厳選 Unix コマンド v1.0.0

— Selected Unix Commands for Beginners —

［著］公立千歳科学技術大学 IT インフラ部

2024 年 2 月 29 日　v1.0.0

■免責

本書は情報の提供のみを目的としています。
本書の内容を実行・適用・運用したことで何が起きようとも、それは実行・適用・運用した人自身の責任であり、著者や関係者はいかなる責任も負いません。

■商標

本書に登場するシステム名や製品名は、関係各社の商標または登録商標です。
また本書では、™、®、© などのマークは省略しています。

まえがき / はじめに

IT インフラ修行中の三年生が製作した「厳選 Unix コマンド」本をお届けします。執筆者一同の考える「おすすめ」かつ「必須」Unix コマンド群です。

この書籍の配布物には、電子版と印刷版があります。どちらも、次の URL からダウンロードできます。

`https://selected-unix-commands.techbooks.fml.org/`

学内の方は、印刷版を無料配布しているので手に取ってみてください。

本書の対象読者

おおまかには Unix オペレーティングシステムのコマンド操作の初心者が対象ですが、授業の構成を考えると、情報システム工学科 3 年春学期に副読本として用いると最も効果的と考えています。

演習授業のおともに、ご活用ください

問い合わせ先

メール: infra-club@cist.fml.org

学内のかたは、H205 に遊びにきてくれると歓迎されます

感想・意見等をお待ちしております

目次

5

第 1 章
厳選 Unix コマンド

1.1 apt

パッケージ管理用のコマンド。

実行例

```
なし
```

実行結果

```
なし
```

♣ オプション一覧

update

パッケージ情報をダウンロードするために使用される。

実行例

```
sudo apt update
```

実行結果

```
ヒット:1 http://archive.ubuntulinux.jp/ubuntu jammy InRelease
ヒット:2 http://archive.ubuntulinux.jp/ubuntu-ja-non-free jammy InRelease
~中略~
パッケージリストを読み込んでいます... 完了
依存関係ツリーを作成しています... 完了
状態情報を読み取っています... 完了
```

```
  アップグレードできるパッケージが 11 個あります。表示するには 'apt list --upgradable' を実行してく
›ださい。
```

upgrade

システムに現在インストール済みのすべてのパッケージで利用可能なアップグレードをインストールするために使用される。依存関係を満たすために必要な場合は新しいパッケージがインストールされるが、既存のパッケージが削除されることはない。

実行例

```
  sudo apt upgrade
```

実行結果

```
  パッケージリストを読み込んでいます... 完了
  依存関係ツリーを作成しています... 完了
  状態情報を読み取っています... 完了
  アップグレードパッケージを検出しています... 完了
  ~略~
```

full-upgrade

アップグレードの機能を実行するが、システム全体をアップグレードするために必要とされる場合には、現在インストール済みのパッケージを削除する。

実行例

```
  sudo apt full-upgrade
```

実行結果

```
  パッケージリストを読み込んでいます... 完了
  依存関係ツリーを作成しています... 完了
  状態情報を読み取っています... 完了
  アップグレードパッケージを検出しています... 完了
  ~略~
```

install もしくは reinstall もしくは remove もしくは purge

指定された 1 つ以上のパッケージに対して要求された処理を実行する。要求された処理は、特定のパッケージに対してパッケージ名にプラス (+) を追加してパッケージのインストールを、マイナス (-) を追加してパッケージの削除を上書きすることができる。パッケージ名にイコール (=) とパッケージのバージョンを続けることで、選択したバージョンのパッケージをインストールすることができる。また、必要な場合には、パッケージの依存関係を満たすリリースからバージョンを選択する。

パッケージの削除はパッケージの全データを削除するが、削除の事故に備えて、通常は隠れている小さな (修正された) ユーザ設定ファイルを残す。問題が発生した際

は、誤って削除したパッケージのインストール要求を発行すると、以前のようにその機能を復元する。一方、purge を呼び出すことで、既に削除したパッケージの残されたデータを削除することができる。

実行例

```
sudo apt reinstall nano
```

実行結果

```
パッケージリストを読み込んでいます... 完了
依存関係ツリーを作成しています... 完了
状態情報を読み取っています... 完了
アップグレード: 0 個、新規インストール: 0 個、再インストール: 1 個、削除: 0 個、保留: 9 個。
280 kB のアーカイブを取得する必要があります。
この操作後に追加で 0 B のディスク容量が消費されます。
取得:1 http://jp.archive.ubuntu.com/ubuntu jammy/main amd64 nano amd64 6.2-1 [280 kB]
280 kB を 2秒 で取得しました (182 kB/s)
(データベースを読み込んでいます ... 現在 252595 個のファイルとディレクトリがインストールされています。)
.../archives/nano_6.2-1_amd64.deb を展開する準備をしています ...
nano (6.2-1) で (6.2-1 に) 上書き展開しています ...
nano (6.2-1) を設定しています ...
man-db (2.10.2-1) のトリガを処理しています ...
install-info (6.8-4build1) のトリガを処理しています ...
```

autoremove

他のパッケージの依存関係を満たすために自動的にインストールされた後、依存関係の変更あるいは必要としていたパッケージが削除されたことでもう必要なくなったパッケージの削除に使用する。手動でインストールされたパッケージは、自動削除のために提案されない。

実行例

```
sudo apt autoremove
```

実行結果

```
パッケージリストを読み込んでいます... 完了
依存関係ツリーを作成しています... 完了
状態情報を読み取っています... 完了
以下のパッケージは「削除」されます:
  docker-scan-plugin libflashrom1 libftdi1-2 libllvm13
アップグレード: 0 個、新規インストール: 0 個、削除: 4 個、保留: 9 個。
この操作後に 113 MB のディスク容量が解放されます。
続行しますか? [Y/n] y
(データベースを読み込んでいます ... 現在 252615 個のファイルとディレクトリがインストールされています。)
docker-scan-plugin (0.23.0~ubuntu-jammy) を削除しています ...
libflashrom1:amd64 (1.2-5build1) を削除しています ...
libftdi1-2:amd64 (1.5-5build3) を削除しています ...
libllvm13:amd64 (1:13.0.1-2ubuntu2.2) を削除しています ...
libc-bin (2.35-0ubuntu3.5) のトリガを処理しています ...
```

search

指定したパッケージの内容と、その機能を表示する。

実行例

```
sudo apt search nano
```

実行結果

```
ソート中... 完了
全文検索... 完了
alpine-pico/jammy 2.25+dfsg1-1build1 amd64
  Simple text editor from Alpine, a text-based email client

arduino-core-avr/jammy,jammy 1.8.4+dfsg1-1 all
  Arduino Core for AVR microcontroller

bornagain/jammy 1.19.0-3build2 amd64
  Simulate and fit X-ray and neutron GISAS -- binary

bornagain-doc/jammy,jammy 1.19.0-3build2 all
  Simulate and fit X-ray and neutron GISAS -- doc
```

show

指定されたパッケージに関する情報を表示する。依存関係、インストールおよびダウンロードサイズ、パッケージが入手可能な取得元、パッケージの内容の説明などを含む。パッケージの削除をさせる前や、インストールする新しいパッケージを検索する際に、この情報を見て参考にすることができる。

実行例

```
sudo apt show nano
```

実行結果

```
Package: nano
Version: 6.2-1
Priority: standard
Section: editors
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Jordi Mallach <jordi@debian.org>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 881 kB
Depends: libc6 (>= 2.34), libncursesw6 (>= 6), libtinfo6 (>= 6)
Suggests: hunspell
Conflicts: pico
Breaks: nano-tiny (<< 2.8.6-2)
Replaces: nano-tiny (<< 2.8.6-2), pico
Homepage: https://www.nano-editor.org/
Task: standard
Download-Size: 280 kB
APT-Manual-Installed: yes
APT-Sources: http://jp.archive.ubuntu.com/ubuntu jammy/main amd64 Packages
Description: Pico にヒントを得て作られた、コンパクトで使いやすいテキストエディタ
 GNU nano は使いやすいテキストエディタで、当初は Pico の代替品として設計され ました。Pico とは、か›
›つて non-free
```

```
   だったメーラパッケージ Pine の ncurses ベースのエディタです (現在 Pine 自体は、Apache ライセンス
›で Alpine
    という名 前で入手できます)。
    .
    However, GNU nano also implements many features missing in Pico, including:
     - undo/redo
     - line numbering
     - syntax coloring
     - soft-wrapping of overlong lines
     - selecting text by holding Shift
     - interactive search and replace (with regular expression support)
     - a go-to line (and column) command
     - support for multiple file buffers
     - auto-indentation
     - tab completion of filenames and search terms
     - toggling features while running
     - and full internationalization support
```

list

一定の基準を満たすパッケージのリストを表示することができる。

実行例

```
sudo apt list nano
```

実行結果

```
一覧表示... 完了
nano/jammy,now 6.2-1 amd64 [インストール済み]
```

1.2　cat

ファイルの内容を確認するコマンド。複数のファイルの内容を確認することも可能。設定ファイルをエディターで開くと壊れる恐れがあるため、内容をまず確認すると安全。

実行例

```
cat sample.txt
```

実行結果

```
red


blue
yellow
```

♣ オプション一覧

-n

行番号を表示する。

実行例

```
cat -n sample.txt
```

実行結果

```
1 red
2
3
4 blue
5 yellow
```

-b

空行を飛ばして行番号を表示する。

実行例

```
cat -b sample.txt
```

実行結果

```
1 red


2 blue
3 yellow
```

- **-s**

連続した空行を 1 行にする。

実行例

```
cat -s sample.txt
```

実行結果

```
red

blue
yellow
```

1.3 cd

cd: cd 作業ディレクトリを変更します。

現在のディレクトリを DIR に変更します。デフォルトの DIR は HOME シェル変数の値です。

変数 CDPATH は、DIR を含むディレクトリの検索パスを定義します。CDPATH 内の代替ディレクトリ名はコロン（:）で区切られます。空のディレクトリ名は現在のディレクトリと同じです。DIR がスラッシュ（/）で始まる場合、CDPATH は使用されません。

ディレクトリが見つからない場合、シェルオプション「cdable_vars」が設定されている場合、単語は変数名と見なされます。その変数に値がある場合、その値が DIR として使用されます。

オプション:

-L シンボリックリンクを強制的にたどる：..’ のインスタンスを処理した後、DIR 内のシンボリックリンクを解決します。

-P シンボリックリンクをたどらずに物理的なディレクトリ構造を使用する：..’ のインスタンスを処理する前に DIR 内のシンボリックリンクを解決します。

-e -P オプションが指定されており、現在の作業ディレクトリを正常に特定できない場合、終了ステータスを非ゼロで終了します。

-@ サポートされているシステムでは、拡張属性を持つファイルをファイル属性を含むディレクトリとして表示します。

デフォルトでは、シンボリックリンクをたどります。‘..’ は、直前のパス名コンポーネントをスラッシュまたは DIR の先頭まで削除して処理されます。

終了ステータス: ディレクトリが変更されると 0 を返し、-P が使用される場合は $PWD が正常に設定された場合に成功します。それ以外の場合は非ゼロを返します。

実行例.1

cd を単体で使用した場合、ユーザのホームディレクトリへ移動

```
cd
```

実行結果.1

```
$ pwd  //実行前の現在地を確認
home/user/Prmn

$ cd //実行
```

```
$ pwd //実行後の現在地を確認
homo/user  //ユーザのホームディレクトリに移動したことがわかる
```

実行例.2

cd の後ろにディレクトリ名を指定するとそのディレクトリへ移動する。

```
cd ディレクトリ名
```

実行結果.2

```
$ pwd  //実行前の現在地を確認
home/user
$ ls //現在地にあるディレクトリを確認
Prmn test
$ cd Prmn // 実行
$ pwd //実行後の現在地を確認
home/user/Prmn //Prmnディレクトリに移動したことがわかる
```

実行例.3

1 つ前のディレクトリへ戻る

```
cd ..
```

実行結果.3

```
$ pwd //実行前の現在地を確認
home/user/Prmn/test
$ cd ..  //実行
$ pwd //実行後の現在地を確認
home/user/Prmn //1つ前のディレクトリへ戻ったことが確認できる
```

実行例.4

cd 単体でコマンドを実行した場合と同じ挙動をする。主に複雑なディレクトリ間を移動する際に使用.。

```
cd ~/Prmm
```

実行結果.4

```
$ pwd
home/user/prac
$ cd ~/Prmn //一度ユーザのホームディレクトリへ移動してからPrmnへ移動
$ pwd
home/user/Prmn //Prmnディレクトリへ移動していることを確認
```

実行例.5

前に居たディレクトリへ移動する。

```
cd -
```

実行結果.5

```
$ pwd //移動する前のディレクトリを表示
home/user/Prmn
$ cd /etc  // etcへ移動
$ pwd
/etc
$ cd -  // 実行
$ pwd
home/user/Prmn //元居た場所へ戻ったことを確認できる
```

♣ オプション一覧

-L

移動先のディレクトリがシンボリックリンクだった場合、シンボリックリンクに移動するオプション。シンボリックリンクとは別のファイルやディレクトリへの参照を作成するための特殊なファイル。

実行例

```
cd -L tmp
```

実行結果

```
$ pwd
home/user/Prmn //現在地のディレクトリを確認
$ ln -s tmp/ tmp //現在のディレクトリにシンボリックリンクを作成。
$ ls
tmp  //ルート直下のtmpが参照されている
$ cd -L tmp //参照先のtmpへ移動
$ pwd
home/user/Prmn/tmp  /
```

-P

移動先がシンボリックリンクだったらシンボリックリンクのターゲット飛ぶためのオプション。つまり、参照したディレクトリの元のディレクトリへ移動する。

実行例

```
cd -P tmp
```

実行結果

```
$ pwd
home/user/Prmn //現在地のディレクトリを確認
$ ln -s tmp/ tmp //現在のディレクトリにシンボリックリンクを作成。
```

```
$ ls
tmp  //ルート直下のtmpが参照されている
$ cd -P tmp //参照先のtmpへ移動
$ pwd
/tmp  //ルート直下のtmpディレクトリへ移動していることがわかる。
```

1.4 chmod

ファイルやディレクトリのパーミッション（アクセス権限）を変更するためのコマンド。これにより、ファイルやディレクトリに対する読み取り、書き込み、実行などの権限を設定可能。

モードには、数値を用いた絶対値指定と、シンボルによる指定がある。 数値指定では、以下の値を足し合わせた 8 進数を用いる。

4000 (setuid ビット) ファイルにこのビットがセットされている場合、そのファイルを実行したときにそのプロセスは、ファイルの所有者の権限ではなくファイルの所有者として指定されているユーザーの権限を使用する。

2000 (setgid ビット) ファイルにこのビットがセットされている場合、そのファイルを実行したときにそのプロセスは、ファイルの所有者の権限ではなく、ファイルの所有者として指定されているグループの権限を使用する。一般的には、グループが共有データにアクセスできるようにするために使用される。

1000 (sticky ビット) ディレクトリにこのビットがセットされている場合、そのディレクトリ内のファイルは、そのディレクトリを作成したユーザー、またはファイルの所有者以外は削除できない。通常は、一時的なディレクトリなどで使用される。

0400 所有者の読み込みを許可。

0200 所有者の書き込みを許可。

0100 ファイルの場合、所有者の実行を許可。 ディレクトリの場合、所有者の検索を許可。

0040 グループのメンバの読み込みを許可。

0020 グループのメンバの書き込みを許可。

0010 ファイルの場合、グループのメンバの実行を許可。 ディレクトリの場合、グ

ループのメンバの検索を許可。

0004 他者の読み込みを許可。

0002 他者の書き込みを許可。

0001 ファイルの場合、他者の実行を許可。 ディレクトリの場合、他者の検索を許可。

例えば、所有者に読み込み・書き込み・実行を許可し、 グループのメンバに読み込み・実行を許可し、 他者に読み込み・実行を許可し、 set-uid と set-gid を指定しない絶対値指定のモードは、 755 (400+200+100+040+010+004+001) となる。

who シンボルの u", g", o" はそれぞれユーザ、グループ、それ以外に 相当します。 "a" シンボルは ugo" を指定した場合と同じになる。

perm シンボルはモードの各ビットを以下のように表現する。

r 読み込み許可ビット

s 実行時 setuid および実行時 setgid ビット

t sticky ビット

w 書き込み許可ビット

x 実行 もしくは 検索 許可ビット

X 対象がディレクトリであるか、変更前のモードで誰かの実行 もしくは 検索許可ビット が立っている場合に、実行 もしくは 検索許可ビットがセットされる。

perm シンボルでの X" の指定は、 op シンボルを +" で連結する時のみ意味があり、他の場合は無視する。

u 元の、ファイルの所有者許可ビット

g 元の、ファイルのグループ許可ビット

o 元の、ファイルの所有者とグループ以外の許可ビット

実行例

パーミッションが 0664(rw-rw-r--) の text.txt に 0644(rw-r--r--) を設定する。

```
$ chmod -v 644 text.txt
$ chmod -v u=rw,g=r,o=r text.txt
'test.txt' モードを 0664 (rw-rw-r--) から 0644 (rw-r--r--) へ変更しました
```

読み取り・書き込み権限を所有者に、読み取り権限をグループとその他のユーザーに与える。

♣ オプション一覧

-v もしくは --verbose

処理されたファイルごとに診断結果を出力する

実行例

```
$ chmod -v 614 text.txt
```

実行結果

```
'test.txt' モードを 0644 (rw-r--r--) から 0614 (rw---xr--) へ変更しました
```

-c もしくは --changes

verbose と同様に、変更があったときだけ報告する。

実行例

```
$ chmod -c 614 text.txt
```

実行結果

```
'test.txt' モードを 0644 (rw-r--r--) から 0614 (rw---xr--) へ変更しました
```

-f もしくは --silent もしくは --quiet

ほとんどのエラーメッセージを表示しない

実行例

```
$ chmod -f 644 text.txt
```

実行結果

エラーメッセージ例

- 存在しないファイルやディレクトリを対象にした場合

```
$ chmod 644 t.txt
chmod: 't.txt' にアクセスできません: そのようなファイルやディレクトリはありません
```

- パーミッションが不正な場合

```
$ chmod 999 text.txt
chmod: 無効なモード: '999'
```

- 権限がない場合

```
$ chmod 777 /etc/apt
chmod: '/etc/apt' のパーミッションを変更しています: 許可されていない操作です
```

--no-preserve-root

ルートディレクトリ ('/') の保護を無効にする。

通常、Unix 系のオペレーティングシステムでは、ルートディレクトリの変更や削除を防ぐためにセキュリティ対策が施されている。しかし、このオプションを使用することで、ルートディレクトリに対する操作が制限なしに行えるようになる。

実行例

```
$ chmod -v --no-preserve-root 777 /etc/apt
```

実行結果

```
'/etc/apt' のモードを 0644 (rw-r--r--) から 0777 (rwxrwxrwx) へ変更されました
```

--preserve-root

ルートディレクトリ ('/') の保護を有効にする。

実行例

```
$ chmod -v --preserve-root 777 /etc/apt
```

実行結果

```
chmod: '/etc/apt' のパーミッションを変更しています: 許可されていない操作です
'/etc/apt' のモードを 0755 (rwxr-xr-x) から 0777 (rwxrwxrwx) へ変更できませんでした
```

--reference=RFILE

指定したファイルのパーミッションをコピーして、既存のファイルやディレクトリに設定する。

実行例

```
$ chmod --reference=text.txt new_text.txt
```

実行結果

```
'new_text.txt' のモードを 0644 (rw-r--r--) から 0614 (rw---xr--)  へ変更しました
```

-R もしくは --recursive

ファイルとディレクトリを再帰的に変更する

```
.
├──chmod
│    ├──new_text.txt
│    ├──text/txt
```

上記のディレクトリ構造に対して実行する

実行例

```
$ chmod -v -R 777 chmod
```

実行結果

```
'chmod' のモードを 0775 (rwxrwxr-x) から 0777 (rwxrwxrwx) へ変更しました
'chmod/text.txt' のモードを 0641 (rw-r---x) から 0777 (rwxrwxrwx) へ変更しました
'chmod/new_text.txt' のモードを 0641 (rw-r----x) から 0777 (rwxrwxrwx) へ変更しました
```

1.5　find

指定された PATH を始点とするディレクトリツリーを探索し、与えられた式を、優先規則に従いつつ、左から右へ評価することによって検索を行う。結果が確定すると (例えば、and 演算なら左辺が偽になった時点で、or 演算なら左辺が真になった時点で)、find は検査の対象を次のファイル名に移す。後述するオプションのうち、整数を引数として指定するオプションにおいて、n を整数とすると、+n とすると n を超える数であることを意味し、-n とすると n 未満であることを意味する。n とするとぴったり n であることを意味する。

実行例

```
なし
```

実行結果

```
なし
```

♣ オプション一覧

-name

ファイルやディレクトリのベースネーム (パスから最後の要素だけを残し、先行するディレクトリを取り去ったもの) が、マッチすれば出力する。

実行例

```
find /home/user/ -name a.txt
```

実行結果

```
/home/user/a.txt
```

-iname

-name と同じだが、大文字小文字を区別しない。

実行例

```
find /home/user/ -iname a.txt
```

実行結果

```
/home/user/A.txt
/home/user/a.txt
```

-type f

タイプがファイルであれば出力する。

実行例

```
find /home/user/ -type f
```

実行結果

```
~略~
/home/user/project/upload_and_run.py
/home/user/project/index.html
/home/user/.wget-hsts
/home/user/.profile
```

-type d

タイプがディレクトリであれば出力する。

実行例

```
find /home/user/ -type d
```

実行結果

```
~略~
/home/user/project/docker
/home/user/project/upload
/home/user/.vnc
```

-size

ファイルの容量が指定した値であれば出力する。次の接尾辞を使用できる。

b ブロック。1 ブロックは 512 バイトである。接尾辞を使用しない場合のデフォルトである。

c バイト。

w ワード。1 ワードは 2 バイト。

k キロバイト。1 キロバイトは 1024 バイト。

M メガバイト。1 メガバイトは 1048576 バイト。

G ギガバイト。1 ギガバイトは 1073741824 バイト。

実行例

```
find /home/user/ -size 10c
```

実行結果

```
/home/user/snap/firefox/common/.mozilla/firefox/Crash Reports/InstallTime20230512012512
/home/user/snap/firefox/common/.mozilla/firefox/Crash Reports/InstallTime20230414190624
/home/user/snap/firefox/common/.mozilla/firefox/Crash Reports/InstallTime20230710222611
~略~
```

-newer

指定したファイルの内容更新日時よりも、内容更新日時が最近であれば出力する。

実行例

```
find /home/user/ -newer a.txt
```

実行結果

```
/home/user/
/home/user/A.txt
/home/user/.cache/tracker3/files
~略~
```

-anewer

指定したファイルの内容更新日時よりも、最終アクセス日時が最近であれば出力する。

実行例

```
find /home/user/ -anewer a.txt
```

実行結果

```
/home/user/
/home/user/A.txt
/home/user/a.txt
~略~
```

-cnewer

指定したファイルの内容更新日時よりも、最終ステータス変更日時が最近であれば出力する。

実行例

```
find /home/user/ -cnewer a.txt
```

実行結果

```
/home/user/
/home/user/test
/home/user/test/t.txt
~略~
```

-maxdepth

指定されたパスから最大何段階下のディレクトリまで探索するかを指定する。実行例では、/home/user/test/test1 というディレクトリ構造になっており、test ディレクトリに t.txt、test1 ディレクトリに T.txt というファイルを配置したマシンを想定し、引数を変えて実行した。

実行例

```
find /home/user/ -maxdepth 2 -iname t.txt
```

実行結果

```
/home/user/test/t.txt
```

実行例

```
find /home/user/ -maxdepth 3 -iname t.txt
```

実行結果

```
/home/user/test/t.txt
/home/user/test/test1/T.txt
```

-mindepth

指定されたパスから少なくとも何段階下のディレクトリまで探索しないかを指定する。実行例では、/home/user/test/test1 というディレクトリ構造になっており、

test ディレクトリに t.txt、test1 ディレクトリに T.txt というファイルを配置したマシンを想定し、引数を変えて実行した。

実行例

```
find /home/user/ -mindepth 3 -iname t.txt
```

実行結果

```
/home/user/test/test1/T.txt
```

実行例

```
find /home/user/ -mindepth 2 -iname t.txt
```

実行結果

```
/home/user/test/t.txt
/home/user/test/test1/T.txt
```

-daystart

後述する-mtime、-mmin、-atime、-amin、-ctime、-cmin において、デフォルトでは時間を計算する際の基準をコマンド実行時ピッタリの日時となるが、このオプションを先に指定することで、時間を計算する際の基準をコマンド実行時当日の 0 時とする。

実行例

```
find /home/user/ -daystart -ctime -1
```

実行結果

```
/home/user/
/home/user/test
/home/user/test/t.txt
~略~
```

-mtime

ファイルの最終内容更新日時が、指定した数字"日"であれば出力する。

実行例

```
find /home/user/ -mtime 0
```

実行結果

```
   ~略~
   /home/user/.cache/tracker3/files/meta.db-wal
   /home/user/.cache/tracker3/files/http%3A%2F%2Ftracker.api.gnome.org%2Fontology%2Fv3%2Ftr›
›acker%23Documents.db-shm
   /home/user/.cache/update-manager-core/meta-release-lts
```

-mmin

ファイルの最終内容更新日時が、指定した数字"分"であれば出力する。

実行例

```
   find /home/user/ -mmin -30
```

実行結果

```
   ~略~
   /home/user/.cache/tracker3/files/last-crawl.txt
   /home/user/.cache/tracker3/files/http%3A%2F%2Ftracker.api.gnome.org%2Fontology%2Fv3%2Ftr›
›acker%23FileSystem.db-wal
   /home/user/.cache/tracker3/files/meta.db-wal
```

-atime

ファイルの最終アクセス日時が、指定した数字"日"であれば出力する。

実行例

```
   find /home/user/ -atime 0
```

実行結果

```
   ~略~
   /home/user/project/upload
   /home/user/.profile
   /home/user/.vnc
```

-amin

ファイルの最終アクセス日時が、指定した数字"分"であれば出力する。

実行例

```
   find /home/user/ -amin -30
```

実行結果

```
   ~略~
   /home/user/.cache/tracker3/files/last-crawl.txt
   /home/user/.cache/tracker3/files/http%3A%2F%2Ftracker.api.gnome.org%2Fontology%2Fv3%2Ftr›
›acker%23FileSystem.db-wal
   /home/user/.cache/tracker3/files/meta.db-wal
```

-ctime

ファイルの最終ステータス変更日時が、指定した数字"日"であれば出力する。

実行例

```
  find /home/user/ -ctime 0
```

実行結果

```
  ~略~
  /home/user/.cache/tracker3/files/meta.db-wal
  /home/user/.cache/tracker3/files/http%3A%2F%2Ftracker.api.gnome.org%2Fontology%2Fv3%2Ftr›
›acker%23Documents.db-shm
  /home/user/.cache/update-manager-core/meta-release-lts
```

-cmin

ファイルの最終ステータス変更日時が、指定した数字"分"であれば出力する。

実行例

```
  find /home/user/ -cmin -30
```

実行結果

```
  ~略~
  /home/user/.cache/tracker3/files/last-crawl.txt
  /home/user/.cache/tracker3/files/http%3A%2F%2Ftracker.api.gnome.org%2Fontology%2Fv3%2Ftr›
›acker%23FileSystem.db-wal
  /home/user/.cache/tracker3/files/http%3A%2F%2Ftracker.api.gnome.org%2Fontology%2Fv3%2Ftr›
›acker%23Documents.db-shm
```

-empty

空のファイルかディレクトリであれば出力する。

実行例

```
  find /home/user/ -empty
```

実行結果

```
  /home/user/.cache/sessions
  /home/user/.cache/motd.legal-displayed
  /home/user/project/upload
  ~略~
```

-regex

指定した正規表現と一致した場合に出力する。この場合の一致とは、PATH も含めたファイル名全体に対する一致である。例えば、./fubar3 という名前のファイルに

一致させるために、正規表現".*bar."や".*b.*3"は使用できるが、"f.*r3"は使用できない。

実行例

```
find /home/user/ -regex ".*t2.txt"
```

実行結果

```
~略~
/home/user/ダウンロード/go1.20.2.linux-amd64/go/src/go/doc/comment/testdata/text2.txt
/home/user/ダウンロード/go1.20.2.linux-amd64/go/src/go/doc/comment/testdata/list2.txt
/home/user/output2.txt
```

-iregex

-regex と同じだが、大文字小文字を区別しない。

実行例

```
find /home/user/ -iregex .*T2.txt
```

実行結果

```
~略~
/home/user/ダウンロード/go1.20.2.linux-amd64/go/src/go/doc/comment/testdata/text2.txt
/home/user/ダウンロード/go1.20.2.linux-amd64/go/src/go/doc/comment/testdata/list2.txt
/home/user/output2.txt
```

-perm

ファイルの権限が一致した場合に出力する。権限の前に-を付けると、0 以外の権限が全て与えられているファイルを出力する。権限の前に/を付けると、0 以外の権限のどれかが与えられているファイルを出力する。

実行例

```
find /home/user/test/ -maxdepth 1 -perm 777
```

実行結果

```
/home/user/test/t1.txt
```

実行例

```
find /home/user/test/ -maxdepth 1 -perm -707
```

実行結果

```
/home/user/test/t1.txt
```

実行例

```
find /home/user/test/ -maxdepth 1 -perm /706
```

実行結果

```
/home/user/test/t.txt
/home/user/test/t1.txt
/home/user/test/test1
```

-user

ファイルの所有者が指定したユーザ名と一致した場合に出力する。

実行例

```
find /home/user/ -user root
```

実行結果

```
~略~
/home/user/project/docker/Dockerfile
/home/user/project/upload_and_run.py
/home/user/project/index.html
```

-uid

ファイルの所有者が指定したユーザ ID と一致した場合に出力する。

実行例

```
find /home/user/ -uid 0
```

実行結果

```
~略~
/home/user/project/docker/Dockerfile
/home/user/project/upload_and_run.py
/home/user/project/index.html
```

-group

ファイルの属するグループが指定したグループ名と一致した場合に出力する。

実行例

```
find /home/user/ -group root
```

実行結果

```
~略~
/home/user/project/docker/Dockerfile
/home/user/project/upload_and_run.py
/home/user/project/index.html
```

-gid

ファイルの属するグループ ID が指定したグループ ID と一致した場合に出力する。

実行例

```
find /home/user/ -group 0
```

実行結果

```
~略~
/home/user/project/docker/Dockerfile
/home/user/project/upload_and_run.py
/home/user/project/index.html
```

-delete

探索したファイルを消去する。実行結果では、実行前と実行後に ls -la を実行した出力を記述する。

実行例

```
find /home/user/test/ -maxdepth 1 -perm 777 -delete
```

実行結果

```
実行前
drwxrwxr-x  3 user user 4096  1月 30 04:11 .
drwxr-x--- 29 user user 4096  1月 30 02:28 ..
-rw-rw-rw-  1 user user    0  1月 30 02:28 t.txt
drwxrwxr-x  2 user user 4096  1月 30 02:29 test1
-rwxrwxrwx  1 root  root     0  1月 30 04:11 test1.txt

実行後
drwxrwxr-x  3 user user 4096  1月 30 04:12 .
drwxr-x--- 29 user user 4096  1月 30 02:28 ..
-rw-rw-rw-  1 user user    0  1月 30 02:28 t.txt
drwxrwxr-x  2 user user 4096  1月 30 02:29 test1
```

-exec

探索したファイルに対して、指定したコマンドを実行する。-exec コマンド {} \; のように記述する。{}は探索した全てのファイル名に置き換えられる。コマンドの終わりには\; が必要。実行結果では、実行前と実行後に ls -la を実行した出力を記述する。

実行例

```
find /home/user/test/ -perm 777 -exec sudo chmod 666 {} \;
```

実行結果

```
実行前
drwxrwxr-x  3 user user 4096  1月 30 04:23 .
drwxr-x--- 29 user user 4096  1月 30 04:21 ..
-rwxrwxrwx  1 user user    0  1月 30 02:28 t.txt
-rwxrwxrwx  1 root  root    15  1月 30 04:14 test.py
drwxrwxr-x  2 user user 4096  1月 30 02:29 test1
-rwxrwxrwx  1 root  root    22  1月 30 04:23 test1.py

実行後
drwxrwxr-x  3 user user 4096  1月 30 04:23 .
drwxr-x--- 29 user user 4096  1月 30 04:21 ..
-rw-rw-rw-  1 user user    0  1月 30 02:28 t.txt
-rw-rw-rw-  1 root  root    15  1月 30 04:14 test.py
drwxrwxr-x  2 user user 4096  1月 30 02:29 test1
-rw-rw-rw-  1 root  root    22  1月 30 04:23 test1.py
```

-ok

-exec と似ているが、まずユーザに問い合わせを行う。ユーザーが同意すれば、コマンドを実行する。同意しなければ、何もしない。hello と標準出力に出力する test.py と、good morning と標準出力に出力する test1.py を作り、実行した。

実行例

```
find /home/user/test/ -regex .*.py -ok python3 {} \;
```

実行結果

```
< python3 ... /home/user/test/test.py > ? yes
hello
< python3 ... /home/user/test/test1.py > ? yes
good morning
```

-print

探索したファイルの PATH 付きのファイル名を標準出力に表示し、各ファイル名の後ろに改行文字を付ける。

実行例

```
find /home/user/test/ -regex .*.py -print
```

実行結果

```
/home/user/test/test.py
/home/user/test/test1.py
```

-print0

探索したファイルの PATH 付きのファイル名を標準出力に表示し、各ファイル名の後ろにヌル文字を付ける。

実行例

```
find /home/user/test/ -regex .*.py -print0
```

実行結果

```
/home/user/test/test.py/home/user/test/test1.py
```

-fprint

探索したファイルの PATH 付きのファイル名を指定したファイルに出力し、各ファイル名の後ろに改行文字を付ける。実行結果では出力したファイルの内容を記述する。

実行例

```
find /home/user/test/ -regex .*.py -fprint output.txt
```

実行結果

```
/home/user/test/test.py
/home/user/test/test1.py
```

-fprint0

探索したファイルの PATH 付きのファイル名を指定したファイルに出力し、各ファイル名の後ろにヌル文字を付ける。実行結果では出力したファイルの内容を記述する。

実行例

```
find /home/user/test/ -regex .*.py -fprint0 output1.txt
```

実行結果

```
/home/user/test/test.py^@/home/user/test/test1.py^@
```

-a もしくは -and

これで結合された条件式は、and 結合と解釈される。条件式の間に単にスペースを開けただけの場合も、and 結合と解釈される。

実行例

```
find /home/user/test/ -regex .*.txt -and -perm 666
```

実行結果

```
/home/user/test/t.txt
```

-o もしくは -or

これで結合された条件式は、or 結合と解釈される。

実行例

```
find /home/user/test/ -regex .*.txt -or -perm 666
```

実行結果

```
/home/user/test/test.py
/home/user/test/output1.txt
/home/user/test/t.txt
/home/user/test/output.txt
/home/user/test/test1/T.txt
/home/user/test/test1.py
```

-not

これの後に記述された条件式が一致しない、という条件で探索する。

実行例

```
find /home/user/test/ -not -type f
```

実行結果

```
/home/user/test/
/home/user/test/test1
```

1.6　ls

デフォルトでは現在のディレクトリの中にあるファイルに関する情報をリストアップする。-cftuvSUX または --sort のいずれも指定されていない場合は、検索結果をアルファベット順にソートして出力する。

実行例

```
ls
```

実行結果

```
auto_create_md.py  ls.md  man.md  ping.md
```

♣ オプション一覧

-a もしくは --all

. から始まるファイル（ドットファイル）も表示する。

実行例

```
  ls -a
```

実行結果

```
  .  ..  .git  auto_create_md.py  ls.md  man.md  ping.md
```

-A もしくは --almost-all

どこのディレクトリにもあるドットファイルである、「.」および「..」をリストに含めないようにする。

実行例

```
  ls -A
```

実行結果

```
  .git  auto_create_md.py  ls.md  man.md  ping.md
```

--author

-l と併用することで各ファイルの作者を表示できる。

実行例

```
  ls -l --author
```

実行結果

```
  total 36
  -rw-r--r-- 1 author user user  4519 Dec 23 15:10 auto_create_md.py
  -rw-rw-r-- 1 author user user 11692 Dec 23 15:10 ls.md
  -rw-rw-r-- 1 author user user 10558 Dec 10 03:49 man.md
  -rw-rw-r-- 1 author user user  1024 Dec 9 04:23 ping.md
```

上の例でわかりやすく author としているところに作者のアカウント名が表示される。

--color[=WHEN]

ファイルの種類を区別するための出力の色をつけるかどうかを変更できるオプション。

オプションを付与した場合にのみ作用され、常に変更できるわけではない。常に変更する場合は LS_COLORS という環境変数により設定を変更できる。その設定には dircolors コマンドを使用する必要がある。（ここでは記載しない。）

デフォルトの ls では引数 (=WHEN) に always が渡された時と同じく、色がついている状態。以下の値を引数に取ることができる。

- always：デフォルトの挙動。色がつく。

- auto：標準出力がターミナルに接続できている場合にのみ、色がつく。ssh などにより色がつかなくなっているのはこのオプションによるもの。

- never：色がつかない。

実行例

```
ls --colors=never
```

実行結果

```
auto_create_md.py  ls.md  man.md  ping.md
```

この書式上、色が区別できる状態の実行結果を見せることはできないが、上記の実行例ではかならず何も着色されていない状態で出力される。

-f

ソートせず、-aU オプションを有効にし、-ls --color オプションを無効にした出力をするオプション。つまり、詳細情報ではない隠しファイルをふくむすべてのファイルを見つけた順で色なしで表示するオプション。

実行例

```
ls -f
```

実行結果

```
..  .  man.md  create_hoge_files.sh  ls.md  auto_create_md.py  ping.md
```

--format=WORD

引数 (WORD) に以下の単語を渡すことで指定できる出力形式で、出力できるオ

プション。

- across(=-x オプション)：行で出力する。
- commas(=-m オプション)：カンマ区切りで出力する。
- horizontal(=-x オプション)：行で出力する。
- long(=-l オプション)：詳細情報まで出力する。
- single-column(=-1 オプション)：1 行につき 1 ファイルずつ出力する。
- verbose(=-l オプション)：詳細情報まで出力する。
- vertical(=-C オプション)：詳細情報まで出力する。

実行例

```
ls --format=across
```

実行結果

```
各コマンドの出力先を参照。
```

--group-directories-first

ディレクトリをファイルより先に表示するオプション。

--sort オプションと組み合わせて使用することもできるが、--sort=none もしくは -U と併用してしまうと先にディレクトリが表示されないようになってしまう。

実行例

```
ls --group-directories-first
```

実行結果

```
// dir = ディレクトリ
auto_create_md.py  create_hoge_files.sh  dir  ls.md  man.md  ping.md
```

実行結果

```
// dir = ディレクトリ
dir  auto_create_md.py  create_hoge_files.sh  ls.md  man.md  ping.md
```

--hide=PATTERN

シェルで利用できる正規表現を引数 (=PATTERN) に渡すことで指定した正規表現にマッチするファイル名を表示しないで出力するオプション。-a オプションなどとの併用ではマッチしたものも表示されてしまう。

実行例

```
ls --hide='*.md'
```

実行結果

```
auto_create_md.py  create_hoge_files.sh  dir  ls.md  man.md  ping.md
```

実行結果

```
auto_create_md.py  create_hoge_files.sh  dir
```

-l

ファイル (ディレクトリ) の詳細情報を表示するオプション。

total にはディレクトリ内のすべてのファイルの*ブロック数の合計が出力される。

(* ブロック：ディスク上の容量のひとまとまりの単位のこと。通常 1 ブロックは 512 もしくは 1024 バイト)

左から、ファイル (ディレクトリ) のパーミッション、そのファイルに対する*ハードリンクの数、ファイルの所有者、所属グループ、ファイルサイズ、最終変更時刻、ファイル (ディレクトリ) 名が表示される。

(* ハードリンク：同一のファイルに対する別名のこと。ハードリンク同士で同じデータブロックを共有するため、どちらかの変更が他方にも反映される。)

実行例

```
ls -l
```

実行結果

```
total 44
-rw-r--r-- 1 user user  4519 Dec 25 15:10 auto_create_md.py
-rwxrwxr-x 1 user user    63 Dec 26 01:27 create_hoge_files.sh
drwxrwxr-x 2 user user  4096 Dec 28 00:20 dir
-rw-rw-r-- 1 user user 11692 Dec 25 15:10 ls.md
-rw-rw-r-- 1 user user 10558 Dec 11 03:49 man.md
-rw-rw-r-- 1 user user  1024 Dec 11 04:23 ping.md
```

-p もしくは --indicator-style=slash

ディレクトリ名の末尾に / が追加されて表示されるオプション。

実行例

```
ls -p
```

実行結果

```
// dir は ディレクトリ
auto_create_md.py  create_hoge_files.sh  dir/  ls.md  man.md  ping.md
```

-Q もしくは --quote-name

表示されるファイル (ディレクトリ) 名がダブルクォーテーションで囲まれて出力されるオプション。

ファイル (ディレクトリ) 名にスペースや特殊文字が含まれているときに使用することが多い。

実行例

```
ls -Q
```

実行結果

```
 auto_create_md.py      dir     man.md               ping.md
create_hoge_files.sh   ls.md  'one'$'\n''two.txt'
```

実行結果

```
"auto_create_md.py"     "dir"    "man.md"        "ping.md"
"create_hoge_files.sh"  "ls.md"  "one\ntwo.txt"
```

-r もしくは --reverse

ファイル (ディレクトリ) のソートの順番を逆にするオプション。

実行例

```
ls -r
```

実行結果

```
 auto_create_md.py      dir     man.md               ping.mdcreate_hoge_files.sh   ls.md
'one'$'\n''two.txt'
```

実行結果

```
ping.md             man.md   dir                    auto_create_md.py
'one'$'\n''two.txt'   ls.md    create_hoge_files.sh
```

-R もしくは --recursive

*サブディレクトリの中身まで再帰的に表示するオプション。

(* サブディレクトリ：現在のディレクトリの下に存在するディレクトリのこと)

実行例

```
ls -R
```

実行結果

```
// dir ディレクトリの中
user@localhost:~/dir$ ls
hoge-hoge.txt
```

実行結果

```
// dirディレクトリの上の階層のディレクトリの中
user@localhost:~$ ls -R
.:
auto_create_md.py      dir     man.md               ping.md
create_hoge_files.sh   ls.md  'one'$'\n''two.txt'

./dir:
hoge-hoge.txt
```

上記例では *カレントディレクトリは . として表示されている。
(* カレントディレクトリ：現在のディレクトリのこと)

-S

ファイルサイズの大きい順にソートして表示されるオプション。

実行例

```
ls -S
```

実行結果

```
total 52
drwxrwxr-x  3 user user  4096 Dec 28 01:52  ./
drwxr-x--- 15 user user  4096 Dec 26 03:48  ../
-rw-r--r--  1 user user  4519 Dec 25 15:10  auto_create_md.py
-rwxrwxr-x  1 user user    63 Dec 26 01:27  create_hoge_files.sh*
drwxrwxr-x  2 user user  4096 Dec 28 02:11  dir/
-rw-rw-r--  1 user user 11692 Dec 25 15:10  ls.md
-rw-rw-r--  1 user user 10558 Dec 11 03:49  man.md
-rw-rw-r--  1 user user     0 Dec 28 01:52 'one'$'\n''two.txt'
-rw-rw-r--  1 user user  1024 Dec 11 04:23  ping.md
```

実行結果

```
// 横でソートされているのを確認するために
// ls -Sx としている

ls.md                  man.md               auto_create_md.py   dir   ping.md
create_hoge_files.sh  'one'$'\n''two.txt'
```

上記の ls -l の例から ls.md, man.md, auto_create_md.py, ... の順にファイルサイズが大きいことが確認でき、ls -Sx の例からその順でソートされて出力されていることが確認できる。

--sort=WORD

出力を引数 (=WORD) にて指定した方法でソートするオプション。WORD には以下の単語を指定できる。

- none (= -U オプション) : ソートしない。ファイルが見つかった順。
- size (= -S オプション) : ファイルサイズが大きい順。
- time (= -t オプション) : 更新時刻順。
- versino (= -v オプション) : バージョン番号順。
- extension (= -X オプション) : 拡張子順。

実行例

```
ls --sort=none
```

実行結果

```
total 44
-rw-r--r-- 1 user user  4519 Dec 25 15:10  auto_create_md.py
-rwxrwxr-x 1 user user    63 Dec 26 01:27  create_hoge_files.sh
drwxrwxr-x 2 user user  4096 Dec 28 02:11  dir
-rw-rw-r-- 1 user user 11692 Dec 25 15:10  ls.md
-rw-rw-r-- 1 user user 10558 Dec 11 03:49  man.md
-rw-rw-r-- 1 user user     0 Dec 28 01:52 'one'$'\n''two.txt'
-rw-rw-r-- 1 user user  1024 Dec 11 04:23  ping.md
```

以下のソートではソートの結果を行で確認するため、-x オプションを使用している。

実行結果

```
dir     man.md              create_hoge_files.sh  'one'$'\n''two.txt'
ls.md   auto_create_md.py   ping.md
```

実行結果

```
ls.md                  man.md               auto_create_md.py   dir   ping.md
create_hoge_files.sh  'one'$'\n''two.txt'
```

実行結果

```
dir               'one'$'\n''two.txt'   create_hoge_files.sh   ls.md
auto_create_md.py   ping.md             man.md
```

実行結果

```
auto_create_md.py    create_hoge_files.sh   dir   ls.md   man.md
'one'$'\n''two.txt'   ping.md
```

実行結果

```
dir                   ls.md                man.md   ping.md   auto_create_md.py
create_hoge_files.sh  'one'$'\n''two.txt'
```

-u

ファイル (ディレクトリ) のアクセス時刻 (atime) に基づいてソートする際の動作を指定するオプション。

しばしば、-lt オプションや -l オプションなどと併用される。

-lt オプションと併用： ファイルの詳細情報がアクセス時刻 (atime) が新しい順で表示される。

-l オプションと併用： ファイルの詳細情報がファイル名のアルファベット順でアクセス時刻 (atime) とともに表示される。

実行例

```
ls -ltu
ls -lu
```

実行結果

```
total 44
drwxrwxr-x 2 user user  4096 Dec 28 02:11  dir
-rw-rw-r-- 1 user user     0 Dec 28 01:52 'one'$'\n''two.txt'
-rw-rw-r-- 1 user user 11692 Dec 27 22:45  ls.md
-rwxrwxr-x 1 user user    63 Dec 26 01:27  create_hoge_files.sh
-rw-r--r-- 1 user user  4519 Dec 25 15:10  auto_create_md.py
-rw-rw-r-- 1 user user 10558 Dec 11 23:42  man.md
-rw-rw-r-- 1 user user  1024 Dec 11 04:23  ping.md
```

実行結果

```
total 44
-rw-r--r-- 1 user user  4519 Dec 25 15:10  auto_create_md.py
-rwxrwxr-x 1 user user    63 Dec 26 01:27  create_hoge_files.sh
drwxrwxr-x 2 user user  4096 Dec 28 02:11  dir
-rw-rw-r-- 1 user user 11692 Dec 27 22:45  ls.md
-rw-rw-r-- 1 user user 10558 Dec 11 23:42  man.md
-rw-rw-r-- 1 user user     0 Dec 28 01:52 'one'$'\n''two.txt'
-rw-rw-r-- 1 user user  1024 Dec 11 04:23  ping.md
```

-U

ファイル (ディレクトリ) 名をソートせずに表示するオプション。

実行例

```
ls -U
```

実行結果

```
auto_create_md.py      dir     man.md               ping.md
create_hoge_files.sh   ls.md  'one'$'\n''two.txt'
```

実行結果

```
dir      create_hoge_files.sh   ls.md               ping.md
man.md  'one'$'\n''two.txt'     auto_create_md.py
```

-v

ファイル (ディレクトリ) 名の数字が自然な順序で表示するオプション。

実行例

```
ls -v
```

実行結果

```
auto_create_md.py    create_hoge_files.sh   dir   hoge10.txt   hoge1.txt   hoge2.txt   ›
›ls.md   man.md
 'one'$'\n''two.txt'   ping.md
```

実行結果

```
auto_create_md.py    create_hoge_files.sh   dir   hoge1.txt   hoge2.txt   hoge10.txt   ›
›ls.md   man.md
 'one'$'\n''two.txt'   ping.md
```

上記例からこのオプションを使用すると、hoge10.txt の位置が hoge2.txt の後に来ることがわかる。

-w もしくは --width=COLS

出力幅 (横の文字数) を引数 (COLS) にて指定できるオプション。0 を渡すと、制限なしと解釈される。

実行例

```
ls -w 50
もしくは
ls --width=50
```

実行結果

```
auto_create_md.py      hoge2.txt
create_hoge_files.sh   ls.md
dir                    man.md
hoge10.txt            'one'$'\n''two.txt'
hoge1.txt              ping.md
```

-x

ファイル (ディレクトリ) 名を行ごとに表示するオプション。

実行例

```
ls -x
```

実行結果

```
auto_create_md.py      dir          hoge1.txt   ls.md   'one'$'\n''two.txt'
create_hoge_files.sh   hoge10.txt   hoge2.txt   man.md   ping.md
```

実行結果

```
auto_create_md.py   create_hoge_files.sh   dir      hoge10.txt           hoge1.txt
hoge2.txt           ls.md                  man.md  'one'$'\n''two.txt'   ping.md
```

-X

ファイルの拡張子のアルファベット順でソートして表示するオプション。

実行例

```
ls -X
```

実行結果

```
// 容易のため ls -xX で実行

dir                    ls.md        man.md      ping.md     auto_create_md.py
create_hoge_files.sh   hoge10.txt   hoge1.txt   hoge2.txt  'one'$'\n''two.txt'
```

-1

1 行に 1 ファイル (ディレクトリ) のみ表示するオプション。

実行例

```
ls -1
```

実行結果

```
auto_create_md.py
create_hoge_files.sh
dir
hoge10.txt
hoge1.txt
hoge2.txt
ls.md
man.md
'one'$'\n''two.txt'
ping.md
```

1.7 pstree

現在実行されているプロセスをツリー表示するコマンド

実行例

```
pstree
```

実行結果

```
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─agetty
        ├─containerd───8*[{containerd}]
        ├─cron
        ├─dbus-daemon
        ├─dockerd───9*[{dockerd}]
        ├─irqbalance───{irqbalance}
        ├─multipathd───6*[{multipathd}]
        ├─networkd-dispat
        ├─packagekitd───2*[{packagekitd}]
        ├─polkitd───2*[{polkitd}]
        ├─rsyslogd───3*[{rsyslogd}]
        ├─snapd───10*[{snapd}]
        ├─sshd───sshd───sshd───bash───pstree
        ├─systemd───(sd-pam)
        ├─systemd-journal
        ├─systemd-logind
        ├─systemd-network
        ├─systemd-resolve
        ├─systemd-timesyn───{systemd-timesyn}
        ├─systemd-udevd
        ├─tailscaled───10*[{tailscaled}]
        ├─thermald───{thermald}
        ├─udisksd───4*[{udisksd}]
        ├─unattended-upgr───{unattended-upgr}
        └─upowerd───2*[{upowerd}]
```

♣ オプション一覧

-a

各プロセスのコマンドライン引数を表示する

実行例

```
pstree -a
```

実行結果

```
systemd --system --deserialize 36
 ├─ModemManager
 │    └─2*[{ModemManager}]
 ├─agetty -o -p -- \\u --noclear tty1 linux
 ├─containerd
```

```
|    └─8*[{containerd}]
├─cron -f -P
├─dbus-daemon --system --address=systemd: --nofork --nopidfile...
├─dockerd -H fd:// --containerd=/run/containerd/containerd.sock
|    └─9*[{dockerd}]
├─irqbalance --foreground
|    └─{irqbalance}
├─multipathd -d -s
|    └─6*[{multipathd}]
├─networkd-dispat /usr/bin/networkd-dispatcher --run-startup-triggers
├─packagekitd
|    └─2*[{packagekitd}]
├─polkitd --no-debug
|    └─2*[{polkitd}]
├─rsyslogd -n -iNONE
|    └─3*[{rsyslogd}]
├─snapd
|    └─10*[{snapd}]
├─sshd
|    └─sshd
|        └─sshd
|            └─bash
|                └─pstree -a
├─systemd --user
|    └─(sd-pam)
├─systemd-journal
├─systemd-logind
├─systemd-network
├─systemd-resolve
├─systemd-timesyn
|    └─{systemd-timesyn}
├─systemd-udevd
├─tailscaled --state=/var/lib/tailscale/tailscaled.state--socket=/run/ta
|    └─10*[{tailscaled}]
├─thermald --systemd --dbus-enable --adaptive
|    └─{thermald}
├─udisksd
|    └─4*[{udisksd}]
├─unattended-upgr ...
|    └─{unattended-upgr}
└─upowerd
     └─2*[{upowerd}]
```

-c

-c オプションを使用すると、各プロセスの実行可能ファイル (コマンド) の名前を表示する。

実行例

```
pstree -c
```

実行結果

```
systemd─┬─ModemManager─┬─{ModemManager}
        |              └─{ModemManager}
        ├─agetty
        ├─containerd─┬─{containerd}
        |            ├─{containerd}
        |            ├─{containerd}
```

```
        │            ├─{containerd}
        │            ├─{containerd}
        │            ├─{containerd}
        │            ├─{containerd}
        │            └─{containerd}
        ├─cron
        ├─dbus-daemon
        ├─dockerd─┬─{dockerd}
        │         ├─{dockerd}
        │         ├─{dockerd}
        │         ├─{dockerd}
        │         ├─{dockerd}
        │         ├─{dockerd}
        │         ├─{dockerd}
        │         ├─{dockerd}
        │         └─{dockerd}
        ├─irqbalance─────{irqbalance}
        ├─multipathd─┬─{multipathd}
        │            ├─{multipathd}
        │            ├─{multipathd}
        │            ├─{multipathd}
        │            ├─{multipathd}
        │            └─{multipathd}
        ├─networkd-dispat
        ├─packagekitd─┬─{packagekitd}
        │             └─{packagekitd}
        ├─polkitd─┬─{polkitd}
        │         └─{polkitd}
        ├─rsyslogd─┬─{rsyslogd}
        │          ├─{rsyslogd}
        │          └─{rsyslogd}
        ├─snapd─┬─{snapd}
        │       ├─{snapd}
        │       ├─{snapd}
        │       ├─{snapd}
        │       ├─{snapd}
        │       ├─{snapd}
        │       ├─{snapd}
        │       ├─{snapd}
        │       ├─{snapd}
        │       └─{snapd}
        ├─sshd─────sshd─────sshd─────bash─────pstree
        ├─systemd──────(sd-pam)
        ├─systemd-journal
        ├─systemd-logind
        ├─systemd-network
        ├─systemd-resolve
        ├─systemd-timesyn─────{systemd-timesyn}
        ├─systemd-udevd
        ├─tailscaled─┬─{tailscaled}
        │            ├─{tailscaled}
        │            ├─{tailscaled}
        │            ├─{tailscaled}
        │            ├─{tailscaled}
        │            ├─{tailscaled}
        │            ├─{tailscaled}
        │            ├─{tailscaled}
        │            └─{tailscaled}
        ├─thermald─────{thermald}
        ├─udisksd─┬─{udisksd}
        │         ├─{udisksd}
        │         └─{udisksd}
        ├─unattended-upgr─────{unattended-upgr}
        └─upowerd─┬─{upowerd}
```

```
                └─{upowerd}
```

- **-h**

文章説明

実行例

```
pstree -h
```

実行結果

```
systemd─┬─ModemManager─────2*[{ModemManager}]
        ├─agetty
        ├─containerd─────8*[{containerd}]
        ├─cron
        ├─dbus-daemon
        ├─dockerd─────9*[{dockerd}]
        ├─irqbalance─────{irqbalance}
        ├─multipathd─────6*[{multipathd}]
        ├─networkd-dispat
        ├─packagekitd─────2*[{packagekitd}]
        ├─polkitd─────2*[{polkitd}]
        ├─rsyslogd─────3*[{rsyslogd}]
        ├─sh─────node─┬─node─┬─bash─────pstree
        │             │      └─11*[{node}]
        │             ├─node─┬─node─────6*[{node}]
        │             │      └─15*[{node}]
        │             ├─node─────12*[{node}]
        │             └─10*[{node}]
        ├─snapd─────10*[{snapd}]
        ├─sshd─┬─sshd─────sshd─────bash
        │      └─sshd─────sshd
        ├─systemd─────(sd-pam)
        ├─systemd-journal
        ├─systemd-logind
        ├─systemd-network
        ├─systemd-resolve
        ├─systemd-timesyn─────{systemd-timesyn}
        ├─systemd-udevd
        ├─tailscaled─────10*[{tailscaled}]
        ├─thermald─────{thermald}
        ├─udisksd─────4*[{udisksd}]
        ├─unattended-upgr─────{unattended-upgr}
        └─upowerd─────2*[{upowerd}]
```

@{-l}

長い行を表示。デフォルトでは、長い行はディスプレイの幅で切られる。

実行例

```
pstree -l
```

実行結果

```
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─agetty
        ├─containerd───8*[{containerd}]
        ├─cron
        ├─dbus-daemon
        ├─dockerd───9*[{dockerd}]
        ├─irqbalance───{irqbalance}
        ├─multipathd───6*[{multipathd}]
        ├─networkd-dispat
        ├─packagekitd───2*[{packagekitd}]
        ├─polkitd───2*[{polkitd}]
        ├─rsyslogd───3*[{rsyslogd}]
        ├─sh───node─┬─node─┬─bash───pstree
        │           │      └─11*[{node}]
        │           ├─node───12*[{node}]
        │           ├─node───15*[{node}]
        │           └─10*[{node}]
        ├─snapd───10*[{snapd}]
        ├─sshd─┬─sshd───sshd───bash
        │      └─sshd───sshd
        ├─systemd───(sd-pam)
        ├─systemd-journal
        ├─systemd-logind
        ├─systemd-network
        ├─systemd-resolve
        ├─systemd-timesyn───{systemd-timesyn}
        ├─systemd-udevd
        ├─tailscaled───10*[{tailscaled}]
        ├─thermald───{thermald}
        ├─udisksd───4*[{udisksd}]
        ├─unattended-upgr───{unattended-upgr}
        └─upowerd───2*[{upowerd}]
```

-n

同じ親を持つプロセスを、名前ではなく PID でソートしる。

実行例

```
pstree -n
```

実行結果

```
systemd─┬─multipathd───6*[{multipathd}]
        ├─cron
        ├─dbus-daemon
        ├─irqbalance───{irqbalance}
        ├─networkd-dispat
        ├─polkitd───2*[{polkitd}]
        ├─rsyslogd───3*[{rsyslogd}]
        ├─systemd-logind
        ├─thermald───{thermald}
        ├─agetty
        ├─ModemManager───2*[{ModemManager}]
        ├─unattended-upgr───{unattended-upgr}
        ├─tailscaled───10*[{tailscaled}]
        ├─snapd───10*[{snapd}]
        ├─containerd───8*[{containerd}]
        ├─dockerd───9*[{dockerd}]
        ├─systemd-timesyn───{systemd-timesyn}
        ├─packagekitd───2*[{packagekitd}]
```

```
        ├─upowerd─────2*[{upowerd}]
        ├─sshd─┬─┬─sshd─────sshd─────bash
        │      └─sshd─────sshd
        ├─systemd-journal
        ├─udisksd─────4*[{udisksd}]
        ├─systemd-udevd
        ├─systemd-network
        ├─systemd-resolve
        ├─systemd─────(sd-pam)
        └─sh─────node─┬─┬─10*[{node}]
                      ├─node─┬─┬─11*[{node}]
                      │      └─bash─────pstree
                      ├─node─────12*[{node}]
                      └─node─────15*[{node}]
```

-p

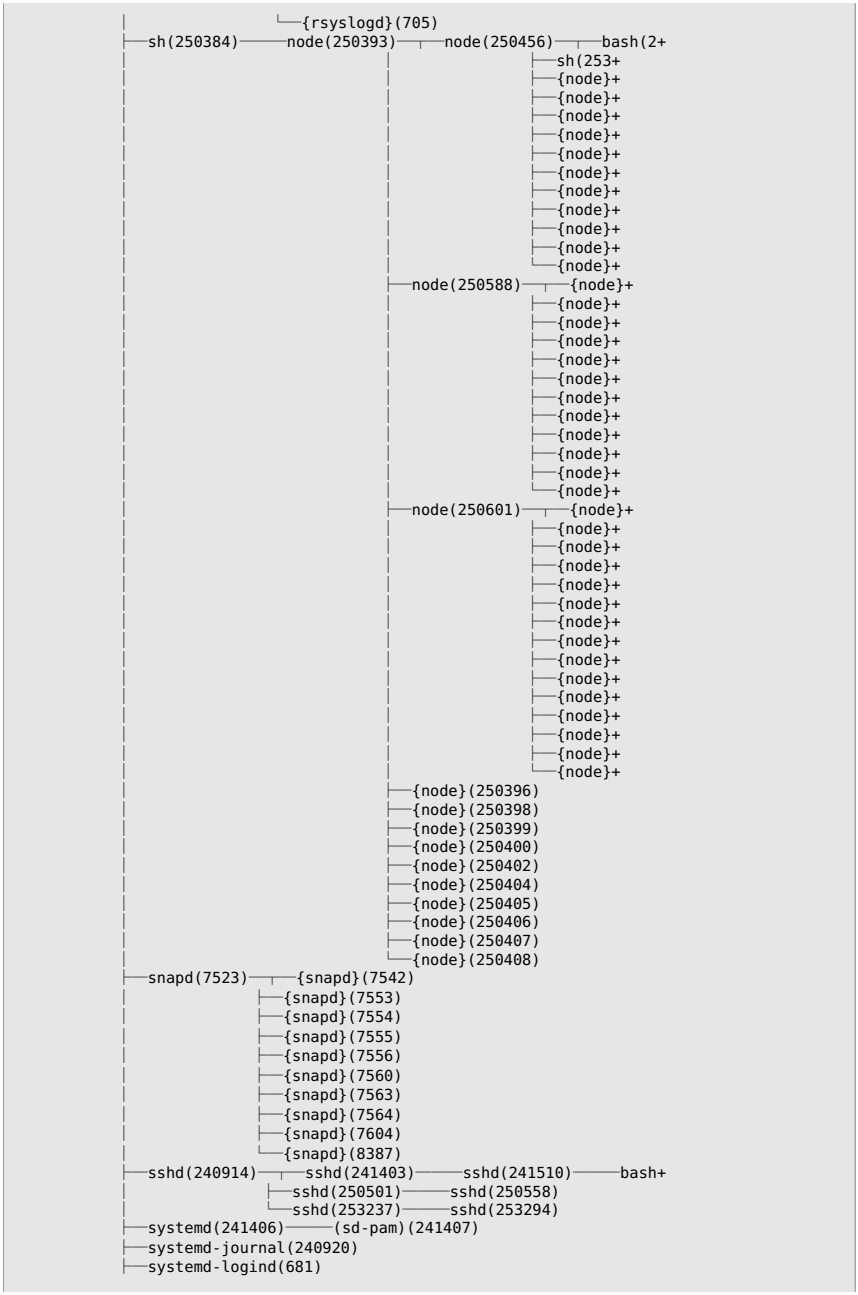
文章説明

実行例

```
pstree -p
```

実行結果

```
systemd(1)─┬─┬─ModemManager(743)─┬─┬─{ModemManager}(749)
           │                     └─{ModemManager}(755)
           ├─agetty(702)
           ├─containerd(12189)─┬─┬─{containerd}(12190)
           │                   ├─{containerd}(12191)
           │                   ├─{containerd}(12192)
           │                   ├─{containerd}(12193)
           │                   ├─{containerd}(12194)
           │                   ├─{containerd}(12195)
           │                   ├─{containerd}(12196)
           │                   └─{containerd}(12199)
           ├─cron(665)
           ├─dbus-daemon(666)
           ├─dockerd(12520)─┬─┬─{dockerd}(12521)
           │                ├─{dockerd}(12522)
           │                ├─{dockerd}(12523)
           │                ├─{dockerd}(12524)
           │                ├─{dockerd}(12525)
           │                ├─{dockerd}(12526)
           │                ├─{dockerd}(12527)
           │                ├─{dockerd}(12528)
           │                └─{dockerd}(12531)
           ├─irqbalance(673)─────{irqbalance}(688)
           ├─multipathd(457)─┬─┬─{multipathd}(462)
           │                 ├─{multipathd}(463)
           │                 ├─{multipathd}(464)
           │                 ├─{multipathd}(465)
           │                 ├─{multipathd}(466)
           │                 └─{multipathd}(467)
           ├─networkd-dispat(674)
           ├─packagekitd(240910)─┬─┬─{packagekitd}(240915)
           │                     └─{packagekitd}(240916)
           ├─polkitd(675)─┬─┬─{polkitd}(689)
           │              └─{polkitd}(729)
           ├─rsyslogd(676)─┬─┬─{rsyslogd}(703)
           │               ├─{rsyslogd}(704)
```


└─{rsyslogd}(705)
├─sh(250384)───node(250393)─┬─node(250456)─┬─bash(2+
├─sh(253+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
└─{node}+
├─node(250588)─┬─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
└─{node}+
├─node(250601)─┬─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
├─{node}+
└─{node}+
├─{node}(250396)
├─{node}(250398)
├─{node}(250399)
├─{node}(250400)
├─{node}(250402)
├─{node}(250404)
├─{node}(250405)
├─{node}(250406)
├─{node}(250407)
└─{node}(250408)
├─snapd(7523)─┬─{snapd}(7542)
├─{snapd}(7553)
├─{snapd}(7554)
├─{snapd}(7555)
├─{snapd}(7556)
├─{snapd}(7560)
├─{snapd}(7563)
├─{snapd}(7564)
├─{snapd}(7604)
└─{snapd}(8387)
├─sshd(240914)─┬─sshd(241403)─────sshd(241510)─────bash+
├─sshd(250501)─────sshd(250558)
└─sshd(253237)─────sshd(253294)
├─systemd(241406)─────(sd-pam)(241407)
├─systemd-journal(240920)
├─systemd-logind(681)

```
           ├─systemd-network(240946)
           ├─systemd-resolve(240948)
           ├─systemd-timesyn(240909)─────{systemd-timesyn}(24094+
           ├─systemd-udevd(240939)
           ├─tailscaled(793)─┬─{tailscaled}(794)
           │                 ├─{tailscaled}(795)
           │                 ├─{tailscaled}(796)
           │                 ├─{tailscaled}(797)
           │                 ├─{tailscaled}(798)
           │                 ├─{tailscaled}(799)
           │                 ├─{tailscaled}(800)
           │                 ├─{tailscaled}(802)
           │                 ├─{tailscaled}(803)
           │                 └─{tailscaled}(811)
           ├─thermald(684)─────{thermald}(746)
           ├─udisksd(240921)─┬─{udisksd}(240924)
           │                 ├─{udisksd}(240926)
           │                 ├─{udisksd}(240932)
           │                 └─{udisksd}(240943)
           ├─unattended-upgr(744)─────{unattended-upgr}(765)
           └─upowerd(240913)─┬─{upowerd}(240930)
                             └─{upowerd}(240931)
```

-u

文章説明

実行例

```
pstree -u
```

実行結果

```
systemd─┬─ModemManager─────2*[{ModemManager}]
        ├─agetty
        ├─containerd─────8*[{containerd}]
        ├─cron
        ├─dbus-daemon(messagebus)
        ├─dockerd─────9*[{dockerd}]
        ├─irqbalance─────{irqbalance}
        ├─multipathd─────6*[{multipathd}]
        ├─networkd-dispat
        ├─packagekitd─────2*[{packagekitd}]
        ├─polkitd─────2*[{polkitd}]
        ├─rsyslogd(syslog)─────3*[{rsyslogd}]
        ├─sh(tomo)─────node─┬─node─┬─bash─────pstree
        │                   │      ├─sh─────cpuUsage.sh─────sleep
        │                   │      └─11*[{node}]
        │                   ├─node─────12*[{node}]
        │                   ├─node─────15*[{node}]
        │                   └─10*[{node}]
        ├─snapd─────10*[{snapd}]
        ├─sshd─┬─sshd─────sshd(tomo)─────bash
        │      └─2*[sshd─────sshd(tomo)]
        ├─systemd(tomo)─────(sd-pam)
        ├─systemd-journal
        ├─systemd-logind
        ├─systemd-network(systemd-network)
        ├─systemd-resolve(systemd-resolve)
        ├─systemd-timesyn(systemd-timesync)─────{systemd-timesyn+
        ├─systemd-udevd
```

```
        ├─tailscaled─────10*[{tailscaled}]
        ├─thermald───────{thermald}
        ├─udisksd────────4*[{udisksd}]
        ├─unattended-upgr────{unattended-upgr}
        └─upowerd────────2*[{upowerd}]
```

1.8 pwd

自分が現在自分がいるディレクトリを表示するコマンド

実行例

```
pwd
```

実行結果

```
/home/user   //現在自分がいるディレクトリを表示
```

♣ オプション一覧

-L

オプション無しと同じ挙動をします。オプションをつけても付けなくてもいい。

実行例

```
pwd -L
```

実行結果

```
/home/user  //現在自分がいるディレクトリを表示
```

-P

このオプションは、カレントディレクトリの物理的なパスを表示する。シンボリックリンクが含まれる場合でも、シンボリックリンクが解決され、実際のディレクトリのパスが表示される。別のファイルやディレクトリへの参照を作成するための特殊なファイル。

実行例

```
pwd -P
```

実行結果

```
$ ln -s /tmp tmp  /現在のディレクトリにシンボリックリンクを貼る

$ ls
tmp //現在のディレクトリで/（ルート）直下のtmpを参照できる
$ cd tmp
$ pwd    //オプション無し
home/user/Prmn/pwd

$ pwd -P //オプションあり
/tmp
```

1.9 top

top コマンドは、実行中のシステムの負荷状態を動的に確認できる。カーネルによって管理されているプロセスやスレッドのリストだけでなく、システムの概要情報も表示可能。それぞれのラベルの意味は以下に示す。

PID:プロセスの番号｜USER:実行者｜PR:優先度｜NI:相対優先度｜VIRT:仮想メモリ｜RES:物理メモリ｜SHR:共有メモリ｜S:状態 (D:割り込み不可,R:実行中,S:スリープ状態,T:停止中,Z:ゾンビプロセス)｜%CPU,%MEM:CPU, メモリの使用率｜TIME+:プロセスの起動してからの経過時間。

オプション無しでは CPU 使用率順にソートされて表示される。詳しい操作方法は h を推すことで確認が可能である。

以下のオプションは top コマンドが動いている状態でキーを押しても同様な操作をできる。

実行例

```
top
```

実行結果

```
 top - 19:46:21 up 3 days,  6:20,  4 users,  load average: 0.75, 0.43, 0.36
Tasks: 243 total,   1 running, 242 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.8 us,  0.7 sy,  0.0 ni, 96.2 id,  2.3 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 16204116 total,   418872 free,  2862832 used, 12922412 buff/cache
KiB Swap:  2097148 total,  2094076 free,     3072 used. 12585868 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   2208 shunsuke  20   0   12.2g 600384 238248 S   2.7   3.7 158:45.45 firefox
  33026 shunsuke  20   0 2589236 202292  99788 S   1.3   1.2   7:03.68 Isolated Web Co
  30332 root      20   0 1388388  48296  19712 S   0.7   0.3  86:50.32 tailscaled
  32841 shunsuke  20   0 2616980 175496  94456 S   0.7   1.1  20:30.79 Isolated Web Co
   2942 shunsuke  20   0 2704040 281128  98600 S   0.3   1.7   6:23.49 Isolated Web Co
  33014 shunsuke  20   0 2668360 167660  93040 S   0.3   1.0  11:43.88 Isolated Web Co
  47534 systemd+  20   0   14828   6912   6144 S   0.3   0.0   3:50.44 systemd-oomd
  57499 shunsuke  20   0   14396   4224   3328 R   0.3   0.0   0:00.31 top
      1 root      20   0  166664  11648   8320 S   0.0   0.1   0:14.75 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.12 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
```

```
    5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 slub_flushwq
~~プロセスが多いので省略~~
```

上の結果を見ると Firefox ブラウザ (PID 2208) は、約 12.2GB の仮想メモリを使用し、約 600MB の物理メモリを消費しており、CPU 使用率は 2.3%、メモリ使用率は 3.8% だということがわかる。

♣ オプション一覧

-b

バッチモード。キー操作を受け付けず、「-n」で指定された回数か強制終了するまで実行を続ける。top の結果をファイルに保存する際などに役立つ。

実行例

```
tload -s 10
```

実行結果

```
 2208 user  20   0   12.2g 546440 238592 S   4.0   3.4  60:48.81 firefox
30332 root      20   0 1320912  50064  19712 S   1.3   0.3  59:20.04 tailscaled
2733 user  20   0 2949016 310416 113852 S   0.7   1.9   3:01.25 Isolated Web Co
15 root      20   0       0      0      0 I   0.3   0.0   1:11.40 rcu_preempt
~~(プロセスが多いので省略~~)
```

-d

グラフの更新間隔を秒単位で指定する。

実行例

```
top -d 10
```

実行結果

```
(以下、ソートおよび表示変更のため実行結果を記載しない)
```

-d

グラフの更新間隔を秒単位で指定する。

実行例

```
top -d 10
```

実行結果

```
(以下ソートおよび表示変更のものは実行結果を記載しない)
```

-E

top のエリアに表示されるメモリの表示単位を指定する。

実行例

```
top -E k
```

実行結果

```
KiB Mem : 16204116 total,   390028 free,  2894688 used, 12919400 buff/   cache
KiB Swap:  2097148 total,  2094076 free,     3072 used. 12554084 avail Mem
```

-e

タスクエリアに表示されるメモリの表示単位を指定する。

実行例

```
top -e k
```

実行結果

```

```

-H

top に個別のスレッドを表示する。このオプションを使用しないと、各プロセスのすべてのスレッドごとの合計が表示される。

実行例

```
top -e k
```

実行結果

```

```

-i

プロセスが cpu を使用しているもののみを表示する。

実行例

```
top -e k
```

実行結果

```
```

-n

表示する回数を指定する。

実行例

```
top -n 1
```

実行結果

```
```

-o

ソートなどで指定できる利用可能なフィールド名を表示して終了する。

実行例

```
top -e k
```

実行結果

```
PID
PPID
~~(長いので省略)~~
RSsh
CGNAME
NU
```

-o

タスクエリアの表示をフィールド名を指定してソートする。

実行例

```
top -n 1
```

実行結果

```
```

-p

特定のプロセス ID のプロセスのみ表示する。

実行例

```
top -n 1
```

実行結果

```
```

-S

最後に記憶された「S」状態からの計測した累積時間モードに切り替える。

実行例

```
top -S 1
```

実行結果

```
```

-U

指定したユーザーのプロセスに関連する情報だけでなく、他のプロセスも表示。

実行例

```
top -U root
```

実行結果

```
```

-u

指定したユーザーに関連するプロセスのみが表示。

実行例

```
top -U root
```

実行結果

```
```

-w

出力の幅を指定する。

実行例

```
top -w 110
```

実行結果

-1 ※小文字 L ではない

サマリーエリアの CPU ステータスを切り替える。

実行例

```
top -1
```

実行結果

```
top - 00:29:54 up 2 days, 11:03,  4 users,  load average: 0.12, 0.23, 0.25
Tasks: 243 total,   1 running, 242 sleeping,   0 stopped,   0 zombie
%Cpu0  :  5.9 us,  0.0 sy,  0.0 ni, 94.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu1  :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu2  :  5.9 us,  0.0 sy,  0.0 ni, 94.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu3  :  6.2 us,  0.0 sy,  0.0 ni, 93.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 16204116 total,   443740 free,  2837256 used, 12923120 buff/cache
KiB Swap:  2097148 total,  2094076 free,     3072 used. 12611496 avail Mem
```

あとがき / おわりに

「厳選 Unix コマンド」(初版; v1.0.0 〜公立千歳科学技術大学 IT インフラ部へよこそ〜) は、いかがだったでしょうか。

フォーマット変換の仕組みづくりと PDF ビルド部分だけ担当した深町です。

フォーマット変換がうまくいかないところは、元原稿に手をいれて直しましたが、日本語自体は手をいれていません。そのため文体もマチマチですが、それもオムニバス執筆の味ということで、ひとつよろしくお願いします。

IT インフラ修行自体は春学期からやっているわけですが、本原稿自体は、情報システム工学科 3 年秋学期のプロジェクト科目の成果物です。

元は GFM (github flavoured markdown) なので、それを Re:VIEW 形式にフォーマット変換するしくみを作り、自動変換しました。その後 Re:VIEW 2.5 で PDF を製作しています。

電子版 PDF は、成果物のすべてが収録されていて 300 ページくらいあります。印刷版には「収録したいコマンド」を電子版から選りすぐってもらいました。

前書きでは「演習授業のおともに」と書いてありますが、厳密には授業 (演習の詳細) とシンクロしていません。毎年、授業内容が改変もといチューニングされていっているので、2024 年度の構成が決まらない中、平行しての執筆作業なので、なかなかシンクロは難しかったでしょう。

ただ、「そろそろ授業内容も固まってきた」という感触があるので、次回は、もっと授業内容とシンクロした、いわばカリカリに授業チューンアップされた「厳選 Unix コマンド 第二版　〜届けたいコマンド〜」が製作できると思います。大絶賛、執筆者募集中です！！

♣ 著者紹介

川合志門、北川武、後藤駿介、田村跳飛、中西和音、栁田颯太 (五十音順)

ラピダス建設中のクレーンが工場の向こうに見える千歳市美々より (左端が大学)

厳選 Unix コマンド v1.0.0

Selected Unix Commands for Beginners

2024 年 2 月 29 日　v1.0.0

著　者　公立千歳科学技術大学 IT インフラ部
発行者　深町賢一
連絡先　infra-club@cist.fml.org
　　https://selected-unix-commands.techbooks.fml.org/
印刷所　株式会社イシダ印刷

(powered by Re:VIEW Starter)