる。

source /etc/network/interfaces

iface wlan0 inet dhcp

address 192.168.0.10

netmask 255.255.0.0

gateway 192.168.0.1

dns-nameservers 192.168.0.2

特に一昔前はこの設定ファイルを用いていた様子。

DHCPの場合？

iface eth0 inet dhcp

wireless-essid myessid

wireless-key 123456789e

[https://qref.sourceforge.net/Debian/reference/ch-gateway.ja.html#s-net-dhcp](https://qref.sourceforge.net/Debian/reference/ch-gateway.ja.html" \l "s-net-dhcp) 情報元

### /dpkg

/etc/dpkg/dpkg.conf dpkgの設定ファイル。

### /skel

ユーザー追加時のホームディレクトリのひな形になる。（そのままコピーされる）。skeleton の意らしい。

（関連）useradd，adduser

### /.ssh

大体デフォルトで存在するが，ない場合は自分で作成する必要がある。

なお，このディレクトリ（/etc/.ssh）の権限は700限定らしい（20.11.09未確認）

~/.ssh/sshd\_config SSHデーモンの設定ファイル

|  |  |  |
| --- | --- | --- |
| 項目 | 値 | 説明 |
| AllowUsers | 略 | ログインユーザー等の制限。user@ip\_addr となる。 |
| Match User |  |  |
| Port | 数値 | 接続を受け付けるポート番号。 |
| PermitRootLogin | yes/no | rootユーザーでのログインの可否。 |
| PasswordAuthentication | yes/no | パスワードでのログインの可否。 |
| PubkeyAuthentication | yes/no | 公開鍵でのログインの可否。 |
| Protocol | 1/2 | 2が推奨らしい |
| ChallengeResponseAuthentication | yes/no | パスワード認証が否なのでこちらも否 |

~/.ssh/config sshクライアントの設定ファイル

ログインが楽になる，というだけで必須ではない。

※設定項目の詳細などは，別ドキュメント「ソフトウェア情報 – OpenSSH」に移動

### /systemd

/etc/systemd/system/default.target システム起動時に最初に実行されるUnitへのハードリンク。

/etc /systemd/system/\*.wants 自動起動するターゲット（≒ランレベル）の追加されているもの

ls -l /etc/systemd/system/default.target #

ls -l /lib/systemd/system/default.target # Ubuntuではここにあったが？

### /udev/

/etc/udev/rules.d

udevがデバイスファイルを動的に作成する時の挙動を定義する

.rules設定ファイルが格納されるディレクトリ。

### /xdg

LXDEと言うデスクトップ環境の設定ファイルを格納するディレクトリ。

autostart

2017年前後のDebian系で使われていた。

参考：https://qiita.com/karaage0703/items/ed18f318a1775b28eab4

lxsession/LXDE-pi/autostart

### /wpa\_supplicant

/etc/wpa\_supplicant/wpa\_supplicant.conf ネットワーク設定。

## /lib

特徴：

/bin や /sbinに格納されたコマンドを実行する際に必要となる共有ライブラリを格納。

ルートパーティションである必要がある。システムの起動に必要。

比較

/lib 共有ライブラリ

/lib64 64ビット版共有ライブラリ（usr/lib64）

/usr/lib 起動には不要な，一般的なアプリケーションで使用される共有ライブラリ。

### /systemd/system 起動(init)に関する設定ファイル(Unit)を格納

/etc/system/system/default.target に置くのがシンボリックリンク。

リンク先はこのディレクトリで，標準ランレベルへのリンクとなっている。

参考）[Unit](#_Unit)

### /shared

一般的な共有ライブラリの格納場所

関連　/usr/lib

### /udev

#### /rules.d

50-udev-default.rules

## /dev

特徴：

ハードウェアへのアクセスを抽象化した「デバイスファイル」が格納される。

ルートパーティションである必要がある。

紛らわしい点：

デバイスファイルはデバイスドライバとは異なる。各プログラムが**デバイスを扱う為の仮想ファイル**。

デバイスドライバはカーネルの機能（カーネルモジュール）として利用できる。

ハードウェアの情報が格納されている訳ではない。ハードウェア情報はdev/proc

メモリ上のファイルでは無く，デバイスファイルは物理的に存在する。

（補足）通常ファイルと言えばテキストファイルかバイナリファイルにだが，Linuxではあらゆるものをファイルとして扱う。

メモ

ホットプラグデバイスの制御はudev(Userspace Device management) によって行われる。

カーネルはデバイスを検知すると，まず/sys以下（sysfsと呼ばれる）の情報を更新する。その情報をudevが検知し /dev 以下にデバイスファイルを動的に更新する。その時の動作は /etc/udev/rules.d ディレクトリに配置された，拡張子に.rulesを持つ設定ファイルによって定義されている。

### 直下ファイル

#### tty\*

ttyは標準入出力の意

ttyS0 シリアル通信デバイスの1つ目。

ttyUSB0 USB接続デバイスの1つ目。

video\*

カメラデバイス

### デバイス名

#### /fd0 フロッピーディスク

fd0, fd1 となる。

#### /cdrom　CDドライブ

#### /sda scsi(sata)接続されたストレージ

１つ目のディスクがsda，２つ目のディスクがsdbとなる。

次に，パーティション番号に従ってsda1, sda2となる。

#### /hda IDE接続されたストレージ

#### /disk 永続的なマウントポイント（？）

/disk/by-uuid

ls -l /dev/disk/by-uuid # マウントポイントの一覧

ls -l /dev/disk/by-uuid | grep sda1 # そのままgrepで検索する場合

/disk/by-label

#### その他のデバイス名

/vda レンタルサーバーなどでの仮想ストレージ？

/mmcblk0

SDカードを示す。最初に見つけたSDカードが0番，次が1番になる。(mmcblk1)

Multi Media Card Blockの略？

### /loop ループバックデバイス

内部ファイルを，あたかもブロック型デバイス（外部ストレージ）であるかの様にあつかう。

パソコン上でイメージ・ファイルを直接操作したい場合などに使う。ISOイメージの中身の確認なども行える。以下のようにして用いる。

# ディスクイメージを作成

dd if=/dev/zero of=loop1.img bs=1M count=100

# 作成したイメージを/dev/loop0と関連付ける

sudo losetup /dev/loop0 loop1.img

### /tty 出入力を扱う

標準出力，シリアル通信でのデバイス等を扱う

なお，ttyコマンドで現在の標準出力になっているものが表示される。

※本来はTeleTYpewriterの意らしい

### /null

ファイルのデータ消去で使う特別なディレクトリ。ファイルの中身が空になる。

dd if=/dev/null of=/home/*user1*/test.txt # test.txt の内容を空にする

/dev/null 2>&1 # コマンドが出力する全てのメッセージを破棄（ディスプレイに非表示）

### /zero

ファイルのデータ消去で使う特別なディレクトリ。いわゆるゼロクリア。

/nullと違うのは bs, count のオプションと併用しないと，対象ディレクトリまたはデバイスのデータを際限なしに0で埋め尽くす（完全に破壊）。

dd if=/dev/zero of=/home/plcpi/src/test.txt bs=1024 count=3 # test.txt

（参考）[dd](#_dd)

### /urandom

cat /dev/urandom > result2.txt

## /proc

特徴：

起動中のプロセス，カーネルが認識しているハードウェア，

およびシステムリソースなどの詳細な情報を扱う。メモリ上の仮想ファイルシステム。

（仮想と言う意味ではtmpディレクトリに少し似ているが，Linuxシステムに特化している点が異なる）

※RedHatにはこのディレクトリが存在しないらしい。

### 直下ファイル

cmdline ブートローダからカーネルに渡されたパラメータ

cpuinfo CPUに関する情報

dma 使用中のDMAチャンネルに関する情報

cat /proc/dma # このようにして確認する。

※DMA；Direct Memory Access Channel

1台のパソコンで複数の作業を同時に処理する際、CPUの負担を減らすしくみ

interrupts IRQに関する情報。

IRQ(Interrupt ReQuest) とはマウスなどの割り込み要求の事。

ioports I/Oアドレスの情報。I/Oポートアドレスなどを知る事ができる。

mounts カーネルが提供する，全マウントデバイス

meminfo メモリに関する情報。メモリの使用状況。

modules ロードされているカーネルモジュールに関する情報

### /net

/proc/net/arp 現在の（メモリ上の）arpテーブル？

### /PID プロセス情報

実際にはPID番号をディレクトリ名とする。例 /proc/1771/ など。

sudo ls /proc/1 # PID1，つまりinitプロセスのファイル情報をみる

### /bus システム固有の内部バス

システム上で使用可能な様々な内部バスの固有の情報が格納

|  |  |
| --- | --- |
| /proc/bus/usb/devices | 接続されたUSBデバイスの情報を表示。また，lsusb コマンドでも確認できる。  ※旧式のみ？Ubuntuなどでは存在しない 2021-08-31 |
| /proc/bus/pci/devices | PCIデバイスに関する情報 |

### /scsi

/proc/scsi/scsi SCSIデバイスに関する情報

### /self 　現在実行中のプロセス

/proc/self/mounts 自プロセスでマウントしているもの？？

（参考）/etc/mtab，/proc/mounts

### /sys カーネルの諸設定

## その他のディレクトリ

### 概要

|  |  |  |
| --- | --- | --- |
| ディレクトリ | 特徴 | 概要／備考 |
| /media |  | CDなどの一時的なマウントポイント |
| /mnt |  | 一時的なマウントポイント |
| /sys |  | デバイスやドライバの情報。  メモリ上の仮想ファイルシステム。 |

### /mnt

取り外し可能な装置に対する、一時的なマウントポイント用ディレクトリ。

マウントしたデバイスの「/（ルート）」として使用される。

新しいシステムでは/mediaに置き換わる傾向。

関連：/media

## 特筆すべきファイルなど

### デスクトップエントリ

https://wiki.archlinux.jp/index.php/%E3%83%87%E3%82%B9%E3%82%AF%E3%83%88%E3%83%83%E3%83%97%E3%82%A8%E3%83%B3%E3%83%88%E3%83%AA

## 別パーティションの利用

### homeディレクトリ

補足：ここでは外付けドライブ/dev/sdbにhomeディレクトリを移すものとする。

# パーティションとファイルシステムを作成（フォーマット）

# 補足：マウント先のシステムのterminalでやったほうが無難

sudo fdisk /dev/sdb # 対話形式でパーティション作成

Command (m for help): d → 2 # パーティション番号1を削除

Command (m for help): n　→ p → 2 # primary のパーティション1を再作成

# ファイルシステムを変更（フォーマット）

sudo mkfs.ext4 /dev/sdb1

# 外付けドライブをマウント

sudo mount /dev/sdb1 /mnt/

# ファイルをすべてコピー（下のいずれか）

sudo rsync -aX /home/\* /mnt

sudo cp -arp /home/\* /mnt

# 念の為ファイルがすべてコピーできたかを確認

diff -r /home /mnt

# もとの/homeのファイルを削除（必要？）

sudo rm -rf /home/\*

# 一度アンマウントして、homeディレクトリとして仮マウントする

sudo umount /dev/sdb1

sudo mount /dev/sdb1 /home/

# 結果を確認

df -hl

# fstab編集の為にデバイスのUUIDを取得（以下のいずれか）

lsblk --output NAME,UUID,FSTYPE

blkid /dev/sdb1

再起動時の自動マウントの為にfstabを編集

[ /etc/fstab ]

UUID={ID} /home ext4 defaults 0 2

補足

ext4でなく、FAT32なども選択可能だが、FATは権限をサポートしていないためよろしくない。

fstabではデバイス名指定でも可。詳細はfstabのマニュアルを参照。

参考元１：[https://ja.linux-console.net/?p=2430#gsc.tab=0](https://ja.linux-console.net/?p=2430" \l "gsc.tab=0)

### varディレクトリ

### 資料

homeディレクトリのファイルのコピー例２：Google Coralの例から

sudo rsync -aXS --exclude='/\*/.gvfs' /home /mnt

sudo diff -r /home /mnt/home -x ".gvfs/\*"

以下、不要っぽい

# 所有者の変更

sudo chown -R mendel:mendel /mnt/home/mendel

ホームディレクトリが無くなってしまった場合

usermod -d /home/mendel –m my\_user # ホームディレクトリを再設定

sudo chown -R my\_user:my\_user /home/my\_user # 所有者の変更

参考

[https://ja.linux-console.net/?p=2430#gsc.tab=0](https://ja.linux-console.net/?p=2430" \l "gsc.tab=0)

<https://www.tecmint.com/convert-home-directory-partition-linux/>

# 標準サービス

## 概要

※どこからがサービスで、どこからがアプリかは区別しにくいが、ここでは標準的なサービスをまとめる。

hostapd 無線LAN(Wifi)のアクセスポイントを提供する

# パッケージ（アプリ）

## 概要

### 標準

Postfix # メール転送エージェント。 (see also) dovecot

Dovecot # IMAPとPOP3のサーバ (see also) postfix

Apache #

nginx # （エンジンエックス）軽量なWebサーバー。近年Apache よりも人気。

Nextcloud# オンラインストレージ機能

could-init # クラウドインスタンスを展開する為のカスタマイズを行う

ownCloud#

MySQL

MariaDB # データベース

PostgreSQL#

Samba #

NFS ファイル共有ソフト

GIMP ビットマップ画像の加工・編集。やや重い（らしい）

Pinta 動きの軽いペイント風アプリ

KolourPaint

GrafX2 ドット画の編集がしやすい

cairo Gtkに含まれる２次元の図形を描くライブラリ

Inkscape ベクター画像の加工・編集

KVM Kernel-based Virtual Machine

仮想化基盤。カーネルのモジュールとして機能するのでカーネルコンパイルの段階でモジュールを有効にしなければ利用できない。

Xen ゼン。１つのハードウェアを用いて、複数のOSを並列実行・制御する。

libglu1-mesa

LXC Linux Containers．コンテナ型ソフトウェア。（同一OSに限定した部分的仮想化）

### 有志

progress cpなどの進捗の表示：<https://github.com/Xfennec/progress>

## 基本設定

### ALSA（オーディオ）

#### aplay

オプション

-l 再生デバイスのリストを表示

--device

## 事務用途

### pdf編集

#### Master PDF Editor 5

有料だが，無料制限の範囲内でもそこそこ使えるらしい。

[https://code-industry.net/free-pdf-editor/#get](https://code-industry.net/free-pdf-editor/" \l "get)

#### LibreOffice：Draw

こちらでもpdfの編集はできる。ただし保存する際はpdfにエキスポート，といった形を取るしか無い。

### vi

vi [options] *filename*

-R # 読み取り専用で開く

ファイル名を省略した場合は空のファイルで開き，保存する際にファイル名を指定する事になる

・コマンドモード

カーソルの移動，コピー＆ペースト，ファイルの保存などを行う。viの起動直後はコマンドモードになっている。コマンドの文字列は，画面最下に表示される。

・入力モード

入力モードではテキスト本文の編集を行うことができます。

コマンドモードに戻るにはEscを押す。

・設定ファイル

.vimrc と言うファイル名

・方向キーでABCDが出る問題

:set nocompatible # 互換モードの解除。これで解決する。

#### コマンドモード：標準

i **カーソルの左**から入力開始　insertから

a **カーソルの右**から入力開始　appendから

I カーソルのある**行の先頭**から入力開始

A カーソルのある**行の末尾**から入力開始

o カーソルのある行の**下に空白行を挿入**し，その行から入力開始。**o**pen line から

O カーソルにある行の**上に空白行**を挿入し，その行から入力開始。

Esc 入力モードからコマンドモードへ移行

Ctrl + [ 入力モードからコマンドモードへ移行

:q 終了。内容が変更されている場合には警告が出る。quitから。(also) qa, qall

:q! 強制終了

:w [ *filename* ] 編集内容を保存。 writeから

:wq 保存して終了し，viを終了　 (also) :x(バツ印ボタンと覚える), ZZ

#### コマンドモード：移動

コマンドの前に数字をつけると，その回数分コマンドが繰り返される。e.g. 5 カーソルが5文字下へ

移動：h←ひ(Hi)だり j↓地(Ji)面 k↑：昇竜Ken l →：lだけど右

⇒ 派生：5h カーソルを左へ５つ移動させる

0 行頭へ移動　※これはオーでなくゼロ

^ その行の最初の文字へ 正規表現の^（行の先頭）

$ 行末へ移動 正規表現の$（行の最後）

⇒ 「山(^)（≒困難）が行（行動）の初めに来て，金($)は最後に来る」とでも覚える

gg **ファイル**の先頭へ移動 g（小文字）は頭の方が重そう。G（大文字）は下の方が重そう

G **ファイル**の最終行へ移動 でっかいG（重力）は下に行く　とでも覚える。

H **画面**の最上行へ移動 Home of window

L **画面**の最上行へ移動 Lowest of window

nG ファイルのn行目へ移動 e.g. 5G ファイルの5行目に移動。Gyou（行）で覚える

G最終行へ移動と混同しないように

:n ファイルのn行目へ移動 e.g. :5 ファイルの5行目に移動。

Ctrl+f 次の画面へ移動 Forward

Ctrl+b 次の画面へ移動 Back

#### コマンドモード：検索／置き換え

/ 文字列 指定した文字列を前方向に向かって検索

? 文字列 指定した文字列をカーソル位置から逆方向（先頭に向かって）検索

/sbin # sbin という文字列をファイルの末尾にむかって検索

?sbin # sbin という文字列をファイルの先頭に向かって検索

# / と ? は同じキー。?で検索できるが，逆方向に検索する時は手間(shiftを押す) と覚える

n 更に検索（前へ）

N 更に検索（後ろへ）

:%s /*str1*/ *str2*/g置き換え %はファイル全体の行，sは置き換え，gは範囲内の全てのstr1を置換

g が無いと最初のstr1のみ置換対象。sedに似ている。

:'s/ping-t/hoge/g' # ping-tという文字列を全てhogeに変換する

#### コマンドモード：編集

yy カーソルの位置の**行**をコピー copyy 　又はYでも良い ≒dd（ただし削除無し）

5yy # 現在のカーソルのある行を含めて5行コピーする

yw カーソルの位置の**単語**をコピー copY Word？ ≒dw（ただし削除無し）

dd カーソルの位置の**行**を切り取り

dw カーソルの位置の**単語**を切り取り delete word？

x カーソル位置の**1文字**を切り取り delボタンでも同様

X カーソル位置の**左の文字**を切り取り

p カーソルの右側（／下）に貼り付け past

yyp # 現在の行をコピーして下の行に挿入（２つ以上のコマンドが組み合わせられる,という例）

P カーソルの左側（／上）に挿入貼り付け　LARGE letter はsmall の逆方向

u 元に戻す。Undo

U 行全体の操作を取りやめる

※編集モードの時にカーソルキー（←↑↓→の矢印）を押すとABCD と文字がタイプされる。

#### コマンドモード：その他

:nu 行番号が下の方に表示　（参考） (also) number

:set nu 行番号を表示。何故かset numではない　(also) :set number

:set nonu 行番号を非表示　(also) :set nonumber

:set ts=*1* タブ幅を数値で指定 (also) :set tabstop=*1*

:! viエディタを終了することなくLinux(bash) コマンドを実行できる

:r *filename* 指定したファイルの内容を読み込んでカレント行の後に挿入 read file

## Web系

### npm

#### 概要

Node.js（JavaScriptの便利ライブラリ）のパッケージを管理する。

インストール

※2021-11-29 apt経由でインストールしたら、「新バージョンを入れろ」というエラーが発生。

<https://nodejs.org/ja/> にアクセス→**ソースコード**をダウンロード

tar xzvf node-v16.13.0.tar.gz # 展開（バージョンは2021-11-29現在のもの）

cd node-v16.13.0 # 移動

./configure # インストールに必要な環境変数やライブラリを設定

make # make

make install # インストール（1時間くらいかかる）

#### サブコマンド

install (also) i, [ npm i となる]

書式 npm install パッケージ名

オプション

-g グローバルにパッケージをインストールする。 (also) --global

補足：-gオプションを使用した際は--saveオプションは無効

sudo npm install npm -g # アップデート

# アップグレードは？

sudo npm install npm@6.14.4 # バージョンを指定する方法

uninstall (also) un

書式 npm uninstall パッケージ名

unlink

書式 npm unlink パッケージ名

remove (also) rm, r, [ npm r となる]

書式 npm remove パッケージ名

search

書式 npm search 検索キーワード

list

オプション

-g グローバルインストールされたパッケージを表示する

#### オプション

-S package.json の dependencies欄 にパッケージ名が記録される (also)--save,-save

補足：近年のバージョンでは標準でdependencies欄への記載されるので不要オプションとなったらしい

package.json

カレントディレクトリ配置されるファイル。package.jsonにはインストールするモジュールを記入することができる。

#### 資料

npm ERR!の対処法：<https://qiita.com/wafuwafu13/items/2fe43414aa6e1899f494>

補足

関連：yarn Node.jsを管理するパッケージマネージャ

flat

## ネットワーク系

### dnsmasq

概要

DNS，DHCP，TFTP を提供。TFTP（Trivial File Transfer Protocol）　UDPを用いたファイル転送プロトコル。その為、信頼性よりも転送効率を重視。FTPのようにユーザ名とパスワード認証が行われない。

設定ファイル：/etc/dnsmasq.conf

domain-needed 「.」またはドメインを含まない名前は上位 DNS にフォワード(転送) しない

bogus-priv プライベート IP は上位 DNS にフォワード(転送) しない

local=/my-local-domain.com/ ローカルドメイン設定

expand-hosts ドメイン自動補完 :「domain=」にドメインが設定された場合に有効

domain=my-local-domain.com

dhcp-host 接続ホストの情報？

dhcp-host = aa:aa:aa:aa:aa:aa, hoge, 192.168.0.10, infinite

設定ファイル：/etc/dnsmasq.more.conf

追加の設定ファイル。内容は/etc/dnsmasq.confと同じ。

補足：/etc/dnsmasq.confでconf-file=/etc/dnsmasq.more.confの#を外し、有効化する必要がある。

関連ファイル：/etc/hosts

DNSサーバー自身のIP アドレス・FQDN・サーバ名を必ず登録

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

192.168.20.1 server01.my-local-domain.com server01

### gFTP

FTPやSCPのGUIソフト。（SCPはSSH2を選択）　WindowsのWinSCPのようなもの。

### vsftpd

FTPサーバーのソフト

### samba

#### 概要

設定ファイル /etc/samba/smb.conf

ログ /var/log/samba/log.smbd

#### コマンド

pdbedit

#### 設定ファイル

security 説明

user ユーザー名とパスワードでローカル認証を行う（デフォルト）

share パスワードだけで認証を行う

domain ドメインコントローラにより認証を行う

server 他のSMBサーバにより認証を行う

ads ADドメインのドメインコントーローラで認証を行う

read only

def.=yes

参考：https://qiita.com/leomaro7/items/186ed1f9495233e2c6b9

### net-tools

ifconfig, arp, routeなど

実は非推奨のツールではある。

sudo apt install net-tools

## 開発系

libusb-dev USB アプリケーションを プログラミングするためのライブラリ。

## 仮想化

### Docker

別ドキュメント「ソフトウェア情報.docx」に移動。

### chroot

rootよりも下位の特定のディレクトリを、別の仮想的なrootディレクトリと定める技術。

コンテナ化の基礎になった技術

## デバイス系

### v4l-utils

Webカメラの情報を取得する

インストール

sudo apt install v4l-utils

## オーディオ・ビジュアル系

### ffmpeg

ffmpeg -i INPUT\_FILE -vf OUTPUT\_FILE

#### オプション

-i 入力ファイルを指定する

-v

-list\_devices [true/false] 接続済のデバイスの表示

flags

-f [format]

動画系 matroska

音声系 mp3

関連：-formats使用可能なformatを表示する

scale width:height[:flags]

16:9の代表的解像度

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 通称 | width(px) | height(px) | ffmpegオプション |  |
| 8K | 7680 | 4320 | -s 7680x4320 |  |
| 4K | 3840 | 2160 | -s 3840x2160 |  |
| WQHD | 2560 | 1440 | -s 2560x1440 |  |
| FHD | 1920 | 1080 | -s 1920x1080 |  |
| WXGA++ | 1600 | 900 | -s 1600x900 |  |
| HD | 1280 | 720 | -s 1280x720 |  |
| DVD | 720 | 480 |  |  |

val=-1 で自動になる（アスペクト比維持）

scale=-1:2160 # 4K に指定

flags (fast\_bilinear, bilinear, bicubic, experimental, neighbor, area, bicublin, gauss, sinc, lanczos, spline)

lanczos

rotate

#### 具体例

mp4でエンコード

ffmpeg -i INPUT\_FILE -vf scale=-1:1080 -vcodec libx264 OUTPUT\_FILE

音声をmp3にする

ffmpeg -i INPUT\_FILE -acodec libmp3lame -ab 160k OUTPUT\_FILE # ビットレート160k

ffmpeg -r 12 -i image\_%04d.jpg -vcodec libx264 -pix\_fmt yuv420p -vf "scale=trunc(iw/2)\*2:trunc(ih/2)\*2" -r 24 out.mp4

#### 資料

参考資料：<https://wave.hatenablog.com/entry/2016/09/14/080900>

## その他

### p7zip

7zipを扱う。

p7zip-full

書式：7z サブコマンド [<switches>...] <archive\_name> [ ファイル名...] [<@listfiles...>]

サブコマンド

x 展開

a アーカイブの作成（圧縮）

圧縮例

$ 7za a /tmp/test.7z /home/test.gqv

解凍例

$ 7za x -o"/tmp" /tmp/test.7z

### lame

lame -b 128 sample.wav sample.mp3 # mp3へのエンコード

# セキュリティ関連

## カーネルロックダウン

Linuxカーネルの5.4(2019年11月26日)から搭載

悪意のあるユーザーがroot権限を手に入れた場合を想定し、rootユーザであってもシステムの変更を伴う操作に制限を付ける機能。

# memo

## 自身のグローバルIPの確認方法

curl inet-ip.info

curl http://httpbin.org/ip

curl globalip.me

xxx.xxx.xxx.xxx

# 用語

## 一般

block ファイルシステムに依存するファイルの最小領域。通常sectorサイズの整数倍。

D-Bus；Desktop Bus

プログラム同士が情報を伝達するプロセス間通信（IPC；Inter-Process Communication）機構のひとつ。

この機能により，例えば新しいデバイスの認識情報を他のアプリケーションに伝達し，そのアプリケーションが新しいデバイスをすぐに使えるようになる。Linuxではdbus-daemonなどを用いる。

systemdを使っているならD-Busは自動で有効にされている。

ESP；EFI System Partition

UEFIファームウェアがロードするUEFIブートローダやアプリケーションを格納する物理パーティション

FAT32でフォーマットされている。

UEFIに準拠するコンピュータで使われている。

Linux だけでなく，Windows でも使用されている。

FHS；Filesystem Hierarchy Standard

ディストリビューションに依存しない，Linuxの標準的なディレクトリの構造を定めた規格。

但し，必ずしもこの規格に準じてはいない為，把握しなければならないファイルなどもある。

GDB；GNU Debugger

GNUソフトウェア・システムで動く標準のデバッガ。

GPL；GNU General Public License

プログラムの利用，改変・改良，改良版を含めた再頒布を許諾しているライセンスで，ソフトウェアの一部にGPLを持つソースコードを使用した場合でもGPLは適用される。

PMAP

プロセスのメインマッピングに関する情報をprocファイルしすてむから取得し，表示するコマｊンド。

UEFI

BIOSに代わる新しい起動システム。最初にアクセスされるESPは/boot/efiにマウントされる。

GPTというパーティション形式に対応しており，3TB以上のHDDからの起動をサポートする。

特に近年のWindows系のブートローダで採用されている。（参考）grub

UUID；Universally Unique Identifier

ソフトウェア上でオブジェクトを一意に識別するための識別子。

※仮想マシンの設定情報もUUIDとして扱われる？？？

inode ファイルやフォルダに与えられる一意の整数値

・SUID；Set User ID 実行ファイルに付与され，指定ユーザーの実行権限で実行される。

言い換えると，実行時に一時的に所有者として実行する。所有ユーザの項に付与される。

・SGID；Set Group ID ディレクトリに付与され，指定グループの権限となる。

実行時やファイル作成時には指定グループとして実行される。 所有グループの項に付与される。

（関連）[chmod](#_chmod)，[ls](#_ls)

LVM；Logical Volume Manager

複数の物理ボリュームから１つの「論理ボリューム」を構成し，ハード的なパーティションの圧迫問題の対策となる仕組み。

論理ボリュームは動的にサイズを変更できる。

単体のディスクを超えるサイズの論理ボリュームの作成，新たな物理ボリュームの追加が可能。

特定のボリュームだけをボリュームグループから排除する事が可能

なお，論理ボリューム(LV)のデバイスファイルは /dev/VG名/LV名 となる。

LXDE

PIXEL

POSIX

Portable Operating System Interface；ポジックス

UNIX系OSに共通する機能などの標準を定めた規格。

sector HDDなどの物理的な作りに依存するサイズ区分

fdisk -l /dev/sda1 | grep 'Sector size' # fdiskでセクタサイズ、ブロックサイズを確認

fdisk -l /dev/sda1 | grep 'セクタサイズ' # 日本語環境の場合

whatisデータベース オンラインマニュアルページ(manページ)に関する情報を蓄積するデータベース

XFS eXtents File System

シリコングラフィックス社(SGI)が開発した高性能ジャーナリングファイルシステム。

ジャーナリングファイルシステム

新履歴をログに記録しておく機能。ジャーナルに記録されている情報を使用することで，障害時のデータ復旧を短時間で行うことができます。ext3以降のファイルシステムに付属。

ボリュームグループ

物理ボリュームを集めて構成された仮想的な領域。≒RAIDなどで使う？仮想的なハードディスク。

（参考）vgcreate

拡張ファイル属性

extended attribute

ファイルにカスタムのメタデータを結びつけるファイルシステムの機能。

ファイル属性にはパーミションなどLinuxシステムで共通に定義されているが、そのカスタム版と考えると分かりやすい。

## 物理ボリューム

PV：Physical Volume　物理的なハードディスクやパーティション

（参考）pvcreate

## 倫理ボリューム

LVM(Logical Volume Manager) によって作成される，物理ボリューム内の仮想的なパーティションの事で，サイズを後から変更したりとかなり柔軟な管理が可能。

（参考）lvcreate

### スナップショット

倫理ボリュームへの変更前の情報を保持する機能。任意のバックアップ対象を含める事で，運用を停止する事なく，安全にバックアップを行う事ができる。

## その他情報

### 公式資料

man ページのオンライン版：<https://www.die.net/>

### LPIC

試験内容はほとんど同じ。

Linuc LPI-Japanが運営。日本向け。

LPIC カナダのLPI組織。国際標準の資格。

65〜75％程度が合格ライン。

値段：15,000円（Essentialsは10,000）　但し101と102はそれぞれ購入が必要なので、計3万円。