厳選 Unix コマンド v0.3.0

— 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　v0.3.0

■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

第 1 章
頭文字が a のコマンド

1.1 aa-enabled

AppArmor(Linux のセキュリティプロファイルを結びつけることで、ネットワークアクセスや Raw socket アクセス、ファイルへの読み書き実行などの機能に制限をかけることができるプログラム。例えば、あるプログラムに対して、参照しかしないファイルへの書き込み、利用するはずがないファイルへのアクセス、関係がない別のプログラムの呼び出しを禁止できる。"/etc/apparmor/parser.conf"に記述された通りに制限がかけられる。) の状態を確認するためのコマンド。AppArmor が有効になっている場合は"Yes"、無効になっている場合は"No"が出力される。

実行例

```
aa-enabled
```

実行結果

```
Yes
```

♣ オプション一覧

-q

何も出力しない。単に AppArmor が有効になっているかどうかを判断するための終了コード。

実行例

```
aa-enabled -q
```

実行結果

```

```

-x

AppArmor が有効であるとみなされるには、共有 LSM インターフェイスへの排他的アクセスが必要。

実行例

```
aa-enabled -x
```

実行結果

```
Yes
```

1.2 aa-exec

指定した AppArmor(Linux のセキュリティプロファイルを結びつけることで、ネットワークアクセスや Raw socket アクセス、ファイルへの読み書き実行などの機能に制限をかけることができるプログラム。例えば、あるプログラムに対して、参照しかしないファイルへの書き込み、利用するはずがないファイルへのアクセス、関係がない別のプログラムの呼び出しを禁止できる。"/etc/apparmor/parser.conf"に記述された通りに制限がかけられる。) プロファイルや、指定した namespace(イメージとしてはコンテナに近い。OS 起動時にはデフォルトの namespace が存在し、全てのプロセスはデフォルトの namespace に属している。プロセスの起動時に独立した namespace で実行する指定を行うと、そのプロセスは別の namespace で実行できる。独立した namespace で実行することで、namespace ごとにドメイン名を持てたり、namespace ごとにマウント操作を行えたり、namespace ごとに各種ネットワークリソースを持つことができたりする。) で、特定のプログラムを起動するためのコマンド。namespace とプロファイルの両方を指定した場合、新しい namespace 内でプロファイルの制限を受けることになる。namespace のみを指定した場合、現在の制約のプロファイル名が使用される。プロファイルも namespace も指定しない場合、aa-exec コマンドなしで実行された場合と同様の実行結果となる。root 権限が必要。

実行例

```
sudo aa-exec /bin/aa-enabled
```

実行結果

```
Yes
```

♣ オプション一覧

-p "PROFILE"

指定したプロファイルでコマンドを制限して実行する。

実行例

```
sudo aa-exec -p /usr/bin/evince aa-enabled
```

実行結果

```
Yes
```

-n "NAMESPACE"

指定した namespace 内のプロファイルでコマンドを制限して実行する。

実行例

```
sudo aa-exec -n ns1 aa-enabled
```

実行結果

```
Yes
```

-i

現在のプロファイルに従ってコマンドを制限して実行する。このオプションを実行する前に指定したいプロファイルに遷移しておく必要がある。

実行例

```
sudo aaa-exec -i aa-enabled
```

実行結果

```
Yes
```

-v

実行するコマンドを表示する。

実行例

```
sudo aa-exec -v aa-enabled
```

実行結果

```
[3037] exec aa-enabled
Yes
```

-d

実行するコマンドとエラーコードを表示する

実行例

```
sudo aa-exec -d -- cd aaaaaaaaaa
```

実行結果

```
[3911] aa-exec: ERROR: Failed to execute "cd": No such file or directory
```

--

--以前のオプションを終了させ、--以降の引数はコマンドの引数として扱われる。aa-exec によって呼び出されるコマンドに引数を渡す場合に便利。

実行例

```
sudo aa-exec -d -- cd aaaaaaaaaa
```

実行結果

```
[3911] aa-exec: ERROR: Failed to execute "cd": No such file or directory
```

1.3 aa-status

AppArmor(Linux のセキュリティプロファイルを結びつけることで、ネットワークアクセスや Raw socket アクセス、ファイルへの読み書き実行などの機能に制限をかけることができるプログラム。例えば、あるプログラムに対して、参照しかしないファイルへの書き込み、利用するはずがないファイルへのアクセス、関係がない別のプログラムの呼び出しを禁止できる。"/etc/apparmor/parser.conf"に記述された通りに制限がかけられる。) の状態を表示するコマンド。デフォルトでは、--verbose オプションと同じ情報が表示される。実行には root 権限が必要。

実行例

```
sudo aa-status
```

実行結果

```
apparmor module is loaded.
49 profiles are loaded.
47 profiles are in enforce mode.
 /snap/snapd/19993/usr/lib/snapd/snap-confine
 /snap/snapd/19993/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
 /snap/snapd/20092/usr/lib/snapd/snap-confine
 /snap/snapd/20092/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
 /snap/snapd/20290/usr/lib/snapd/snap-confine
 /snap/snapd/20290/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
 /usr/bin/evince
 /usr/bin/evince-previewer
 /usr/bin/evince-previewer//sanitized_helper
 /usr/bin/evince-thumbnailer
 /usr/bin/evince//sanitized_helper
 /usr/bin/man
 /usr/lib/NetworkManager/nm-dhcp-client.action
 /usr/lib/NetworkManager/nm-dhcp-helper
 /usr/lib/connman/scripts/dhclient-script
 /usr/lib/cups/backend/cups-pdf
 /usr/lib/snapd/snap-confine
 /usr/lib/snapd/snap-confine//mount-namespace-capture-helper
 /usr/sbin/cups-browsed
 /usr/sbin/cupsd
 /usr/sbin/cupsd//third_party
 /{,usr/}sbin/dhclient
 docker-default
 libreoffice-senddoc
 libreoffice-soffice//gpg
 libreoffice-xpdfimport
 lsb_release
 man_filter
 man_groff
 nvidia_modprobe
 nvidia_modprobe//kmod
 snap-update-ns.firefox
 snap-update-ns.snap-store
 snap-update-ns.snapd-desktop-integration
 snap.firefox.firefox
 snap.firefox.geckodriver
 snap.firefox.hook.configure
 snap.firefox.hook.connect-plug-host-hunspell
 snap.firefox.hook.disconnect-plug-host-hunspell
 snap.firefox.hook.post-refresh
 snap.snap-store.hook.configure
 snap.snap-store.snap-store
 snap.snap-store.ubuntu-software
 snap.snap-store.ubuntu-software-local-file
 snap.snapd-desktop-integration.hook.configure
 snap.snapd-desktop-integration.snapd-desktop-integration
 tcpdump
2 profiles are in complain mode.
 libreoffice-oosplash
 libreoffice-soffice
0 profiles are in kill mode.
0 profiles are in unconfined mode.
7 processes have profiles defined.
7 processes are in enforce mode.
 /usr/bin/man (93992)
 /usr/bin/less (94000) /usr/bin/man
 /usr/sbin/cups-browsed (94258)
 /usr/sbin/cupsd (94221)
 /bin/snap-store (1413) snap.snap-store.ubuntu-software
 /bin/snapd-desktop-integration (872) snap.snapd-desktop-integration.snapd-desktop-integ›
›ration
 /bin/snapd-desktop-integration (1216) snap.snapd-desktop-integration.snapd-desktop-inte›
›gration
```

```
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
0 processes are in mixed mode.
0 processes are in kill mode.
```

♣ オプション一覧

--enabled

AppArmor が利用不可の場合にエラーコードを表示する。

実行例

```
sudo aa-status --enabled
```

実行結果

```
```

--profiled

読み込まれた AppArmor プロファイル (AppArmor を利用して設定された権限の情報が書かれたファイル) の数を表示する。

実行例

```
sudo aa-status --profiled
```

実行結果

```
49
```

--enforced

読み込まれた強制 AppArmor プロファイルの数を表示する。強制 AppArmor プロファイルは受けている制限が強制されるとともに、違反の試行が記録される。

実行例

```
sudo aa-status --enforced
```

実行結果

```
47
```

--complaining

読み込まれた苦情 AppArmor プロファイルの数を表示する。苦情 AppArmor プロファイルは受けている制限が強制されないが、違反の試行は記録される。

実行例

```
sudo aa-status --complaining
```

実行結果

```
2
```

--kill

違反時にタスクを強制終了する強制 AppArmor プロファイルを読み込み、数を表示する。

実行例

```
sudo aa-status --kill
```

実行結果

```
0
```

--special-unconfined

制限を受けていないモードである、非強制 AppArmor プロファイルを読み込み、数を表示する。

実行例

```
sudo aa-status --special-unconfined
```

実行結果

```
0
```

--process-mixed

異なるモードのプロファイルを持つプロファイルスタックによって制限されたプロセスの数を表示する。

実行例

```
sudo aa-status --process-mixed
```

実行結果

```
0
```

--verbose

AppArmor プロファイルの状態を表示する

実行例

```
sudo aa-status --verbose
```

実行結果

```
apparmor module is loaded.
49 profiles are loaded.
47 profiles are in enforce mode.
 /snap/snapd/19993/usr/lib/snapd/snap-confine
 /snap/snapd/19993/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
 /snap/snapd/20092/usr/lib/snapd/snap-confine
 /snap/snapd/20092/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
 /snap/snapd/20290/usr/lib/snapd/snap-confine
 /snap/snapd/20290/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
 /usr/bin/evince
 /usr/bin/evince-previewer
 /usr/bin/evince-previewer//sanitized_helper
 /usr/bin/evince-thumbnailer
 /usr/bin/evince//sanitized_helper
 /usr/bin/man
 /usr/lib/NetworkManager/nm-dhcp-client.action
 /usr/lib/NetworkManager/nm-dhcp-helper
 /usr/lib/connman/scripts/dhclient-script
 /usr/lib/cups/backend/cups-pdf
 /usr/lib/snapd/snap-confine
 /usr/lib/snapd/snap-confine//mount-namespace-capture-helper
 /usr/sbin/cups-browsed
 /usr/sbin/cupsd
 /usr/sbin/cupsd//third_party
 /{,usr/}sbin/dhclient
 docker-default
 libreoffice-senddoc
 libreoffice-soffice//gpg
 libreoffice-xpdfimport
 lsb_release
 man_filter
 man_groff
 nvidia_modprobe
 nvidia_modprobe//kmod
 snap-update-ns.firefox
 snap-update-ns.snap-store
 snap-update-ns.snapd-desktop-integration
 snap.firefox.firefox
 snap.firefox.geckodriver
 snap.firefox.hook.configure
 snap.firefox.hook.connect-plug-host-hunspell
 snap.firefox.hook.disconnect-plug-host-hunspell
 snap.firefox.hook.post-refresh
 snap.snap-store.hook.configure
 snap.snap-store.snap-store
 snap.snap-store.ubuntu-software
 snap.snap-store.ubuntu-software-local-file
 snap.snapd-desktop-integration.hook.configure
 snap.snapd-desktop-integration.snapd-desktop-integration
 tcpdump
2 profiles are in complain mode.
 libreoffice-oosplash
 libreoffice-soffice
0 profiles are in kill mode.
0 profiles are in unconfined mode.
7 processes have profiles defined.
7 processes are in enforce mode.
 /usr/bin/man (93992)
 /usr/bin/less (94000) /usr/bin/man
 /usr/sbin/cups-browsed (94258)
```

```
   /usr/sbin/cupsd (94221)
   /bin/snap-store (1413) snap.snap-store.ubuntu-software
   /bin/snapd-desktop-integration (872) snap.snapd-desktop-integration.snapd-desktop-integ›
›ration
   /bin/snapd-desktop-integration (1216) snap.snapd-desktop-integration.snapd-desktop-inte›
›gration
  0 processes are in complain mode.
  0 processes are unconfined but have a profile defined.
  0 processes are in mixed mode.
  0 processes are in kill mode.
```

--json

AppArmor プロファイルの状態を、JSON 形式で表示する。

実行例

```
sudo aa-status --json
```

実行結果

```
{"version": "2", "profiles": {"/snap/snapd/19993/usr/lib/snapd/snap-confine": "enforce",›
› "/snap/snapd/19993/usr/lib/snapd/snap-confine//mount-namespace-capture-helper": "enforce"›
›, "/snap/snapd/20092/usr/lib/snapd/snap-confine": "enforce", "/snap/snapd/20092/usr/lib/sn›
›apd/snap-confine//mount-namespace-capture-helper": "enforce", "/snap/snapd/20290/usr/lib/s›
›napd/snap-confine": "enforce", "/snap/snapd/20290/usr/lib/snapd/snap-confine//mount-namesp›
›ace-capture-helper": "enforce", "/usr/bin/evince": "enforce", "/usr/bin/evince-previewer":›
› "enforce", "/usr/bin/evince-previewer//sanitized_helper": "enforce", "/usr/bin/evince-thu›
›mbnailer": "enforce", "/usr/bin/evince//sanitized_helper": "enforce", "/usr/bin/man": "enf›
›orce", "/usr/lib/NetworkManager/nm-dhcp-client.action": "enforce", "/usr/lib/NetworkManage›
›r/nm-dhcp-helper": "enforce", "/usr/lib/connman/scripts/dhclient-script": "enforce", "/usr›
›/lib/cups/backend/cups-pdf": "enforce", "/usr/lib/snapd/snap-confine": "enforce", "/usr/li›
›b/snapd/snap-confine//mount-namespace-capture-helper": "enforce", "/usr/sbin/cups-browsed"›
›: "enforce", "/usr/sbin/cupsd": "enforce", "/usr/sbin/cupsd//third_party": "enforce", "/{,›
›usr/}sbin/dhclient": "enforce", "docker-default": "enforce", "libreoffice-senddoc": "enfor›
›ce", "libreoffice-soffice//gpg": "enforce", "libreoffice-xpdfimport": "enforce", "lsb_rele›
›ase": "enforce", "man_filter": "enforce", "man_groff": "enforce", "nvidia_modprobe": "enfo›
›rce", "nvidia_modprobe//kmod": "enforce", "snap-update-ns.firefox": "enforce", "snap-updat›
›e-ns.snap-store": "enforce", "snap-update-ns.snapd-desktop-integration": "enforce", "snap.›
›firefox.firefox": "enforce", "snap.firefox.geckodriver": "enforce", "snap.firefox.hook.con›
›figure": "enforce", "snap.firefox.hook.connect-plug-host-hunspell": "enforce", "snap.firef›
›ox.hook.disconnect-plug-host-hunspell": "enforce", "snap.firefox.hook.post-refresh": "enfo›
›rce", "snap.snap-store.hook.configure": "enforce", "snap.snap-store.snap-store": "enforce"›
›, "snap.snap-store.ubuntu-software": "enforce", "snap.snap-store.ubuntu-software-local-fil›
›e": "enforce", "snap.snapd-desktop-integration.hook.configure": "enforce", "snap.snapd-des›
›ktop-integration.snapd-desktop-integration": "enforce", "tcpdump": "enforce", "libreoffice›
›-oosplash": "complain", "libreoffice-soffice": "complain"}, "processes": {"/bin/snap-store›
›": [{"profile": "snap.snap-store.ubuntu-software", "pid": "1413", "status": "enforce"}], "›
›/bin/snapd-desktop-integration": [{"profile": "snap.snapd-desktop-integration.snapd-deskto›
›p-integration", "pid": "872", "status": "enforce"}, {"profile": "snap.snapd-desktop-integr›
›ation.snapd-desktop-integration", "pid": "1216", "status": "enforce"}], "/usr/bin/less": [›
›{"profile": "/usr/bin/man", "pid": "94000", "status": "enforce"}], "/usr/bin/man": [{"prof›
›ile": "/usr/bin/man", "pid": "93992", "status": "enforce"}], "/usr/sbin/cups-browsed": [{"›
›profile": "/usr/sbin/cups-browsed", "pid": "94258", "status": "enforce"}], "/usr/sbin/cups›
›d": [{"profile": "/usr/sbin/cupsd", "pid": "94221", "status": "enforce"}]}}
```

--pretty-json

AppArmor プロファイルの状態を、人間にも読みやすい形式で表示する。

実行例

```
sudo aa-status --pretty-json
```

実行結果


```
   {
          "version":      "2",
          "profiles":     {
                  "/snap/snapd/19993/usr/lib/snapd/snap-confine": "enforce",
                  "/snap/snapd/19993/usr/lib/snapd/snap-confine//mount-namespace-capture-h›
›elper": "enforce",
                  "/snap/snapd/20092/usr/lib/snapd/snap-confine": "enforce",
                  "/snap/snapd/20092/usr/lib/snapd/snap-confine//mount-namespace-capture-h›
›elper": "enforce",
                  "/snap/snapd/20290/usr/lib/snapd/snap-confine": "enforce",
                  "/snap/snapd/20290/usr/lib/snapd/snap-confine//mount-namespace-capture-h›
›elper": "enforce",
                  "/usr/bin/evince":      "enforce",
                  "/usr/bin/evince-previewer":    "enforce",
                  "/usr/bin/evince-previewer//sanitized_helper":  "enforce",
                  "/usr/bin/evince-thumbnailer":  "enforce",
                  "/usr/bin/evince//sanitized_helper":    "enforce",
                  "/usr/bin/man": "enforce",
                  "/usr/lib/NetworkManager/nm-dhcp-client.action":        "enforce",
                  "/usr/lib/NetworkManager/nm-dhcp-helper":       "enforce",
                  "/usr/lib/connman/scripts/dhclient-script":     "enforce",
                  "/usr/lib/cups/backend/cups-pdf":       "enforce",
                  "/usr/lib/snapd/snap-confine":  "enforce",
                  "/usr/lib/snapd/snap-confine//mount-namespace-capture-helper":  "enforce›
›",
                  "/usr/sbin/cups-browsed":       "enforce",
                  "/usr/sbin/cupsd":      "enforce",
                  "/usr/sbin/cupsd//third_party": "enforce",
                  "/{,usr/}sbin/dhclient":        "enforce",
                  "docker-default":       "enforce",
                  "libreoffice-senddoc":  "enforce",
                  "libreoffice-soffice//gpg":     "enforce",
                  "libreoffice-xpdfimport":       "enforce",
                  "lsb_release":  "enforce",
                  "man_filter":   "enforce",
                  "man_groff":    "enforce",
                  "nvidia_modprobe":      "enforce",
                  "nvidia_modprobe//kmod":        "enforce",
                  "snap-update-ns.firefox":       "enforce",
                  "snap-update-ns.snap-store":    "enforce",
                  "snap-update-ns.snapd-desktop-integration":     "enforce",
                  "snap.firefox.firefox": "enforce",
                  "snap.firefox.geckodriver":     "enforce",
                  "snap.firefox.hook.configure":  "enforce",
                  "snap.firefox.hook.connect-plug-host-hunspell": "enforce",
                  "snap.firefox.hook.disconnect-plug-host-hunspell":      "enforce",
                  "snap.firefox.hook.post-refresh":       "enforce",
                  "snap.snap-store.hook.configure":       "enforce",
                  "snap.snap-store.snap-store":   "enforce",
                  "snap.snap-store.ubuntu-software":      "enforce",
                  "snap.snap-store.ubuntu-software-local-file":   "enforce",
                  "snap.snapd-desktop-integration.hook.configure":        "enforce",
                  "snap.snapd-desktop-integration.snapd-desktop-integration":     "enforce›
›",
                  "tcpdump":      "enforce",
                  "libreoffice-oosplash": "complain",
                  "libreoffice-soffice":  "complain"
          },
          "processes":    {
                  "/bin/snap-store":      [{
                                  "profile":      "snap.snap-store.ubuntu-software",
                                  "pid":  "1413",
                                  "status":       "enforce"
                          }],
                  "/bin/snapd-desktop-integration":       [{
                                  "profile":      "snap.snapd-desktop-integration.snapd-de›
›sktop-integration",
                                  "pid":  "872",
                                  "status":       "enforce"
                          }, {
                                  "profile":      "snap.snapd-desktop-integration.snapd-de›
›sktop-integration",
```

```
                                "pid":  "1216",
                                "status":       "enforce"
                        }],
                "/usr/bin/less":        [{
                                "profile":      "/usr/bin/man",
                                "pid":  "94000",
                                "status":       "enforce"
                        }],
                "/usr/bin/man": [{
                                "profile":      "/usr/bin/man",
                                "pid":  "93992",
                                "status":       "enforce"
                        }],
                "/usr/sbin/cups-browsed":       [{
                                "profile":      "/usr/sbin/cups-browsed",
                                "pid":  "94258",
                                "status":       "enforce"
                        }],
                "/usr/sbin/cupsd":      [{
                                "profile":      "/usr/sbin/cupsd",
                                "pid":  "94221",
                                "status":       "enforce"
                        }]
        }
}
```

1.4 aa-teardown

全ての AppArmor(Linux のセキュリティプロファイルを結びつけることで、ネットワークアクセスや Raw socket アクセス、ファイルへの読み書き実行などの機能に制限をかけることができるプログラム。例えば、あるプログラムに対して、参照しかしないファイルへの書き込み、利用するはずがないファイルへのアクセス、関係がない別のプログラムの呼び出しを禁止できる。"/etc/apparmor/parser.conf"に記述された通りに制限がかけられる。) プロファイルを無効化する。

実行例

```
sudo aa-teardown
```

実行結果

```
Unloading AppArmor profiles
```

1.5 accessdb

Linux のマニュアルページのデータベース内のデータを人間が読める形式で出力するコマンド。デフォルトでは/var/cache/man/index.<db-type>からデータを出力

するが、引数を指定すると上書きできる。

実行例

```
accessdb
```

実行結果

```
$version$ -> "2.5.0"
.ldaprc -> "- 5 5 1690841590 0 C ldap.conf - gz "
/etc/adduser.conf -> "- 5 5 1609913810 0 C adduser.conf - gz "
/etc/anacrontab -> "- 5 5 1648028411 0 C anacrontab - gz "
/etc/deluser.conf -> "- 5 5 1609913810 0 C deluser.conf - gz "
/etc/mailcap.order -> "- 5 5 1639178040 0 C mailcap.order - gz "
/etc/modules -> "- 5 5 1629191993 0 C modules - gz "
00-upstream-settings -> "- 5 5 1649592997 0 A - - gz dconf configuration file"

以下マニュアルの一覧がアルファベット順に表示される。
```

1.6 apparmor_status

AppArmor(Linux のセキュリティプロファイルを結びつけることで、ネットワークアクセスや Raw socket アクセス、ファイルへの読み書き実行などの機能に制限をかけることができるプログラム。例えば、あるプログラムに対して、参照しかしないファイルへの書き込み、利用するはずがないファイルへのアクセス、関係がない別のプログラムの呼び出しを禁止できる。"/etc/apparmor/parser.conf"に記述された通りに制限がかけられる。) の状態を表示するコマンド。デフォルトでは、--verbose オプションと同じ情報が表示される。実行には root 権限が必要。

実行例

```
sudo apparmor_status
```

実行結果

```
apparmor module is loaded.
19 profiles are loaded.
19 profiles are in enforce mode.
   /snap/snapd/20290/usr/lib/snapd/snap-confine
   /snap/snapd/20290/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
   /snap/snapd/20671/usr/lib/snapd/snap-confine
   /snap/snapd/20671/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
   snap-update-ns.firefox
   snap-update-ns.snap-store
   snap-update-ns.snapd-desktop-integration
   snap.firefox.firefox
   snap.firefox.geckodriver
   snap.firefox.hook.configure
   snap.firefox.hook.connect-plug-host-hunspell
   snap.firefox.hook.disconnect-plug-host-hunspell
```

```
   snap.firefox.hook.post-refresh
   snap.snap-store.hook.configure
   snap.snap-store.snap-store
   snap.snap-store.ubuntu-software
   snap.snap-store.ubuntu-software-local-file
   snap.snapd-desktop-integration.hook.configure
   snap.snapd-desktop-integration.snapd-desktop-integration
0 profiles are in complain mode.
0 profiles are in kill mode.
0 profiles are in unconfined mode.
0 processes have profiles defined.
0 processes are in enforce mode.
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
0 processes are in mixed mode.
0 processes are in kill mode.
```

♣ オプション一覧

--profiled

読み込まれた AppArmor プロファイル (AppArmor を利用して設定された権限の情報が書かれたファイル) の数を表示する。

実行例

```
sudo apparmor_status --profiled
```

実行結果

```
19
```

--enforced

読み込まれた強制 AppArmor プロファイルの数を表示する。強制 AppArmor プロファイルは受けている制限が強制されるとともに、違反の試行が記録される。

実行例

```
sudo apparmor_status --enforced
```

実行結果

```
19
```

--complaining

読み込まれた苦情 AppArmor プロファイルの数を表示する。苦情 AppArmor プロファイルは受けている制限が強制されないが、違反の試行は記録される。

実行例

```
sudo apparmor_status --complaining
```

実行結果

```
0
```

--kill

違反時にタスクを強制終了する強制 AppArmor プロファイルを読み込み、数を表示する。

実行例

```
sudo apparmor_status --kill
```

実行結果

```
0
```

--special-unconfined

制限を受けていないモードである、非強制 AppArmor プロファイルを読み込み、数を表示する。

実行例

```
sudo apparmor_status --special-unconfined
```

実行結果

```
0
```

--process-mixed

異なるモードのプロファイルを持つプロファイルスタックによって制限されたプロセスの数を表示する。

実行例

```
sudo apparmor_status --process-mixed
```

実行結果

```
0
```

--verbose

AppArmor プロファイルの状態を表示する

実行例

```
sudo apparmor_status --verbose
```

実行結果

```
apparmor module is loaded.
19 profiles are loaded.
19 profiles are in enforce mode.
   /snap/snapd/20290/usr/lib/snapd/snap-confine
   /snap/snapd/20290/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
   /snap/snapd/20671/usr/lib/snapd/snap-confine
   /snap/snapd/20671/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
   snap-update-ns.firefox
   snap-update-ns.snap-store
   snap-update-ns.snapd-desktop-integration
   snap.firefox.firefox
   snap.firefox.geckodriver
   snap.firefox.hook.configure
   snap.firefox.hook.connect-plug-host-hunspell
   snap.firefox.hook.disconnect-plug-host-hunspell
   snap.firefox.hook.post-refresh
   snap.snap-store.hook.configure
   snap.snap-store.snap-store
   snap.snap-store.ubuntu-software
   snap.snap-store.ubuntu-software-local-file
   snap.snapd-desktop-integration.hook.configure
   snap.snapd-desktop-integration.snapd-desktop-integration
0 profiles are in complain mode.
0 profiles are in kill mode.
0 profiles are in unconfined mode.
0 processes have profiles defined.
0 processes are in enforce mode.
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
0 processes are in mixed mode.
0 processes are in kill mode.
```

--json

AppArmor プロファイルの状態を、JSON 形式で表示する。

実行例

```
sudo apparmor_status --json
```

実行結果

```
   {"version": "2", "profiles": {"/snap/snapd/20290/usr/lib/snapd/snap-confine": "enforce",
 "/snap/snapd/20290/usr/lib/snapd/snap-confine//mount-namespace-capture-helper": "enforce"
, "/snap/snapd/20671/usr/lib/snapd/snap-confine": "enforce", "/snap/snapd/20671/usr/lib/sn
apd/snap-confine//mount-namespace-capture-helper": "enforce", "snap-update-ns.firefox": "e
nforce", "snap-update-ns.snap-store": "enforce", "snap-update-ns.snapd-desktop-integration
": "enforce", "snap.firefox.firefox": "enforce", "snap.firefox.geckodriver": "enforce", "s
nap.firefox.hook.configure": "enforce", "snap.firefox.hook.connect-plug-host-hunspell": "e
nforce", "snap.firefox.hook.disconnect-plug-host-hunspell": "enforce", "snap.firefox.hook.
post-refresh": "enforce", "snap.snap-store.hook.configure": "enforce", "snap.snap-store.sn
ap-store": "enforce", "snap.snap-store.ubuntu-software": "enforce", "snap.snap-store.ubunt
u-software-local-file": "enforce", "snap.snapd-desktop-integration.hook.configure": "enfor
ce", "snap.snapd-desktop-integration.snapd-desktop-integration": "enforce"}, "processes":
{}}
```

--pretty-json

AppArmor プロファイルの状態を、人間にも読みやすい形式で表示する。

実行例

```
sudo apparmor_status --pretty-json
```

実行結果

```
{
        "version":      "2",
        "profiles":     {
                "/snap/snapd/20290/usr/lib/snapd/snap-confine": "enforce",
                "/snap/snapd/20290/usr/lib/snapd/snap-confine//mount-namespace-capture-h
elper": "enforce",
                "/snap/snapd/20671/usr/lib/snapd/snap-confine": "enforce",
                "/snap/snapd/20671/usr/lib/snapd/snap-confine//mount-namespace-capture-h
elper": "enforce",
                "snap-update-ns.firefox":       "enforce",
                "snap-update-ns.snap-store":    "enforce",
                "snap-update-ns.snapd-desktop-integration":     "enforce",
                "snap.firefox.firefox": "enforce",
                "snap.firefox.geckodriver":     "enforce",
                "snap.firefox.hook.configure":  "enforce",
                "snap.firefox.hook.connect-plug-host-hunspell": "enforce",
                "snap.firefox.hook.disconnect-plug-host-hunspell":      "enforce",
                "snap.firefox.hook.post-refresh":       "enforce",
                "snap.snap-store.hook.configure":       "enforce",
                "snap.snap-store.snap-store":   "enforce",
                "snap.snap-store.ubuntu-software":      "enforce",
                "snap.snap-store.ubuntu-software-local-file":   "enforce",
                "snap.snapd-desktop-integration.hook.configure":        "enforce",
                "snap.snapd-desktop-integration.snapd-desktop-integration":     "enforce
"
        },
        "processes":    {
        }
}
```

1.7　apt-mark

パッケージを自動/手動インストール済みとマークできる。自動/手動インストール済みとは、パッケージをインストールすると要求されるもので、別のパッケージが依存関係を満たすためにインストールされた場合に依存関係に自動インストール済みとマークされる。自動的にインストールされたパッケージは手動でインストールしたパッケージに依存されなくなると、apt-get や aptitude により削除が提案される。

実行例

```
なし
```

実行結果

```
なし
```

♣ オプション一覧

auto

パッケージを自動インストール済みとマークする。このパッケージに依存する手動でインストールされたパッケージがなくなると、このパッケージを削除する。

実行例

```
sudo apt-mark auto nano
```

実行結果

```
nano は自動でインストールしたと設定されました。
```

manual

パッケージを手動インストール済みとマークする。このパッケージに依存するほかのパッケージがなくなっても、このパッケージを自動的に削除するのを防ぐ。

実行例

```
sudo apt-mark manual nano
```

実行結果

```
nano は手動でインストールしたと設定されました。
```

showauto

手動でインストールされたパッケージを、パッケージごとに改行して表示する。パッケージを指定しないと、手動でインストールされたパッケージをすべて一覧表示する。パッケージを指定すると、そのパッケージが手動でインストールされている場合に表示する。

実行例

```
sudo apt-mark showauto
```

実行結果

```
accountsservice
acl
acpi-support
acpid
~略~
```

showmanual

自動的にインストールされたパッケージを、パッケージごとに改行して表示する。パッケージを指定しないと、自動的にインストールされたパッケージをすべて一覧表示する。パッケージを指定すると、そのパッケージが自動的にインストールされている場合に表示する。

実行例

```
sudo apt-mark showmanual
```

実行結果

```
apparmor-notify
apparmor-profiles
apparmor-profiles-extra
apparmor-utils
~略~
```

hold

パッケージを保留としてマークする。パッケージを自動的なインストール、アップグレード、削除から防ぐ。

実行例

```
sudo apt-mark hold nano
```

実行結果

```
nano は保留に設定されました。
```

unhold

以前保留したパッケージを、また操作できるようキャンセルするのに使用する。

実行例

```
sudo apt-mark unhold nano
```

実行結果

```
nano の保留を解除しました。
```

showhold

保留されたパッケージを、パッケージごとに改行して表示する。パッケージを指定しないと、保留されたパッケージをすべて一覧表示する。パッケージを指定すると、そのパッケージが保留されている場合に表示する。

実行例

```
sudo apt-mark showhold
```

実行結果

```
nano
```

1.8　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]
```

1.9　aptdcon

ソフトウェアのインストールや削除などのパッケージ管理タスクを実行できる。

実行例

```
なし
```

実行結果

```
なし
```

♣ オプション一覧

-i もしくは --install

パッケージをインストールする。複数のパッケージをインストールしたい場合は、パッケージ名を""で囲う。

実行例

```
sudo aptdcon -i "nano init"
```

実行結果

```
[\]   0% 依存関係を解決しています
[\]   0% ソフトウェアの一覧を読み込んでいます
[/]   0% 処理の開始を待っています
The following NEW packages will be installed (2):
  init nano
Do you want to continue [Y/n]?y
[-]   0% 認証を待っています
[-]   0% タスクを実行中です
[-]  11% タスクを実行中です
[+] 100%  成功しました
```

--reinstall

パッケージを再インストールする。複数のパッケージを再インストールしたい場合は、パッケージ名を""で囲う。

実行例

```
sudo aptdcon --reinstall "nano init"
```

実行結果

```
[\]   0% 依存関係を解決しています
[\]   0% ソフトウェアの一覧を読み込んでいます
[/]   0% 処理の開始を待っています
The following packages will be reinstalled (2):
  init nano
Do you want to continue [Y/n]?y
[-]   0% 認証を待っています
[-]   0% タスクを実行中です
[-]  11% タスクを実行中です
[+] 100%  成功しました
```

-r もしくは --remove

パッケージを削除する。複数のパッケージを削除したい場合は、パッケージ名を""で囲う。remove の場合設定ファイルは残るが、purge の場合は設定ファイルも削除される。

実行例

```
sudo aptdcon -r nano
```

実行結果

```
[\]   0% 依存関係を解決しています
[\]   0% ソフトウェアの一覧を読み込んでいます
[/]   0% 処理の開始を待っています
The following package will be REMOVED (1):
  nano
After this operation, -880640.0B of additional disk space will be freed.
Do you want to continue [Y/n]?y
[-]   0% 認証を待っています
[-]   0% タスクを実行中です
[-]  11% タスクを実行中です
[-]  11% ダウンロード中です
[-]  11% ダウンロード中です
[-]  50% ダウンロード中です
[-]  50% 変更を適用しています
(データベースを読み込んでいます ... 現在 253199 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
。)
nano (6.2-1) を削除しています ...
update-alternatives: using /usr/bin/vim.tiny to provide /usr/bin/editor (editor) in auto
mode
install-info (6.8-4build1) のトリガを処理しています ...
man-db (2.10.2-1) のトリガを処理しています ...
[\]  76% 終了しました インストール後トリガ dpkg-exec を実行しています
[\] 100% 終了しました インストール後トリガ dpkg-exec を実行しています
[\] 100% 終了しました インストール後トリガ dpkg-exec を実行しています
[\] 100% 終了しました インストール後トリガ dpkg-exec を実行しています
[\] 100% 終了しました インストール後トリガ dpkg-exec を実行しています
/usr/lib/python3/dist-packages/aptdaemon/console.py:337: Warning: Source ID 22 was not f
ound when attempting to remove it
  GLib.source_remove(wid)
/usr/lib/python3/dist-packages/aptdaemon/console.py:337: Warning: Source ID 23 was not f
ound when attempting to remove it
  GLib.source_remove(wid)
[+] 100%  成功しました
```

-p もしくは --purge

パッケージを完全に削除する。複数のパッケージを完全に削除したい場合は、パッケージ名を""で囲う。remove の場合設定ファイルは残るが、purge の場合は設定ファイルも削除される。

実行例

```
sudo aptdcon -p nano
```

実行結果

```
[\]   0% 依存関係を解決しています
[\]   0% ソフトウェアの一覧を読み込んでいます
[/]   0% 処理の開始を待っています
The following packages will be REMOVED (2):
  nano nano
After this operation, -880640.0B of additional disk space will be freed.
Do you want to continue [Y/n]?y
[-]   0% 認証を待っています
[-]   0% タスクを実行中です
[-]  11% タスクを実行中です
```

```
   [-]  11% ダウンロード中です
   [-]  11% ダウンロード中です
   [-]  50% ダウンロード中です
   [-]  50% 変更を適用しています
   (データベースを読み込んでいます ... 現在 253199 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
› 。)
   nano (6.2-1) を削除しています ...
   update-alternatives: using /usr/bin/vim.tiny to provide /usr/bin/editor (editor) in auto
› mode
   install-info (6.8-4build1) のトリガを処理しています ...
   man-db (2.10.2-1) のトリガを処理しています ...
   (データベースを読み込んでいます ... 現在 253127 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
› 。)
   nano (6.2-1) の設定ファイルを削除しています ...
   [\]  82% 終了しました インストール後トリガ dpkg-exec を実行しています
   [\] 100% 終了しました インストール後トリガ dpkg-exec を実行しています
   [\] 100% 終了しました インストール後トリガ dpkg-exec を実行しています
   [\] 100% 終了しました インストール後トリガ dpkg-exec を実行しています
   [\] 100% 終了しました インストール後トリガ dpkg-exec を実行しています
   /usr/lib/python3/dist-packages/aptdaemon/console.py:337: Warning: Source ID 22 was not f
›ound when attempting to remove it
     GLib.source_remove(wid)
   /usr/lib/python3/dist-packages/aptdaemon/console.py:337: Warning: Source ID 23 was not f
›ound when attempting to remove it
     GLib.source_remove(wid)
   [+] 100%  成功しました
```

-u もしくは --upgrade

パッケージをアップグレードする。複数のパッケージをアップグレードしたい場合は、パッケージ名を""で囲う。

実行例

```
   sudo aptdcon -u "nano init"
```

実行結果

```
   [\]   0% 依存関係を解決しています
   [\]   0% ソフトウェアの一覧を読み込んでいます
   [/]   0% 処理の開始を待っています
   The following packages will be upgraded (2):
     init nano
   Do you want to continue [Y/n]?y
   [-]   0% 認証を待っています
   [-]   0% タスクを実行中です
   [-]  11% タスクを実行中です
   [+] 100%  成功しました
```

--upgrade-system

システム全体をアップグレードする。

実行例

```
   sudo aptdcon --upgrade-system
```

実行結果

```
   [\]   0% 依存関係を解決しています
   [\]   0% ソフトウェアの一覧を読み込んでいます
   [/]   0% 依存関係を解決しています
   [-]   0% 処理の開始を待っています
   The following packages will be upgraded (7):
     linux-firmware python3-software-properties python3-update-manager software-properties-›
›common software-properties-gtk update-manager-core update-manager
   The following packages have been kept back (3):
     dnsmasq-base gjs libgjs0g
   Need to get 251.6MB of archives.
   After this operation, 19.9MB of additional disk space will be used.
   Do you want to continue [Y/n]?y
   [\]   0% 認証を待っています
   [\]   0% タスクを実行中です
   [\]  11% タスクを実行中です
   [\]  11% ダウンロード中です
   [\]  11% ダウンロード中です (Downloaded 0.0B of 251.6MB)
   ~略~
```

--fix-install

"dpkg --configure -a"を呼び出し、以前にキャンセルされたインストールを実行する。

実行例

```
   sudo aptdcon --fix-install nano
```

実行結果

```
   [\]   9% 依存関係を解決しています
   [\]   9% 処理の開始を待っています
   [\]   9% 認証を待っています
   [\]   9% タスクを実行中です
   [\]  11% タスクを実行中です
   [\]  11% クリーンアップ中です
   [\]  11% 変更を適用しています
   [\]  11% ソフトウェアの一覧を読み込んでいます
   [\]   1% ソフトウェアの一覧を読み込んでいます
   /usr/lib/python3/dist-packages/aptdaemon/console.py:337: Warning: Source ID 20 was not f›
›ound when attempting to remove it
     GLib.source_remove(wid)
   /usr/lib/python3/dist-packages/aptdaemon/console.py:337: Warning: Source ID 21 was not f›
›ound when attempting to remove it
     GLib.source_remove(wid)
   [+] 100%  成功しました
```

第 2 章
頭文字が b のコマンド

2.1 basename

ファイルのパスからファイル名の前についたディレクトリ構造を削除し、ファイル名を表示する。

実行例

```
basename /home/kawai/testdir/testdir1/test.txt
```

実行結果

```
test.txt
```

♣ オプション一覧

-a

複数のパスを引数として渡し、複数のファイル名を順に表示する。

実行例

```
basename -a /home/kawai/testdir/testdir1/test /home/kawai/testdir/testdir1/test.txt
```

実行結果

```
test
test.txt
```

-s

ファイル名の接尾辞を指定し、削除する。

実行例

```
basename -s .txt /home/kawai/testdir/testdir1/test.txt
```

実行結果

```
test
```

-z

出力の区切り文字を改行でなく NULL にする。

実行例

```
basename -z -a /home/kawai/testdir/testdir1/test /home/kawai/testdir/testdir1/test.txt
```

実行結果

```
testtest.txt
```

第 3 章
頭文字が c のコマンド

3.1 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
```

3.2 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。
```

3.3　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
```

3.4 chown

ファイルやディレクトリの所有者（ユーザー）を変更するために利用する。chown コマンドは通常、ルートユーザーまたはファイルの現在の所有者で実行する必要がある。

実行例

```
$ chown -v new_user text.txt
'text.txt' の所有者を old_user から new_user へ変更しました
```

実行例

```
$ chown -v new_user:new_user text.txt
'text.txt' の所有者を old_user:old_user から new_user:new_user へ変更しました
```

♣ 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変更内容を出力する。

実行例

```
$ chown -v new_user text.txt
```

実行結果

```
'text.txt' の所有者を old_user から new_user 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

```
$ chown -c new_user text.txt
```

実行結果

```
'text.txt' の所有者を old_user から new_user 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

```
$ chown -f new_user text.txt
```

実行結果

エラーメッセージ例

- 権限がない場合

```
$ chown new_user text.txt
chown: 'text.txt' の所有者を変更中: 許可されていない操作です
```

- 対象のファイルやディレクトリが存在しない場合

```
$ chown new_user t.txt
chown: 'text.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
```

- 存在しないユーザーやグループを指定した場合

```
$ chown user text.txt
chown: ユーザー指定が不正: 'user'
```

-h もしくは --no-dereference

指定したファイルがシンボリックリンクであるとき、そのシンボリックリンクの所有者を変更する。このオプションが指定されていない場合は、そのシンボリックリンクによって参照されるファイルの所有者が変更される。

- シンボリックリンク

Unix および Unix 系オペレーティングシステムで使用されるファイルやディレクトリへの参照。シンボリックリンクは、別のファイルやディレクトリを指す「参照」のようなもので、通常のファイルとは異なり、単なるポインタのような存在である。

```
.
├──chown
│   ├──text.txt
│
├──ln
│   ├──text
```

実行例

```
$ chown -v -h new_user text
```

実行結果

```
'text' の所有者を old_user から new_user へ変更しました
```

-R

ファイルとディレクトリを再帰的に変更する。

```
.
├──chown
│    ├──new_text.txt
│    ├──text.txt
```

実行例

```
$ chown -v -R new_user chown
```

実行結果

```
'chown/text.txt' の所有者を old_user から new_user へ変更しました
'chown/new_text.txt' の所有者を old_user から new_user へ変更しました
'chown' の所有者を old_user から new_user へ変更しました
```

第 4 章
頭文字が d のコマンド

4.1 date

与えられたフォーマットで現在時刻を表示するか、システム日付を設定する。
長いオプションに必須の引数は、短いオプションにも必須である。

実行例

```
date
```

実行結果

```
2023年 12月21日 木曜日 13:30:13 JST
```

♣ オプション一覧

-d もしくは --date=STRING

STRING で指定された時刻、を表示する。

実行例

```
date -d "2023-09-10"
date --date=tomorrow
```

実行結果

```
2023年 9月 10日 日曜日 00:00:00 JST
2023年  12月 22日 金曜日 13:32:22 JST
```

--debug

解析された日付に注釈を付け、疑わしい使い方については標準エラー出力に警告を出す。

実行例

```
date --debug
```

実行結果

```
2023年 12月21日 木曜日 14:50:19 JST
```

-f もしくは --file=DATEFILE

"--date"と同様

DATEFILE の各行に対して 1 回ずつ

実行例

```
date -f time.txt
```

実行結果

```
2023年 12月 22日 金曜日 05:00:00 JST
```

-I'FMT' もしくは --iso-8601=FMT

ISO 8601 形式の日付/時刻を出力する

フォーマットは'date':日付のみ (デフォルト)、'hours':時、'minutes':分、'seconds':秒または'ns' ナノ秒で表す

実行例

```
date -I'date'
date -I'ns' -d tomorrow
```

実行結果

```
2023-12-21
2023-12-22T15:14:26,204385695+09:00
```

-R もしくは --rfc-email

RFC 5322 形式の日付と時刻を出力する

実行例

```
date -R
```

実行結果

```
Fri, 22 Dec 2023 22:42:39 +0900
```

-r もしくは --reference=FILE

FILE の最終更新時刻を表示

実行例

```
date -r time.txt
```

実行結果

```
2023年 12月 22日 金曜日 22:41:36 JST
```

-s もしくは --set=STRING

STRING で指定された時刻を設定する

実行例

```
sudo date -s "2023-01-01"
```

実行結果

```
2023年 1月 1日 日曜日 00:00:00 JST
```

-u もしくは --utc もしくは --universal

協定世界時 (UTC) を表示または設定する

実行例

```
date -u
```

実行結果

```
2023年 12月 22日 金曜日 13:49:10 UTC
```

フォーマットについて

"+%"の形で書くことのできる表示形式のこと

%%

% を文字として表す

実行例

```
date "+%%"
```

実行結果

```
%
```

%a

地域の短縮形の曜日名

実行例

```
date "+%%-%a"
```

実行結果

```
%-金
```

%A

地域の完全な曜日名

実行例

```
date "%%-%a-%A"
```

実行結果

```
%-金-金曜日
```

%b

地元の短縮形の月名

実行例

```
date "%b"
```

実行結果

```
12月
```

%B

地域の完全な月名

実行例

```
date "%b-%B"
```

実行結果

```
12月-12月
```

%c

地域の日付と時刻

実行例

```
date "+%c"
```

実行結果

```
2023年12月22日 22時57分50秒
```

%C

世紀; 下 2 桁を省略する以外は %Y と同じ

実行例

```
date "+%C"
```

実行結果

```
20
```

%d

日付

実行例

```
date "+%d"
```

実行結果

```
22
```

%D

日付;%m/%d/%y

実行例

```
date "+%D"
```

実行結果

```
12/22/23
```

%e

%d と同様

スペースパッドされた日

スペースパッドとは、文字列の周りにスペースや他の文字を追加して、特定のフォーマットや整列を実現するために使われる

スペースを使って余白を埋めることを指している

実行例

```
date "+%e"
```

実行結果

```
22
```

%F

完全な日付を表示する

"%+4Y-%m-%d"と同様

実行例

```
date "+%F"
```

実行結果

```
2023-12-22
```

%g

ISO 週番号の年の下二桁

詳細は %G を参照

実行例

```
date "+%g"
```

実行結果

```
23
```

%G

ISO 週番号の年
詳細は %V を参照
%V のみより使用頻度が高い

実行例

```
date "+%G"
```

実行結果

```
2023
```

%h

%b と同じ

実行例

```
date "+%h"
```

実行結果

```
12月
```

%H

時間 (00..23)

実行例

```
date "+%H"
```

実行結果

```
23
```

%I

時間 (01..12)

実行例

```
date "+%I"
```

実行結果

```
11
```

%j

日にち (001..366)

実行例

```
date "+%j"
```

実行結果

```
356
```

%k

時間、スペース・パッド (0..23);%_H と同じ

実行例

```
date "+%k"
```

実行結果

```
23
```

%l

時間、スペース・パッド (0..23)

実行例

```
date "+%l"
```

実行結果

```
11
```

%m

月 (01..12)

実行例

```
date "+%m"
```

実行結果

```
12
```

%M

分 (00.59)

実行例

```
date "+%M"
```

実行結果

```
07
```

%n

改行する

実行例

```
date "+%M%n%m"
```

実行結果

```
07
12
```

%N

ナノ秒 (000000000. . 999999999)

実行例

```
date "+%N"
```

実行結果

```
407200293
```

%p

地域の AM または PM に相当

実行例

```
date "+%p"
```

実行結果

```
午後
```

%P

%p のようなものだが、小文字
日本語では変化なし

実行例

```
date "+%P"
```

実行結果

```
午後
```

%q

年の四半期 (1..4)

実行例

```
date "+%q"
```

実行結果

```
4
```

%r

地域の 12 時間時計の時刻

実行例

```
date "+%r"
```

実行結果

```
午後11時10分38秒
```

%R

24 時間制の時間と分

実行例

```
date "+%R"
```

実行結果

```
23:11
```

%s

1970-01-01 00:00:00 UTC からの秒数

実行例

```
date "+%s"
```

実行結果

```
1703254359
```

%S

秒 (00..60)

実行例

```
date "+%S"
```

実行結果

```
08
```

%t

タブ

実行例

```
date "+%S%t%s"
```

実行結果

```
42    1703254422
```

%T

時間

%H:%M:%s と同じ

実行例

```
date "+%T"
```

実行結果

```
23:14:19
```

%u

曜日 (1..7)
1 は月曜日

実行例

```
date "+%u"
```

実行結果

```
5
```

%U

週番号 (00.53)
日曜日を週の最初の曜日とする

実行例

```
date "+%U"
```

実行結果

```
51
```

%V

ISO 週番号 (01..53)
月曜日が週の初日
1 は月曜日

実行例

```
date "+%V"
```

実行結果

```
51
```

%w

曜日 (0..6)
0 は日曜日

実行例

```
date "+%w"
```

実行結果

```
5
```

%W

年の週番号 (00..53)
月曜日を週の最初の曜日とする

実行例

```
date "+%W"
```

実行結果

```
51
```

%x

地域の日付表現

実行例

```
date "+%x"
```

実行結果

```
2023年12月22日
```

%X

地域の時刻表限

実行例

```
date "+%X"
```

実行結果

```
23時18分20秒
```

%y

年の下 2 桁 (00..99)

実行例

```
date "+%y"
```

実行結果

```
23
```

%Y

西暦の年

実行例

```
date "+%Y"
```

実行結果

```
2023
```

%z

+hhmm 数字タイムゾーン

実行例

```
date "+%z"
```

実行結果

```
+0900
```

%:z

+hh:mm 数値タイムゾーン

実行例

```
date "+%:z"
```

実行結果

```
+09:00
```

%::z

+hh:mm:ss 数値タイムゾーン

実行例

```
date "+%::z"
```

実行結果

```
+09:00:00
```

%:::z

必要な精度の数値タイムゾーン

実行例

```
date "+%:::z"
```

実行結果

```
+09
```

%Z

アルファベットによるタイムゾーンの省略形

実行例

```
date "+%Z"
```

実行結果

```
JST
```

4.2　dir

指定されたディレクトリ内のファイルとサブディレクトリのリストを一覧表示できる

オプション一部一覧

-a もしくは -all

隠しファイルを含む全てのファイルを表示する

実行例

```
dir -a
dir -all
```

実行結果

```
.       ..      file1.txt   .hidden_file   directory1   .hidden_directory
```

-A もしくは --almost-all もしくは --author

"."と".."は除く"."で始まる隠しファイルを含む全てのファイルを表示する

実行例

```
dir -A
dir --almost-all
```

実行結果

```
file1.txt   .hidden_file   directory1   .hidden_directory
```

-b もしくは --escape もしくは --block-size=SIZE

エスケープ文字を表示する

実行例

```
dir -b
dir --escape
```

実行結果

```
file1.txt  file2.txt  directory1  directory2
```

-B もしくは --ignore-backups

ファイルの最後に「〜（チルダ）」がついたバックアップファイルを表示しない

実行例

```
dir -B
dir --ignore-backups
```

実行結果

```
file1.txt  file2.txt  directory1  directory2
```

-c

変更日時 (ctime) の新しい順でソートする

実行例

```
dir -c
```

実行結果

```
file3.txt    ctime: Dec 5 2023 12:30
file1.txt    ctime: Dec 3 2023 09:15
directory2   ctime: Dec 1 2023 16:45
directory1   ctime: Nov 30 2023 08:20
```

-color

カラー表示にする
ディレクトリは青色
実行可能なファイルは緑色
一般的なファイルは白色

実行例

```
dir -color
```

実行結果

```
  directory  file1.txt
```

-d もしくは --directory

ディレクトリそのものの情報を表示する

実行例

```
dir -d
dir --directory
```

実行結果

```
.
```

-i

i ノード番号を表示

実行例

```
dir -i
```

実行結果

```
9851624185227536 directory
7881299348254011 file.txt
```

-l

詳細な情報を表示する

実行例

```
dir -l
```

実行結果

```
drwxr-xr-x 1 name 197609 0 Dec  6 23:55 file.txt
drwxr-xr-x 1 name 197609 0 Dec  23 11:32 directory
```

-t

ファイル更新日時でソートして表示する

実行例

```
dir -t
```

実行結果

```
file3.txt  file1.txt directory2  directory1
```

- **-S**

ファイルサイズが大きいものからソートする

実行例

```
dir -S
```

実行結果

```
file1.txt  file2.txt  directory  directory2
```

- **-1**

リストを縦１列で表示する

実行例

```
dir -1
```

実行結果

```
directory
directory2
file1.txt
file4.txt
file2.txt
```

4.3 dircolors

Linux システム上のディレクトリとファイルの表示色を設定する

♣ オプション一覧

-b もしくは --sh もしくは --bourne-shell

Bourne シェル形式で LS_COLORS を出力する

実行例

```
dircolors -b
```

実行結果

```
LS_COLORS='rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;
01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=
01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:
*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01
;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz
2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31
:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpi
o=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:
*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.
pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=0
1;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35
:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.webp=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.m
p4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;
35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd
=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:
*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.og
g=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:';
  export LS_COLORS
```

-c もしくは --csh もしくは --c-shell

C シェル形式で LS __ COLORS を出力する

実行例

```
実行例
```

実行結果

```
実行結果
```

-p もしくは --print-database

デフォルト値を標準入力に表示する

実行例

```
実行例
```

実行結果

```
実行結果
```

4.4 dirname

指定されたパスからディレクトリの部分を取得する

♣ オプション一覧

-z

ヌル文字（\0）を区切り文字として使用して出力する
この出力形式は、他のコマンドと組み合わせて使用する際に便利

実行例

```
find . -name "file.txt" -print0 | xargs -0 dirname -z
```

実行結果

```
./path/to
"file.txt"を見つけ、ディレクトリパスをヌル文字で区切って出力する
```

4.5 dmesg

Linux カーネルが起動時に出力したメッセージを表示するコマンド
システム起動時のトラブルシューティングやデバイスの接続情報の確認、デバッグ情報の収集に用いることができる
カーネルの操作のためルート権限が必要

実行例

```
sudo dmesg
```

実行結果（以下一部のみ）

```
   [0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-6.2.0-37-generic root=UUID=b784ca7c-86
›02-4f1c-8436-8089242f37f3 ro quiet splash
```

♣ オプション一覧

-l

表示レベルを指定する

「emerg(高)」「alert」「crit」「err」「warn」「notice」「info」「debug(低)」から選択、複数指定は","で区切る

実行例

```
   sudo dmesg -l warn
```

実行結果

```
   [0.890430] acpi PNPOA03:00: fail to add MMCONFIG information, can't access extended conf
›iguration space under this bridge
   [2.420791] device-mapper: core: CONFIG_IMA_DISABLE_HTABLE is disabled. Duplicate IMA mea
›surements will not be recorded in the IMA log.
   [2.420872] platform eisa.0: EISA: Cannot allocate resource for mainboard
   [2.420873] platform eisa.0: Cannot allocate resource for EISA slot 1
   [2.420874] platform eisa.0: Cannot allocate resource for EISA slot 2
   [2.420875] platform eisa.0: Cannot allocate resource for EISA slot 3
   [2.420876] platform eisa.0: Cannot allocate resource for EISA slot 4
   [2.420876] platform eisa.0: Cannot allocate resource for EISA slot 5
   [2.420877] platform eisa.0: Cannot allocate resource for EISA slot 6
   [2.420878] platform eisa. 0: Cannot allocate resource for EISA slot 7
   [2.420879] platform eisa.0: Cannot allocate resource for EISA slot 8
   [20.683554] kauditd_printk_skb: 8 callbacks suppressed
   [25.849810] kauditd_printk_skb: 8 callbacks suppressed
   [61.011374] kauditd_printk_skb: 4 callbacks suppressed
```

-f

表示対象を指定する

「kern(カーネル)(-k オプション)」「user(-u オプション)」「mail」「daemon」「auth」「syslog」「lpr」「news」から選択、複数指定は","で区切る

実行例

```
   sudo dmesg -f kern
```

実行結果

```
   [0.000000] Linux version 6.2.0-37-generic (buildd@bos03-amd64-055) (x86_64-linux-gnu-gcc
›-11 (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0, GNU ld (GNU Binutils forUbuntu) 2.38) #38~22.04
›.1-Ubuntu SMP PREEMPT_DYNAMIC Thu Nov 2 18:01:13 UTC 2 (Ubuntu 6.2.0-37.38~22.04.1-generic
› 6.2.16)
   [0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-6.2.0-37-generic root=UUID=b784ca7c-86
›02-4f1c-8436-8089242f37f3 ro quiet splash
   [0.000000] KERNEL supported cpus:
   Intel GenuineIntel
```

```
   AMD AuthenticAMD
   Hygon HygonGenuine
   Centaur CentaurHaulszhaoxin
   [0.000000] BIOS-provided physical RAM map:
```

-x

表示レベルと表示対象を表示する

実行例

```
   sudo dmesg -x
```

実行結果

```
   kern :notice: [0.000000] Linux version 6.2.0-37-generic (buildd@bos03-amd64-055) (x86_64›
›-linux-gnu-gcc-11 (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0, GNU ld (GNU Binutils for Ubuntu) ›
›2.38) #38~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Thu Nov 2 18:01:13 UTC 2 (Ubuntu 6.2.0-37.38~›
›22.04.1-generic 6.2.16)
   kern :info : [0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-6.2.0-37-generic root=UUI›
›D=b784ca7c-8602-4f1c-8436-8089242f37f3 ro quiet splash
   kern:info : [0.000000] KERNEL supported cpus:
   kern:info : [0.000000]Intel GenuineIntel
   kern:info : [0.000000]AMD AuthenticAMD
   kern:info : [0.000000]Hygon HygonGenuine
   kern:info : [0.000000]Centaur CentaurHauls
   kern:info : [0.000000]zhaoxin  Shanghai
   kern:info : [0.000000] BIOS-provided physical RAM map:
```

-d

直前のメッセージからの経過時間を表示する

実行例

```
   sudo dmesg -d
```

実行結果

```
   [  86.241961 <0.000447>] audit: type=1400 audit(1702012655.625:77): apparmor="DENIED" op›
›eration="capable" class="cap" profile="/snap/snapd/19457/usr/lib/snapd/snap-confine" pid=1›
›850 comm="snap-confine" capability=38 capname="perfmon"
```

-e

メッセージの表示時刻とメッセージ間の経過時間を表示する

実行例

```
   sudo dmesg -e
```

実行結果

```
   [+0.000447] audit: type=1400 audit(1702012655.625:77): apparmor="DENIED" operati
   on="capable" class="cap" profile="/snap/snapd/19457/usr/lib/snapd/snap-confine" pid=1850
 comm="snap-confine" capability=38 capname="perfmon"
```

-T

メッセージが出力された時刻で表示する

実行例

```
   sudo dmesg -T
```

実行結果

```
   [金 12月 8 14:17:35 2023] audit: type=1400 audit(1702012655.625:77): apparmor="DENIED" o
peration="capable" class="cap" profile="/snap/snapd/19457/usr/lib/snapd/snap-confine" pid=
1850 comm="snap-confine" capability=38 capname="perfmon"
```

-t

時間を表示しない

実行例

```
   sudo dmesg -t
```

実行結果

```
   audit: type=1400 audit(1702012655. 625:77): apparmor="DENIED" operation="capable" class=
"cap" profile="/snap/snapd/19457/usr/lib/snapd/snap-confine" pid=1850 comm="snap-confine"
capability=38 capname="perfmon"
```

-H

読みやすいスタイルで表示する

実行例

```
   sudo dmesg -H
```

実行結果

```
   [+0.000002] DMA [mem 0x0000000000001000-0x0000000000ffffff]
   [+0.000003] DMA32 [mem 0x0000000001000000-0x00000000ffffffff]
   [+0.000001] Normal
   [+0.000002] [mem 0x0000000100000000-0x000000011fffffff]
   [+0.000001] Movable zone start for each node
```

-r

メッセージを加工せずに表示する

実行例

```
  sudo dmesg -r
```

実行結果

```
  <5>[86.241961] audit: type=1400 audit(1702012655.625:77) : apparmor="DENIED" operation="
›capable" class="cap" profile="/snap/snapd/19457/usr/lib/snapd/snap-confine" pid=1850 comm=
›"snap-confine" capability=38 capname="perfmon"
```

-L

色付きで表示する
出力に変化がないので割愛

-w

新しいメッセージがカーネルから出力されるのを待つ

実行例

```
  sudo dmesg -w
```

実行結果

```
  [ 2664.473779] audit: type=1326 audit(1702015233.848:78): auid=1000 uid=1000 gid=1000 se
›s=2 subj=snap.firefox.firefox pid=2429 comm="firefox" exe="/snap/firefox/2987/usr/lib/fire
›fox/firefox" sig=0 arch=c000003e syscall=314 compat=0 ip=0x7f14f03e973d code=0x50000
```

-F

バッファーの代わりに指定したファイルを表示する

実行例

```
  sudo dmesg -F file.txt
```

実行結果

```
  [   0.000000] Hello
```

-S

syslog を使って表示する

実行例

```
  sudo dmesg -S
```

実行結果

```
   [86.241961] audit: type=1400 audit(1702012655.625:77): apparmor="DENIED" operation="capa›
›ble" class="cap" profile="/snap/snapd/19457/usr/lib/snapd/snap-confine" pid=1850 comm="sna›
›p-confine" capability=38 capname="perfmon"
```

-C

バッファーをクリアする

実行例

```
   sudo dmesg -C
   sudo dmesg
```

実行結果

```
   何も表示されなくなる
```

-c

リンクバッファーを出力してクリアする

実行例

```
   sudo dmesg -c
   sudo dmesg
```

実行結果

```
   sudo dmesg -c
   [ 3082.092138]audit:type=1326 audit(1702015651.467:79): auid=1000 uid=1000 gid=1000 ses=›
›2 subj=snap.firefox.firefox pid=3081 comm="firefox" exe="/snap/firefox/2987/usr/lib/firefo›
›x/firefox" sig=0 arch=c000003e syscall=314 compat=0 ip=0x7efc049f373d code=0x50000
   sudo dmesg
   何も表示されない
```

-D

コンソールへの出力を無効にする

実行例

```
   sudo dmesg -D
```

実行結果

```
   なし
```

-E

コンソールへの出力を有効にする

実行例

```
sudo dmesg -E
```

実行結果

```
なし
```

-n

コンソールに表示するレベルを指定する

実行例

```
sudo dmesg -n alert
```

実行結果

```
なし
```

4.6 dmidecode

ハードウェアの情報を表示するコマンド DMI テーブルに格納されている情報を見やすい形に成型して表示する

ハードウェアの特定と確認、BIOS の情報確認、メモリの詳細な情報の取得、サーバー管理とトラブルシューティングに利用できる詳細を見る場合はルート権限

実行例

```
sudo dmidecode
```

実行結果（一部）

```
  # dmidecode 3.3
Getting SMBIOS data from sysfs.
SMBIOS 2.5 present.
10 structures occupying 456 bytes.
Table at 0x000E1000.
```

♣ オプション一覧

-d

情報元のファイルを指定する
(デフォルトは/dev/main)

実行例

```
sudo dmidecode -d /dev/main
```

実行結果

```
上記と同様
```

-q

文章説明

実行例

```
実行例
```

実行結果

```
BIOS Information
   Vendor: innotek GmbH
   Version: VirtualBox
   Release Date: 12/01/2006
   Address: 0xE0000
   Runtime Size: 128 kB
   ROM Size: 128 kB
   Characteristics:
        ISA is supported
        PCI is supported
        Boot from CD is supported
        Selectable boot is supported
        8042 keyboard services are  supported (int 9h)
        CGA/mono video services are supported (int 10h)
        ACPI is supported
```

-s

指定したキーワードの情報だけを表示する

指定できるキーワードの一覧は「dmidecode -s」で確認可能

実行例

```
dmidecode -s

sudo dmidecode -s bios-vendor
```

実行結果

```
dmidecode: option requires an argument -- 's'
String keyword expected
Valid string keywords are:
bios-vendor
bios-version
bios-release-date
bios-revision
firmware-revision
system-manufacturer
```

```
system-product-name
system-version
system-serial-number

innotek GmbH
```

-t

指定したタイプの情報だけを表示する

指定できるタイプの一覧は「dmidecode -t」で確認可能

実行例

```
dmidecode -t

sudo dmidecode -t chassis
```

実行結果

```
dmidecode: option requires an argument -- 't'
Type number or keyword expected
Valid type keywords are:
bios
system
baseboard
chassis
processor
memor  y
cache
connector
slot


# dmidecode 3.3
Getting SMBIOS data from sysfs.
SMBIOS 2.5 present.

Handle 0x0003, DMI type 3, 13 bytes
Chassis Information
Manufacturer: Oracle Corporation
Type: Other
Lock: Not Present
Version: Not Specified
Serial Number: Not Specified
Asset Tag: Not Specified
Boot-up State: Safe
Power Supply State: Safe
Thermal State: Safe
Security Status: None
```

-H

指定した CLI ハンドルの情報だけを表示する

実行例

```
実行例
```

実行結果

```
# dmidecode 3.3Getting SMBIOS data from sysfs.
SMBIOS 2.5 present.
10 structures occupying 456 bytes.
Table at 0x000E1000.
Handle 0x0003, DMI type 3, 13 byte
Chassis Information
Manufacturer: Oracle Corporation
Type: Other
Lock: Not Present
Version: Not Specified
Serial Number: Not Specified
Asset Tag: Not Specified
Boot-up State: Safe
Power Supply State: Safe
Thermal State: Safe
Security Status: None
```

-u

情報をデコードせず 16 進数コードのまま表示する

実行例

```
sudo dmidecode -u
```

実行結果

```
# dmidecode 3.3
Getting SMBIOS data from sysfs.
SMBIOS 2.5 present.
10 structures occupying 456 bytes.
Table at 0x000E1000.
Handle 0x0000, DMI type 0, 20 bytes
Header and Data:
00 14 00 00 01 02 00 E0 03 01 90 80 01 48 00 00 00 00 01 00
Strings:
69 6E 6E 6F 74 65 6B 20 47 6D 62 48 00
innotek GmbH
56 69 72 74 75 61 6C 42 6F 78 00
VirtualBox
31 32 2F 30 31 2F 32 30 30 36 00
12/01/2006
```

4.7 dnsdomainname

ドメイン名を表示する

システムの設定やネットワークにより静的に設定されるため使用頻度は低い

実行例

```
dnssomainname
```

実行結果

```
表示されない
```

♣ オプション一覧

-a

ドメインネームの別名も表示する

実行例

```
dnsdomainname -a
```

実行結果

```
表示されない
```

-A

全てのホスト名を FQDN(ドメイン名を省略) で表示する

実行例

```
dnsdomainname -A
```

実行結果

```
ubuntu-VirtualBox
```

-b

設定したホスト名を、ずっと有効にする

実行例

```
dnsdomainname -b
```

実行結果

```
表示なし
```

-d

DNS のドメイン名を表示する
詳細表示しようとすると FQDN と誤解されるため使用頻度は低い

実行例

```
dnsdomainname -d
```

実行結果

```
表示なし
```

-f

ホスト名を FQDN で表示する

実行例

```
dnsdomainname -f
```

実行結果

```
ubuntu-VitualBox
```

-F

ホスト名をファイルから読み込む

実行例

```
dnsdomainname -F example.com
dnsdomainname
```

実行結果

```
example.com
```

-i

ホスト名に対応する IP アドレスを表示する
このオプションが使えない場合は代わりに「-I」オプションを使う

実行例

```
dnsdomainname
```

実行結果

```
127.0.1.1
```

-I

設定されているすべてのホスト名に対応する IP アドレスを表示する

実行例

```
dnsdomainname -I
```

実行結果

```
10.0.2.15
```

-s

短縮表現のホスト名を表示する

実行例

```
dnsdomainname -s
```

実行結果

```
ubuntu-VitualBox
```

-y

NIS ドメイン名 (NIS の仕組みで使われるドメイン名) を表示・設定する

実行例

```
sudo dnsdomainname -y domainname
domainname
```

実行結果

```
domainname
```

4.8　do-release-upgrade

新しい Ubuntu がリリースされた際に最新のバージョンにアップグレードするために使用今回はアップグレードしないため Prompt を変更しない

Prompt はアップグレード中に表示される質問や確認のことを示す

通常ルート権限で実行する

実行例

```
sudo do-release-upgrade
```

実行結果

```
  新しい ubuntu のリリースをチェックしています
入手可能なLTSの開発版はありません。
最新の非LTS開発リリースにアップグレードする
/etc/update-manager/release-upgradesにPrompt=normalとセットしてください。
```

♣ オプション一覧

-d

サポートされている最新のリリースを使用している場合、開発版リリースにアップグレードする

実行例

```
sudo do-release-upgrade -d
```

実行結果

```
新しい ubuntu のリリースをチェックしています
アップグレードの前に入手可能なすべてのアップデートをインストールしてください。
```

-p

アップグレードソフトウェアを使って$distro-proposed から最新のリリースへのアップグレードを試す

実行例

```
sudo do-release-upgrade -p
```

実行結果

```
新しい ubuntu のリリースをチェックしています
入手可能なLTSの開発版はありません。
最新の非LTS開発リリースにアップグレードする
/etc/update-manager/release-upgradesにPrompt=normalとセットしてください。
```

-m

特別なアップグレードモードを実行する。現在、デスクトップシステムの標準的なアップグレードを行う

'desktop' または'server' オプションがある

実行例

```
sudo do-release-upgrade -m desktop
```

実行結果

```
新しい ubuntu のリリースをチェックしています
入手可能なLTSの開発版はありません。
最新の非LTS開発リリースにアップグレードする
/etc/update-manager/release-upgradesにPrompt=normalとセットしてください。
```

-f

特定のフロントエンドで実行

非対話型のアップグレードになり、自動処理されるため、アップグレードがスムーズに進行される場合にのみ、このオプションを使用するべき

実行例

```
sudo do-release-upgrade -f server
```

実行結果

```
新しい ubuntu のリリースをチェックしています
入手可能なLTSの開発版はありません。
最新の非LTS開発リリースにアップグレードする
/etc/update-manager/release-upgradesにPrompt=normalとセットしてください。
```

-c

新しいディストリビューション・リリースが利用可能かどうかチェックし、終了コードで結果を通知する

実行例

```
sudo do-release-upgrade -c
```

実行結果

```
新しい ubuntu のリリースをチェックしています
入手可能なLTSの開発版はありません。
最新の非LTS開発リリースにアップグレードする
/etc/update-manager/release-upgradesにPrompt=normalとセットしてください。
```

--data-dir

データファイルの含まれるディレクトリ

実行例

```
sudo -do-release-upgrade --data-dir desktop
```

実行結果

```
新しい ubuntu のリリースをチェックしています
入手可能なLTSの開発版はありません。
最新の非LTS開発リリースにアップグレードする
/etc/update-manager/release-upgradesにPrompt=normalとセットしてください。
```

--allow-third-party

サードパーティのミラーとリポジトリをコメントアウトする代わりに有効にしてアップグレードを試す

実行例

```
sudo do-release-upgrade --allow-third-party
```

実行結果

```
新しい ubuntu のリリースをチェックしています
入手可能なLTSの開発版はありません。
最新の非LTS開発リリースにアップグレードする
/etc/update-manager/release-upgradesにPrompt=normalとセットしてください。
```

-q

アップグレードプロセスの出力が最小限に抑えられ、進行状況や詳細な情報が表示されない

実行例

```
sudo do-release-upgrade -q
```

実行結果

```
表示されない
```

-e

アップグレードプロセスがエクスプレスモードで実行され、エクスプレスモードでは、ユーザの介入なしでアップグレードが進行し、デフォルトの設定がされる

実行例

```
sudo do-release-upgrade -e desktop
```

実行結果

```
新しい ubuntu のリリースをチェックしています
入手可能なLTSの開発版はありません。
最新の非LTS開発リリースにアップグレードする
/etc/update-manager/release-upgradesにPrompt=normalとセットしてください。
```

4.9 domainname

ネットワーキングやドメイン関連の設定を行う際に使用される

しかし、シンプルな構文と変更の即時変更の観点から hostname コマンドが推奨される場合が多いまた、dnsdomainname とも互換性がある

実行例

```
domainname
```

実行結果

```
domainname
```

♣ オプション一覧

-a

ドメイン名の別名も表示する

実行例

```
domainname -a
```

実行結果

```
表示されない
```

-A

全てのホスト名を FQDN(ドメイン名を省略) で表示する

実行例

```
domainname -A
```

実行結果

```
ubuntu-VirtualBox
```

-b

設定したホスト名を、ずっと有効にする

実行例

```
domainname -b
```

実行結果

```
domainname
```

-d

DNS のドメイン名を表示する
詳細表示しようとすると FQDN と誤解されるため使用頻度は低い

実行例

```
domainname -d
```

実行結果

```
domainname -d
```

-f

ホスト名を FQDN で表示する

実行例

```
domainname -f
```

実行結果

```
ubuntu-VirtualBox
```

-F

ホスト名をファイルから読み込む

実行例

```
sudo domainname -F file.txt
domainname
```

実行結果

```
Hello
```

-i

ホスト名に対応する IP アドレスを表示する
このオプションが使えない場合は代わりに「-I」オプションを使う

実行例

```
domainname -i
```

実行結果

```
127.0.1.1
```

-I

設定されているすべてのホスト名に対応する IP アドレスを表示する

実行例

```
domainname -I
```

実行結果

```
10.0.2.15
```

-s

短縮表現のホスト名を表示する

実行例

```
domainname -s
```

実行結果

```
ubuntu-VitualBox
```

-y

NIS ドメイン名 (NIS の仕組みで使われるドメイン名) を表示・設定する

実行例

```
sudo domainname -y domainname1
domainname
```

実行結果

```
domainname1
```

4.10 dosfsck

FAT ファイルシステムのシステムおよび修復を行うために使用されるファイルシステムの破損やエラーが発生したときに使用され、予期せぬシステムクラッシュ、異常なシャットダウン、メディアの障害などが原因で損傷が生じた場合に利用される。他のファイルシステムでは使用できないオプションなしの実行は不可

dosfsck とは"DOS File System Check"で DOS とはディスクオペレーションシステムであり、コマンドラインインターフェースを提供する

FAT とは"File Allocation Table"でストレージ内でファイルやディレクトリについての情報を記録する特殊なシステム領域と構築されるシステムのことである

♣ オプション一覧

-a

自動的にファイルシステムを修復する

実行例

```
dosfsck -a file.txt
```

実行結果

```
fsck.fat 4.2 (2021-01-31)
Got 6 bytes instead of 512 at 0
```

-A

Atari バリアントの FAT ファイルシステムを切り替える

実行例

```
dosfsck -A file.txt
```

実行結果

```
fsck.fat 4.2 (2021-01-31)
Got 6 bytes instead of 512 at 0
```

-b

読み取り専用のブートセクタチェックを行う

実行例

```
dosfsck -b file.txt
```

実行結果

```
fsck.fat 4.2 (2021-01-31)
Got 6 bytes instead of 512 at 0
```

-c

DOS コードページ N を使用して短いファイル名をデコードする (デフォルト:850)

実行例

```
実行例
```

実行結果

```
実行結果
```

-d

PATH のファイルを削除する

実行例

```
dosfsck -d home/usrname
```

実行結果

```
実行結果
```

-f

未使用のチェーンをファイルに回収

実行例

```
dosfsck -f file.txt
```

実行結果

```
fsck.fat 4.2 (2021-01-31)
Got 6 bytes instead of 512 at 0
```

-F

ファイルシステムアクセスに使用される FAT システムバリアントを指定する

バリアントによってビット幅の異なるクラスターエントリを備えたファイルシステムを使用する

実行例

```
dosfsck -F FAT32 /dev/sdb1
```

実行結果

```
dosfsck 3.0.13, 30 Jun 2019, FAT32, LFN
/dev/sdb1: 16 files, 1000/2000 clusters
```

-l

パス名をリストする

実行例

```
dosfsck -l file.txt
```

実行結果

```
fsck.fat 4.2 (2021-01-31)
Got 6 bytes instead of 512 at 0
```

-n

変更せずに対話的にファイルシステムをチェックする

実行例

```
dosfsck -n file.txt
```

実行結果

```
fsck.fat 4.2 (2021-01-31)
Got 6 bytes instead of 512 at 0
```

-p

他の fsck との互換性のための"-a"と同じ

プレテンスモードでファイルシステムのチェックを行わずにチェックを実行する前の予備情報を表示するモードで実行する

実行例

```
dosfsck -p file.txt
```

実行結果

```
fsck.fat 4.2 (2021-01-31)
Got 6 bytes instead of 512 at 0
```

-r

対話的にファイルシステムを修復する

実行例

```
dosfsck -r file.txt
```

実行結果

```
fsck.fat 4.2 (2021-01-31)
Got 6 bytes instead of 512 at 0
```

-S

短いファイル名の途中にスペースを許可しない

実行例

```
dosfsck -S file.txt
```

実行結果

```
fsck.fat 4.2 (2021-01-31)
Got 6 bytes instead of 512 at 0
```

-t

不良クラスタのテストを実行する

実行例

```
dosfsck -t file.txt
```

実行結果

```
fsck.fat 4.2 (2021-01-31)
Got 6 bytes instead of 512 at 0
```

-u

復元可能なディレクトリでないファイルを復元する

実行例

```
dosfsck -u file.txt
```

実行結果

```
表示なし
```

-U

ボリュームとブートラベルに大文字のみを許可する

実行例

```
dosfsck -U file.txt
```

実行結果

```
fsck.fat 4.2 (2021-01-31)
Got 6 bytes instead of 512 at 0
```

-v

詳細モードで実行する

実行例

```
dosfsck -v file.txt
```

実行結果

```
fsck.fat 4.2 (2021-01-31)
Got 6 bytes instead of 512 at 0
```

-V

検証パスを実行する

実行例

```
dosfsck -V oaif
```

実行結果

```
fsck.fat 4.2 (2021-01-31)
Logical sector size (58930 bytes) is not a multiple of the physical sector size
```

--variant

ファイルシステムのバリアント TYPE を処理する

実行例

```
dosfsck --variant=FAT32 /dev/sdb2
```

実行結果

```
dosfsck 3.0.13, 30 Jun 2019, FAT32, LFN
/dev/sdb1: 16 files, 1000/2000 clusters
```

-w

変更を直ちにディスクに書き込む

実行例

```
dosfsck -w file.txt
```

実行結果

```
fsck.fat 4.2 (2021-01-31)
Got 6 bytes instead of 512 at 0
```

-y

他の fsck との互換性のための-a と同じ

非対話モードで実行し、問題を検出すると自動的に修復が行われる

実行例

```
dosfsck -y file.txt
```

実行結果

```
fsck.fat 4.2 (2021-01-31)
Got 6 bytes instead of 512 at 0
```

4.11 dosfslabel

FAT ファイルシステムのラベルを設定、表示するボリューム名を変更、確認する必要がある場合に使用されるため使用頻度はあまり高くないボリューム名はリムーバブルメディアを他のデバイスと区別するために使用されることがある

実行例

```
sudo dosfslabel /dev/sda
```

実行結果

```
Logical sector size is zero.
```

♣ オプション一覧

-i

ラベルの代わりにシリアルナンバーを使い実行する

実行例

```
sudo dosfslabel -i /dev/sda
```

実行結果

```
Logical sector size is zero.
```

-r

ラベルを取り除くかシリアルナンバーを新しく生成する

実行例

```
sudo dosfslabel -r /dev/sda
```

実行結果

```
Logical sector size is zero.
```

-c

ラベルのエンコード/デコードに DOS コードページを使用する

実行例

```
sudo dosfslabel -c 800 /dev/sda
```

実行結果

```
Cannot initialize conversion from codepage 800 to UTF-8: Invalid argument
Cannot initialize conversion from UTF-8 to codepage 800: Invalid argument
```

4.12 du

ディスク上のディレクトリやファイルのディスク使用量を表示するために使用する

システム管理者や開発者など、ディスク使用量の監視やトラブルシューティングを行うため高頻度で使用される一般ユーザではディスク容量が不足している場合や特定のディレクトリやファイルのサイズを確認したい場合に使用されることがある

ディスク使用量を表示するため、大規模なディレクトリやファイルの場合では実行に時間がかかることがあり、正確な計算には適切なオプションが必要である

シンボリックリンクとはファイルシステムにおいて、特定のファイルシステムやディレクトリへの参照を作成するために使用される特殊なファイルファイルやディレクトリのパス名を指定し、そのパス名に対する参照を提供する

実行例

```
du file.txt
```

実行結果

```
4       file.txt
```

♣ オプション一覧

-0

各出力を改行ではなく NULL で終了させる

実行例

```
du -0
```

実行結果

```
   4  ./.cache/tracker3/files/errors14244  . /.cache/mesa_shader_cache/388   . /. cacache/1›
›e8   he/mesa_shader_cache/2512   ache/ce8  ache/6312   ・/ピクチャ4   he/tracker3/files1424›
›8   he/mesa_shader_cache/cf8  ache/1a8   he/mesa_shader_cache/138. /. cache/mesa_shader_ca›
›che/658 . /.cache/mesa_shader_cache/e28./. cac . /.cache/mesa_shader_cache/cc8 ./.cache/me›
›sa_shader_  . cache/mesa_shader_cache/998 ./. cache/mesa_shader_cache/448 ./.cache/mesa_sh›
›ader_cache/068  he/mesa_shader_cache/2e12 . /.cache/mesa_shader_cache/758   ・/ビデオ4  . /›
›. cache/tracker38  . /.cache/mesa_shader_cache/a48 . /.cache/mesa_shader_c . /. cache/mesa›
›_shader_cache/848 . /.cache/mesa_shader_cache/3f8  . /. cache/mesa_shader_cache/2912 . /. ›
›cache/mesa_shader_c  . /.cache/mesa_shader_cache/8d12 . /.cache/mesa_shader_c  . /. cache/›
›mesa_shader_cache/d68 ./.cache/mesa_shader_cache/708   ./. cac   . /. cache/mesa_shader_ca›
›che/268 . /.cache/mesa_shader_c   ./.cache/mesa_shader_cache/3c8 ./.cac
```

-a

ディレクトリだけでなく、すべてのファイルのカウントを表示する

実行例

```
du -a
```

実行結果

```
4     . /.config/mozc/ibus_config. textproto
424   ./.config/mozc
0     . /.config/ibus-mozc-gnome-initial-setup-done
4     . /.config/evolution/sources/system-proxy. source
8     . /.config/evolution/sources
12    . /.config/evolution
580   ./. config
97576 .
```

--apparent-size

ディスク使用量ではなく、表面のサイズを表示する

表面のサイズは通常は小さいが、ファイルのホール、内部の断片化、関節ブロックのために大きくなることがある

実行例

```
du --apparent-size
```

実行結果

```
73     . /.config/pulse
4      . /.config/update-notifier
397    . /.config/mozc
6      . /.config/evolution/sources
10     . /.config/evolution
515    . /.config
97162  .
```

-B

SIZE の倍率として表示する

"-BM"は 1,048,576 バイト単位でサイズを表示する

実行例

```
du -BM /home/ubuntu/.config
```

実行結果

```
1M  /home/ubuntu/.config/gtk-3.0
1M  /home/ubuntu/.config/goa-1.0
1M  /home/ubuntu/.config/ibus/bus
1M  /home/ubuntu/.config/ibus
1M  /home/ubuntu/.config/nautilus
1M  /home/ubuntu/.config/dconf
1M  /home/ubuntu/.config/pulse
```

-c

総合計量を表示する

実行例

```
du -c file.txt
```

実行結果

```
4     file.txt
4     合計
```

-D

コマンドラインで指定されたシンボリックのみをたどる

実行例

```
du -D /home/ubuntu/.config/evolution/
```

実行結果

```
8     /home/ubuntu/.config/evolution/sources
12    /home/ubuntu/.config/evolution/
```

-d

コマンド実行場所から,N 階層以内のディレクトリの合計を表示する

実行例

```
du -d 4 /home/ubuntu/.config
```

実行結果

```
8     /home/ubuntu/.config/gtk-3.0
4     /home/ubuntu/.config/goa-1.0
12    /home/ubuntu/.config/ibus/bus
16    /home/ubuntu/.config/ibus
4     /home/ubuntu/.config/nautilus
8     /home/ubuntu/.config/dconf
84    /home/ubuntu/.config/pulse
4     /home/ubuntu/.config/update-notifier
424   /home/ubuntu/.config/mozc
8     /home/ubuntu/.config/evolution/sources
12    /home/ubuntu/.config/evolution
580   /home/ubuntu/.config
```

-H

シンボリックを解決し、リンク先のファイルやディレクトリを操作するオプションと同じもの

実行例

```
du -H /home/ubuntu/file.txt
```

実行結果

```
4       /home/ubuntu/file.txt
```

-h

人に読みやすい形でサイズを表示する

実行例

```
du -h /home/ubuntu/.config
```

実行結果

```
4.0K    /home/ubuntu/.config/nautilus
8.0K    /home/ubuntu/.config/dconf
84K     /home/ubuntu/.config/pulse
4.0K    /home/ubuntu/.config/update-notifier
424K    /home/ubuntu/. config/mozc
8.0K    /home/ubuntu/.config/evolution/sources
12K     /home/ubuntu/.config/evolution
580K    /home/ubuntu/.config
```

-k

ブロックサイズを 1K 単位とした時と同じ

実行例

```
du -k /home/ubuntu/
```

実行結果

```
4.0K    /home/ubuntu/.config/nautilus
8.0K    /home/ubuntu/.config/dconf
84K     /home/ubuntu/.config/pulse
4.0K    /home/ubuntu/.config/update-notifier
424K    /home/ubuntu/. config/mozc
8.0K    /home/ubuntu/.config/evolution/sources
12K     /home/ubuntu/.config/evolution
580K    /home/ubuntu/.config
```

-L

全てのシンボリックリンクをたどらない

実行例

```
du -L /home/ubuntu/.config
```

実行結果

```
4       /home/ubuntu/.config/nautilus
8       /home/ubuntu/.config/dconf
84      /home/ubuntu/.config/pulse
4       /home/ubuntu/.config/update-notifier
424     /home/ubuntu/. config/mozc
8       /home/ubuntu/.config/evolution/sources
12      /home/ubuntu/.config/evolution
580     /home/ubuntu/.config
```

-l

ハードリンクされた場合その個数分サイズを数える

実行例

```
du -l /home/ubuntu/.config
```

実行結果

```
4       /home/ubuntu/.config/nautilus
8       /home/ubuntu/.config/dconf
84      /home/ubuntu/.config/pulse
4       /home/ubuntu/.config/update-notifier
424     /home/ubuntu/. config/mozc
8       /home/ubuntu/.config/evolution/sources
12      /home/ubuntu/.config/evolution
580     /home/ubuntu/.config
```

-m

ブロックサイズを 1M 単位とした時と同じ

実行例

```
du -m /home/ubuntu/.config
```

実行結果

```
1       /home/ubuntu/.config/nautilus
1       /home/ubuntu/.config/dconf
1       /home/ubuntu/.config/pulse
1       /home/ubuntu/.config/update-notifier
1       /home/ubuntu/. config/mozc
1       /home/ubuntu/.config/evolution/sources
1       /home/ubuntu/.config/evolution
1       /home/ubuntu/.config
```

-P

シンボリックリンクをたどらない

実行例

```
du -P /home/ubuntu/.config
```

実行結果

```
4       /home/ubuntu/.config/nautilus
8       /home/ubuntu/.config/dconf
84      /home/ubuntu/.config/pulse
4       /home/ubuntu/.config/update-notifier
424     /home/ubuntu/.config/mozc
8       /home/ubuntu/.config/evolution/sources
12      /home/ubuntu/.config/evolution
580     /home/ubuntu/.config
```

-S

子ディレクトリのサイズを含めない

実行例

```
du -S /home/ubuntu/.config
```

実行結果

```
4       /home/ubuntu/.config/nautilus
8       /home/ubuntu/.config/dconf
84      /home/ubuntu/.config/pulse
4       /home/ubuntu/.config/update-notifier
424     /home/ubuntu/. config/mozc
8       /home/ubuntu/.config/evolution/sources
4       /home/ubuntu/.config/evolution
16      /home/ubuntu/.config
```

-s

各引数の合計容量のみを表示する

実行例

```
du -s /home/ubuntu/.config
```

実行結果

```
580   /home/ubuntu/.config
```

-t

SIZE を指定して、SIZE が正の場合は SIZE より小さいエントリを無視する
SIZE が負の場合は SIZE より大きなエントリを無視する

実行例

```
du -t 10M
```

実行結果

```
10676  . /snap/firefox/common/.mozilla/firefox/3q93a0v8.default/storage/permanent
10820  . /snap/firefox/common/.mozilla/firefox/3q93a0v8.default/storage
28100  ./snap/firefox/common/.mozilla/firefox/3q93a0v8.default
28124  . /snap/firefox/common/.mozilla/firefox
28132  . /snap/firefox/common/.mozilla
80288  ./snap/firefox/common
80372  ./snap/firefox
80564  . /snap
97576  .
```

--time

ディレクトリとその子ディレクトリに含まれるすべてのファイルでの最終更新時間を表示する

"atime","access","use","ctime","status"で更新時間の代わりにその時間を表示できる

実行例

```
du --time=use file.txt
```

実行結果

```
4    2023-12-10 13:37    file.txt
```

--time-style

時間を STYLE の形式で表示する

STYLE には"full-iso","long-iso","iso","+FORMAT(date と同様)"を指定できる

実行例

```
du --time-style=full-iso file.txt
```

実行結果

```
4     file.txt
```

-X

FILE 内のいずれかのパターンに一致するファイルを除外する

実行例

```
du -X file.txt
```

実行結果

```
4     ./. config/nautilus
8     ./. config/dconf
84    ./. config/pulse
4     ./. config/update-notifier
424   ./. config/mozc
8     ./. config/evolution/sources
12    ./.config/evolution
580   ./.config
97576 .
```

-x

異なるファイルシステム上のディレクトリはスキップする

実行例

```
du -x /home/ubuntu/.config
```

実行結果

```
4       /home/ubuntu/.config/nautilus
8       /home/ubuntu/.config/dconf
84      /home/ubuntu/.config/pulse
4       /home/ubuntu/.config/update-notifier
424     /home/ubuntu/.config/mozc
8       /home/ubuntu/.config/evolution/sources
12      /home/ubuntu/.config/evolution
580     /home/ubuntu/.config
```

4.13 dumpe2fs

dump ext2/ext3/ext4 file system infomation

ext2/ext3/ext4 ファイルシステムのスーパーブロックの情報とブロックグループディスクリプタの内容を出力する

スーパーブロックにはファイルシステムの管理用情報が格納されている

オプションなしの実行は不可

♣ オプション一覧

-b

ファイルシステム内で不良としてマークされたブロックを表示する

実行例

```
sudo dumpe2fs -b /dev/sda1
```

実行結果

```
dumpe2fs 1.46.5 (30-Dec-2021)
dumpe2fs: Bad magic number in super-block while trying to open /dev/sda1
Couldn't find valid filesystem superblock.
```

-o

ファイルシステムを調査する際に、ブロックサイズのバイト単位でブロックを使用する

非常に損傷したファイルシステムの残留物を調査するファイルシステムの専門家以外は必要ないコマンド

実行例

```
sudo dumpe2fs -o superblock=10 /dev/sda2
```

実行結果

```
dumpe2fs 1.46.5 (30-Dec-2021)
dumpe2fs: Bad magic number in super-block while trying to open /dev/sda2
Couldn't find valid filesystem superblock.
/dev/sda2 contains a vfat file system
```

-f

dumpe2fs に一部のファイルシステム機能フラグが理解できない可能性がある場合でもファイルシステムを表示するように強制する

実行例

```
sudo dumpe2fs -f -o blocksize=5 /dev/sda1
```

実行結果

```
dumpe2fs 1.46.5 (30-Dec-2021)
dumpe2fs: Bad magic number in super-block while trying to open /dev/sda1
Couldn't find valid filesystem superblock.
```

-g

グループディスクリプタ情報を、機械可読なコロン区切りの値形式で表示する

グループ番号; グループ内の最初のブロック番号; スーパーブロックの位置; グループディスクリプタが使用するブロックの範囲; ブロックビットマップの位置;i ノードビットマップの位置; および i ノードテーブルが使用するブロック

の形である

実行例

```
sudo dumpe2fs -g -o blocksize=5 /dev/sda3
```

実行結果

```
dumpe2fs 1.46.5 (30-Dec-2021)
dumpe2fs: Filesystem has unexpected block size while trying to open /dev/sda3
Couldn't find valid filesystem superblock.
```

-h

ブロックグループディスクリプタの詳細情報を表示せずに、スーパーブロック情報のみを表示する

実行例

```
sudo dumpe2fs -h -o blocksize=5 /dev/sda3
```

実行結果

```
実行結果
```

-i

e2image が作成したイメージファイルからファイルシステムデータを表示し、パス名としてデバイスを使用する

実行例

```
sudo dumpe2fs -i -o superblock=3 /dev/sda1
```

実行結果

```
dempe2fs 1.46.5(30-Dec-2021)
Couldn't find valid filesystem superblock.
dumpe2fs: Wrong magic number for Ext2 Image Header while trying to open /dev/sda1
```

-m

もしファイルシステムで mmp 機能が有効になっている場合、デバイスが他のノードで使用されているかどうかを確認する

-i オプションと併用する場合、MMP ブロックの情報のみが表示される

実行例

```
sudo dumpe2fs -m -i -o superblock=3 /dev/sda1
```

実行結果

```
dempe2fs 1.46.5(30-Dec-2021)
Couldn't find valid filesystem superblock.
dumpe2fs: Bad magic number in super-block while trying to open /dev/sda1
```

-x

ブロックの詳細なグループ情報を 16 進数の形式で表示する

実行例

```
sudo dumpe2fs -x -o superblock=3 /dev/sda1
```

実行結果

```
dumpe2fs 1.46.5 (30-Dec-2021)
dumpe2fs: Bad magic number in super-block while trying to open /dev/sda1
Couldn't find valid filesystem superblock.
```

4.14 dumpkeys

キーボードドライバの変換ケーブルの現在の内容を指定された形式で標準出力に書き込む。

オプションにより、出力形式の制御やカーネルやプログラムから他の情報を取得することも可能

カーネルから情報を取得しているので管理者権限が必要

実行例

```
sudo dumpkeys
```

実行結果 (一部)

```
keymaps 0-127
keycode  1 = Escape
       alt    keycode 1 = Meta_Escape
       shift  alt    keycode  1 = Meta_Escape
       altgr  alt    keycode  1 = Meta_Escape
       shift  altgr  alt keycode  1 = Meta_Escape
       control alt    keycode  1 = Meta_Escape
       shift  control alt  keycode  1 = Meta_Escape
       altgr  control alt  keycode  1 = Meta_Escape
       shift  altgr   control alt  keycode  1 = Meta_Escape
```

♣ オプション一覧

-i もしくは --short-info

カーネルのキーボード・ドライバの特徴を表示する。

表示される項目

Keycode range supported by the kernel(カーネルがサポートするキーコードの範囲):キーコードの後にどのような値を使うことができるか指定する。

詳細は keymaps(5) を参照

Number of actions bindable to a key(キーにバインドできるアクションの数):ひとつのキーがさまざまな修飾キーを使っていくつの異なるアクションを出力できるかを示す。例えばこの値が 16 であれば、修飾キーと組み合わせて 1 つのキーに最大 16 種類のアクションを定義することができる。値が 16 の場合、カーネルはほぼ 4 つの修飾キーを知っており、キーと異なる組み合わせで押すことで、割り当てられたすべてのアクションにアクセスできる。

Ranges of action codes supported by kernel(カーネルがサポートするアクションコードの範囲):16 進数表記のアクションコード範囲のリストが含まれている。これらはキー定義の右側、行の vv で使用できる値

キーコード xx = VV VV VV VV

(キー定義行のフォーマットについての詳細は keymaps(5) を参照)。

dumpkeys(1) と loadkeys(1) はシンボリック表記をサポートしており、カーネルごとにアクションコードが異なる可能性がある一方で、シンボリック名は通常同じままであるため、数値表記よりもシンボリック表記を推奨している。アクションコードの範囲のリストは、カーネルが loadkeys(1) が知っているすべてのシンボルを実際にサポートしているかどうか、あるいは、loadkeys(1) プログラムではシンボル名を持たない、カーネルによってサポートされているいくつかのアクションがあるかどうかを決定するために使用することができる。これを見るには、この範囲リストとアクションシンボルリストを比較する。

Number of function keys supported by kernel(カーネルがサポートするファンクションキーの数):文字列の出力に使用できるアクションコードの数を示す。これらのアクションコードは伝統的にキーボードの様々なファンクションキーや編集キーにバインドされており、標準的なエスケープシーケンスを送信するように定義されている。しかし、一般的なコマンドラインや電子メールアドレスなど、好きなものを送信するようにこれらを再定義することができる。特に、この項目の数がキーボードのファンクションキーや編集キーの数より多い場合は、例えば AltGr と文字の組み合わせにバインドして、便利な文字列を送信できる「予備の」アクションコードがある可能性がある。

詳細は loadkeys(1) を参照

Function String(ファンクションの文字列):ファンクションキーの定義は次のコマンドで確認できる

dumpkeys --funcs-only

実行例

```
sudo dumpkeys -i
```

実行結果

```
keycode range supported by kernel:        1 - 255
max number of actions bindable to a key:      256
number of keymaps in actual use:      128
of which 121 dynamically allocated ranges of action codes supported by kernel:
      0x0000 - 0x00ff
      0x0100 - 0x01ff
      0x0200 - 0x0213
      0x0300 - 0x0313
      0x0400 - 0x041a
      0x0500 - 0x05ff
      0x0600 - 0x0603
      0x0700 - 0x0708
      0x0800 - 0x08ff
      0x0900 - 0x0919
      0x0a00 - 0x0a08
      0x0b00 - 0x0bff
      0x0c00 - 0x0c08
      0x0doo - 0x0dff
      0x0e00 - 0x0e0a
number of function keys supported by kernel: 256
max nr of compose definitions: 256
nr of compose definitions in actual use: 68
```

-l もしくは -s もしくは --long-info

長い情報リストを表示するよう dumpkeys に指示する。

出力は -- short-info の場合と同じで、 loadkeys(1) と dumpkeys(1) がサポ-トするアクションシンボルのリストと、 シンボルの数値が追加される。

実行例

```
sudo dumpkeys -l
```

実行結果 (一部)

```
keycode range supported by kernel:        1 - 255
max number of actions bindable to a key:      256
number of keymaps in actual use:      128
of which 121 dynamically allocated ranges of action codes supported by kernel:
      0x0000 - 0x00ff
      0x0100 - 0x01ff
      0x0200 - 0x0213
      0x0300 - 0x0313
      0x0400 - 0x041a
      0x0500 - 0x05ff
      0x0600 - 0x0603
      0x0700 - 0x0708
```

```
       0x0800 - 0x08ff
       0x0900 - 0x0919
       0x0a00 - 0x0a08
       0x0b00 - 0x0bff
       0x0c00 - 0x0c08
       0x0doo - 0x0dff
       0x0e00 - 0x0e0a
number of function keys supported by kernel: 256
max nr of compose definitions: 256
nr of compose definitions in actual use: 68
Symbols recognized by dumpkeys:
(numeric value, symbol)
```

-n もしくは --numeric

dumpkeys はアクションコードの値をシンボル表記に変換するのをバイパスし、代わりに 16 進数で表示する。

実行例

```
sudo dumpkeys -n
```

実行結果 (一部)

```
keymaps 0-127
keycode  1 = 0x001b
       alt    keycode 1 = 0x081b
       shift  alt    keycode  1 = 0x081b
       altgr  alt    keycode  1 = 0x081b
       shift  altgr  alt keycode  1 = 0x081b
       control alt    keycode  1 = 0x081b
       shift  control alt  keycode  1 = 0x081b
       altgr  control alt  keycode  1 = 0x081b
       shift  altgr   control alt  keycode  1 = 0x081b
```

-f もしくは --full-table

dumpkeys はすべてのショートハンドによるヒューリスティック (keymaps(5) を参照) をスキップし、キーバインドを正規の形式で出力する。最初に、現在定義されている修飾子の組み合わせを記述した keymaps 行が出力される。次に、各キーに対して、それぞれの修飾子の組み合わせの列を持つ行が出力される。例えば、現在使用されているキーマップが 7 つの修飾子を使用している場合、すべての行は 7 つのアクションコードの列を持つことになる。このフォーマットは、例えば dumpkeys の出力を後処理するプログラムで有用

実行例

```
sudo dumpkeys -f
```

実行結果 (一部)

```
   keymaps 0-127
   keycode  1 = Escape       Escape      Escape      Escape      Escape      Escape      Es›
›cape      Escape      Meta_Escape      Meta_Escape      Meta_Escape      Meta_Escape      ›
›Meta_Escape      Meta_Escape      Meta_Escape      Meta_Escape      Escape      Escape    ›
›  Escape      Escape      Escape      Escape      Escape      Escape      Meta_Escape
```

-S もしくは --shape

コンソールのキーボードレイアウトに関連するキーボードとシンボルの対応が表示される。

実行例

```
sudo dumpkeys -S=8
```

実行結果 (一部)

```
keymaps 0-127
keycode  1 = Escape
       alt    keycode 1 = Meta_Escape
       shift  alt    keycode  1 = Meta_Escape
       altgr  alt    keycode  1 = Meta_Escape
       shift  altgr  alt keycode  1 = Meta_Escape
       control alt    keycode  1 = Meta_Escape
       shift  control alt  keycode  1 = Meta_Escape
       altgr  control alt  keycode  1 = Meta_Escape
       shift  altgr   control alt  keycode  1 = Meta_Escape
```

-t もしくは --funcs-only

dumpkeys はファンクションキーの文字列定義のみを表示する。

通常 dumpkeys はキーバインディングと文字列定義の両方を表示する。

実行例

```
sudo dumpkeys -t
```

実行結果

```
表示なし
```

-k もしくは --keys-only

dumpkeys はキーバインドだけを表示する。

通常 dumpkeys はキーバインディングと文字列定義の両方を表示する。

実行例

```
実行例
```

実行結果 (一部)

```
keymaps 0-127
keycode  1 = Escape
       alt    keycode 1 = Meta_Escape
       shift  alt    keycode  1 = Meta_Escape
       altgr  alt    keycode  1 = Meta_Escape
       shift  altgr  alt keycode  1 = Meta_Escape
       control alt    keycode  1 = Meta_Escape
       shift  control alt  keycode  1 = Meta_Escape
       altgr  control alt  keycode  1 = Meta_Escape
       shift  altgr   control alt  keycode  1 = Meta_Escape
```

-d もしくは --compose-only

dumpkeys は compose キーの組み合わせだけを表示する。

カーネルがコンポーズ・キーをサポートしている場合にのみ利用できる。

compose キーとは特殊なキーで、複数キーを連続して押すことで特殊な文字が入力できる方法。

実行例

```
sudo dumpkeys -d
```

実行結果 (一部)

```
compose '`' 'A' to Agrave
compose '`' 'a' to agrave
compose '\' 'A' to Aacute
compose '\' 'a' to aacute
compose '^' 'A' to Acircumflex
compose '^' 'a' to acircumflex
compose '~' 'A' to Atilde
compose '~' 'a' to atilde
compose '"' 'A' to Adiaeresis
compose '"' 'a' to adiaeresis
```

-c もしくは --charset=charset

指定された文字セットに従って文字コード値を解釈するように dumpkeys に指示する。

文字コード値からシンボル名への変換にのみ影響する。

現在有効な文字コードの値は iso-8859-X である。

charset が指定されていない場合、iso-8859-1 がデフォルトとして使用される。

このオプションは出力行 ‘charset "iso-8859-x"’ を生成し、loadkeys にキーマップの解釈方法を伝える。

(例えば、"division "は iso-8859-1 では oxf7 だが、iso-8859-8 では oxba)。

実行例

```
sudo dumpkeys -c iso-10646-18
```

実行結果

```
charset "iso-10646-18"
keymaps 0-127
keycode  1 = Escape
       alt    keycode 1 = Meta_Escape
       shift  alt    keycode  1 = Meta_Escape
       altgr  alt    keycode  1 = Meta_Escape
       shift  altgr  alt keycode  1 = Meta_Escape
       control alt    keycode  1 = Meta_Escape
       shift  control alt  keycode  1 = Meta_Escape
       altgr  control alt  keycode  1 = Meta_Escape
       shift  altgr   control alt  keycode  1 = Meta_Escape
```

-v もしくは --verbose

dumpkeys がサポートしている総裁情報が表示される。

使用用途はキーボードの設定やキーマッピングに関する追加の詳細情報の確認である。

実行例

```
sudo dumpkeys -v
```

実行結果

```
keymaps 0-127
keycode  1 = Escape
       alt    keycode 1 = Meta_Escape
       shift  alt    keycode  1 = Meta_Escape
       altgr  alt    keycode  1 = Meta_Escape
       shift  altgr  alt keycode  1 = Meta_Escape
       control alt    keycode  1 = Meta_Escape
       shift  control alt  keycode  1 = Meta_Escape
       altgr  control alt  keycode  1 = Meta_Escape
       shift  altgr   control alt  keycode  1 = Meta_Escape
```

第 5 章
頭文字が e のコマンド

5.1 e2freefrag

ext2/3/4 ファイルシステムの空き領域の断片化を報告するために使われる。

ext2˜ext4 ファイルシステムとは Linux で標準的に利用されるファイルシステムの一つで、ext4 がよく利用され、オペレーティングシステムをサポートしている。

ブロックビットマップ情報をスキャンして、 どれだけの空きブロックが連続し、整列した空き領域として存在するかを調べる。

ブロックビットマップ情報ブロック毎に空き/使用中を管理している情報

実行例

```
sudo e2freefrag
```

実行結果

```
Device: /dev/sda1
Blocksize: 4096 bytes
Total blocks: 256
Free blocks: 238 (93.0%)


Min. free extent: 12 KB
Max. free extent: 856 KB
Avg. free extent: 236 KB
Num. free extent: 4


HISTOGRAM OF FREE EXTENT SIZES:
Extent Size Range : Free extents  Free Blocks Percent
    8K ...  16K-  :            1            3   1.26%
   16K ...  32K-  :            1            6   2.52%
   32K ...  64K-  :            1           15   6.30%
  512K ...1024K-  :            1          214  89.92%
```

♣ オプション一覧

-c

チャンクサイズが指定された場合、e2freefrag はチャンクサイズの空きチャンクの数をキロバイト単位で表示する

チャンクとはデータベースサーバのデータ格納専用の物理ディスクの最大単位

各チャンクの最大サイズは 4TB である

実行例

```
sudo e2freefrag -c 8 /dev/sda1
```

実行結果

```
Device: /dev/sda1
Blocksize: 4096 bytes
Total blocks: 256
Free blocks: 238 (93.0%)


Chunksize: 8192 bytes(2 blocks)
Total chunks: 129
Free chunk: 118 (91.5%)


Min. free extent: 12 KB
Max. free extent: 856 KB
Avg. free extent: 236 KB
Num. free extent: 4


HISTOGRAM OF FREE EXTENT SIZES:
Extent Size Range : Free extents  Free Blocks Percent
    8K ...  16K-  :            1            3   1.26%
   16K ...  32K-  :            1            6   2.52%
   32K ...  64K-  :            1           15   6.30%
  512K ...1024K-  :            1          214  89.92%
```

5.2　ed

ed は行指向のテキストエディタである。

対話的に、あるいはシェルスクリプトを使って、 テキストファイルの作成、表示、変更、その他の操作に使われる。

ed の制限付きバージョンである red は、カレントディレクトリのファイルのみを編集でき、 シェルコマンドを実行することはできない。

ed は、Unix のオリジナルエディタであり、広く利用できるという意味で、「標準的な」テキストエディタである。

しかし、ほとんどの用途では、GNU Emacs や GNU Moe のようなフルスクリーンエディタが使用されている。

実行例

```
user@Ubuntu:~$ ed
P
*i
Hello
.
*a
Good Morining
Good Night
.
*w test
30
*q
```

実行結果

```
user@Ubuntu:~$ cat test
Hello
Good Morning
Good Night
```

♣ オプション一覧

-E もしくは --extended-regexp

拡張正規表現を使う。

実行例

```
1,3p
1,3s/'o{2}'/o/1
1,3p
```

実行結果

```
Hello
Good Morning
Good Night

Hello
God Morining
Good Night
```

-G もしくは --traditional

従来型、互換モードで実行する。

実行例

```
ed -G test
```

実行結果

```
変化なし
```

-l もしくは --loose-exit-status

コマンドに失敗しても 0 ステータスで終了する。

実行例

```
ed -l test
```

実行結果

```
*aaaa
強制終了
```

-p もしくは --prompt=STRING

STRING を対話型プロンプトとして使う。

実行例

```
ed -p test
```

実行結果

```
test P
test i
Hello
.
test p
Hello
test w test
6
test Q
```

-r もしくは --restricted

制限モードで実行する。

実行例

```
ed -r
```

実行結果

```
変化なし
```

-s もしくは --quiet もしくは --silent

診断、バイト数、'!' プロンプトを表示しない。

実行例

```
ed -s test
```

実行結果

```
*w
*q
```

-v もしくは --verbose

H’ コマンドと同じ。

実行例

```
ed -v test
```

実行結果

```
変化なし
```

--strip-trailing-cr

テキスト行末のキャリッジ・リターンを除去する。

一部のテキストファイルで行末に挿入される\r を削除できる。

実行例

```
ed --strip-trailing-cr test
cat test
```

実行結果

```
Hello
Good Morning
Good Night
```

第 6 章
頭文字が f のコマンド

6.1 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
```

第 7 章
頭文字が g のコマンド

7.1 grub-mkrelpath

指定したパスを、システムが使用するルートディレクトリとする。 grub-mkconfig が使用する。

実行結果は入力したパスがそのまま表示される。

grub-mkrelpath → make a system path relative to its root

実行例

```
grub-mkrelpath PATH
```

実行結果

```
PATH
```

7.2 gs

gs → ghostscript PostScript および PDF ファイルを表示および印刷するためのオープンソースのプログラム

実行例

```
gs -q -sPAPERSIZE=a4 -dBATCH -dNOPAUSE -sDEVICE=pdfwrite -sOutputFile=test.pdf
```

実行結果

```
なし
```

単一のページからなる無地の PDF ファイル test.pdf が作成される。

実行例

```
gs -sDEVICE=pngalpha -o test.png -r300 -dNOPAUSE -q test.pdf -c quit
```

実行結果

```
なし
```

test.pdf を DPI300 で test.png に変換する。

♣ オプション一覧

-sDEVICE=\<device>

出力デバイスを指定する。

以下は device に指定できる項目である。

- PDF デバイス:

pdfwrite: PDF ファイルを生成する。

pdfmark: PDF マークファイルを生成する。

pdfdraw: PDF 描画オブジェクトを生成する。

- 画像デバイス:

pngalpha: PNG 形式で透明な背景を持つ画像を生成する。

jpeg: JPEG 形式の画像を生成する。

tiffg4: TIFF 形式の画像を生成する。

- プリンターデバイス:

ljet4: HP LaserJet 4 プリンターのエミュレーションを行う。

cups: CUPS (Common UNIX Printing System) 用のデバイス。

- 表示デバイス:

x11: X Window System 上に画像を表示する。

display: ディスプレイに表示する。

- SVG デバイス:

svg: SVG (Scalable Vector Graphics) ファイルを生成する。

- PostScript デバイス:

ps2write: Level2 の PostScript ファイルを生成する。

psmono: モノクロの PostScript ファイルを生成する。

-sOutputFile=\<file>

出力ファイルの名前を指定する。

-r\<res>

解像度を指定する。

実行例

```
-r300
```

300 DPI に設定

\<file>

入力ファイルの名前を指定する。

-o \<file>

出力ファイルの名前をしていする。

-dBATCH

PostScript コマンドを読み取る対話型ループに入るのではなく、コマンド ラインで指定されたすべてのファイルを処理した後に Ghostscript を終了するようになる。コマンドラインの最後に -c quit を入力するのと同じである。

-dNOPAUSE

プロンプトを無効にし、各ページの終わりで一時停止します。通常ファイルに出力を生成する場合は-dBATCH と共に使用する必要がある。

-q もしくは -dQUIET

標準出力上のルーチン情報コメントを抑制する。

-dFirstPage=\<n>

変換する最初のページを指定する。

-dLastPage=\<n>

変換する最後のページを指定する。

-sPAPERSIZE=\<size>

用紙サイズを指定する。デフォルトは a4。

以下は size に指定できる項目である。

- a0~a6、b0~b6、c0~c6
- letter、legal、ledger、tabloid、executive、comm10

-c \<command>

PostScript コマンドを実行する。

実行例

```
-c "showpage"
```

-dPDFSETTINGS=\<value>

PDF 変換の品質を調整する。

以下が value に指定できる項目である。

- /screen:

画面表示用の低品質設定。低い解像度で生成される。ファイルサイズは小さくなり、画質も低下する。

- /ebook:

電子書籍用の中程度の品質設定。モノクロ画像は高品質、カラー画像は中程度の品質で生成される。

- /printer:

プリンター用の高品質設定。高解像度で生成される。ファイルサイズは大きくなるが、印刷時に高品質な結果が得られる。

- /prepress:

印刷用の最高品質設定。非常に高い解像度で生成され、印刷物としての品質が求められる場合に適している。

- /default

デフォルト設定。通常、中程度の品質とファイルサイズになる。

g\<width>x\<height>

生成されるページの幅を指定する。入力する値はポイント（1/72 インチ）単位で指定する。

7.3 gsbj

Ghostscript を使用して BubbleJet プリンター向けにテキストをフォーマットおよび印刷する。ヘッダーやフッターの文字列内の % # は page #で置き換えられる。デフォルトのデバイス（-sDEVICE=）と解像度（-r）は次のとおりである。

gsbj bj10e 180

実行例

```
gsbj test.pdf
```

実行結果

```
なし
```

test.pdf を印刷する。

♣ オプション一覧

-12BclqRr もしくは -b\<header> もしくは -f\<font> もしくは -F\<hfont> もしくは -L\<lines> もしくは -p\<outfile>

基本のフォーマットを設定する。

-T\<n>

tab の幅を指定する。

--add-to-space \<units>

各スペースの幅に指定された 1/72 インチ単位の数を追加する（負の値も可能）。

--add-to-width \<units>

各文字の幅に指定された 1/72 インチ単位の数を追加する（負の値も可能）。

--columns \<n>

<n>列で印刷する。

--detect

ファイルが %!で始まる場合、PostScript ファイルとして扱う。

--first-page \<n>

ページ<n>から印刷を開始する。

--kern \<file.afm>

指定された.AFM ファイルの情報を使用してカーニングを行う。

--last-page \<n>

ページ<n>で印刷を停止する。

--(heading|footing)-(left|center|right) \<string>

ヘッダー/フッターのフィールドを設定する。

--margin-(top|bottom|left|right) <inches>

余白を設定する。

--no-eject-(file|formfeed)

end-of-file（EOF）が検出された場合、新しい列は始まるが、新しいページは始まらなくなる。

--spacing <n>

行間の幅を 2 倍（n=2）,3 倍（n=3）などで指定する。 #@#

7.4 gsdj

Ghostscript を使用して DeskJet プリンター向けにテキストをフォーマットおよび印刷する。ヘッダーやフッターの文字列内の % # は page #で置き換えられる。デフォルトのデバイス（-sDEVICE=）と解像度（-r）は次のとおりである。

gsdj deskjet 300

実行例

```
gsdj test.pdf
```

実行結果

```
なし
```

test.pdf を印刷する。

♣ オプション一覧

-12BclqRr もしくは -b\<header> もしくは -f\<font> もしくは -F\<hfont> もしく

は -L\<lines> もしくは -p\<outfile>

基本のフォーマットを設定する。

-T\<n>

tab の幅を指定する。

--add-to-space \<units>

各スペースの幅に指定された 1/72 インチ単位の数を追加する（負の値も可能）。

--add-to-width \<units>

各文字の幅に指定された 1/72 インチ単位の数を追加する（負の値も可能）。

--columns \<n>

<n>列で印刷する。

--detect

ファイルが %!で始まる場合、PostScript ファイルとして扱う。

--first-page \<n>

ページ<n>から印刷を開始する。

--kern \<file.afm>

指定された.AFM ファイルの情報を使用してカーニングを行う。

--last-page \<n>

ページ<n>で印刷を停止する。

--(heading|footing)-(left|center|right) \<string>

ヘッダー/フッターのフィールドを設定する。

--margin-(top|bottom|left|right) <inches>

余白を設定する。

--no-eject-(file|formfeed)

end-of-file（EOF）が検出された場合、新しい列は始まるが、新しいページは始まらなくなる。

--spacing <n>

行間の幅を 2 倍（n=2）,3 倍（n=3）などで指定する。 #@#

7.5 gsdj500

Ghostscript を使用して DeskJet 500 BubbleJet 向けにテキストをフォーマットおよび印刷する。ヘッダーやフッターの文字列内の % # は page #で置き換えられる。デフォルトのデバイス（-sDEVICE=）と解像度（-r）は次のとおりである。

gsdj500 djet500 300

実行例

```
gsdj500 test.pdf
```

実行結果

```
なし
```

test.pdf を印刷する。

♣ オプション一覧

-12BclqRr もしくは -b\<header> もしくは -f\<font> もしくは -F\<hfont> もしくは -L\<lines> もしくは -p\<outfile>

基本のフォーマットを設定する。

-T\<n>

tab の幅を指定する。

--add-to-space \<units>

各スペースの幅に指定された 1/72 インチ単位の数を追加する（負の値も可能）。

--add-to-width \<units>

各文字の幅に指定された 1/72 インチ単位の数を追加する（負の値も可能）。

--columns \<n>

<n>列で印刷する。

--detect

ファイルが %!で始まる場合、PostScript ファイルとして扱う。

--first-page \<n>

ページ<n>から印刷を開始する。

--kern \<file.afm>

指定された.AFM ファイルの情報を使用してカーニングを行う。

--last-page \<n>

ページ<n>で印刷を停止する。

--(heading|footing)-(left|center|right) \<string>

ヘッダー/フッターのフィールドを設定する。

--margin-(top|bottom|left|right) <inches>

余白を設定する。

--no-eject-(file|formfeed)

end-of-file（EOF）が検出された場合、新しい列は始まるが、新しいページは始まらなくなる。

--spacing <n>

行間の幅を 2 倍（n=2）,3 倍（n=3）などで指定する。 #@#

7.6 gslj

Ghostscript を使用して LaserJet プリンター向けにテキストをフォーマットおよび印刷する。ヘッダーやフッターの文字列内の % # は page #で置き換えられる。デフォルトのデバイス（-sDEVICE=）と解像度（-r）は次のとおりである。

gslj laserjet 300

実行例

```
gslj test.pdf
```

実行結果

```
なし
```

test.pdf を印刷する。

♣ オプション一覧

-12BclqRr もしくは -b\<header> もしくは -f\<font> もしくは -F\<hfont> もしく

は -L\<lines> もしくは -p\<outfile>

基本のフォーマットを設定する。

-T\<n>

tab の幅を指定する。

--add-to-space \<units>

各スペースの幅に指定された 1/72 インチ単位の数を追加する（負の値も可能）。

--add-to-width \<units>

各文字の幅に指定された 1/72 インチ単位の数を追加する（負の値も可能）。

--columns \<n>

<n>列で印刷する。

--detect

ファイルが %!で始まる場合、PostScript ファイルとして扱う。

--first-page \<n>

ページ<n>から印刷を開始する。

--kern \<file.afm>

指定された.AFM ファイルの情報を使用してカーニングを行う。

--last-page \<n>

ページ<n>で印刷を停止する。

--(heading|footing)-(left|center|right) \<string>

ヘッダー/フッターのフィールドを設定する。

--margin-(top|bottom|left|right) <inches>

余白を設定する。

--no-eject-(file|formfeed)

end-of-file（EOF）が検出された場合、新しい列は始まるが、新しいページは始まらなくなる。

--spacing <n>

行間の幅を 2 倍（n=2）,3 倍（n=3）などで指定する。 #@#

7.7 gslp

Ghostscript を使用してテキストをフォーマットおよび印刷する。ヘッダーやフッターの文字列内の % # は page #で置き換えられる。デフォルトのデバイス（-sDEVICE=）と解像度（-r）は次のとおりである。

gslp epson 180

実行例

```
gslp test.pdf
```

実行結果

```
なし
```

test.pdf を印刷する。

♣ オプション一覧

-12BclqRr もしくは -b\<header> もしくは -f\<font> もしくは -F\<hfont> もしくは -L\<lines> もしくは -p\<outfile>

基本のフォーマットを設定する。

-T\<n>

tab の幅を指定する。

--add-to-space \<units>

各スペースの幅に指定された 1/72 インチ単位の数を追加する（負の値も可能）。

--add-to-width \<units>

各文字の幅に指定された 1/72 インチ単位の数を追加する（負の値も可能）。

--columns \<n>

<n>列で印刷する。

--detect

ファイルが %!で始まる場合、PostScript ファイルとして扱う。

--first-page \<n>

ページ<n>から印刷を開始する。

--kern \<file.afm>

指定された.AFM ファイルの情報を使用してカーニングを行う。

--last-page \<n>

ページ<n>で印刷を停止する。

--(heading|footing)-(left|center|right) \<string>

ヘッダー/フッターのフィールドを設定する。

--margin-(top|bottom|left|right) <inches>

余白を設定する。

--no-eject-(file|formfeed)

end-of-file（EOF）が検出された場合、新しい列は始まるが、新しいページは始まらなくなる。

--spacing <n>

行間の幅を 2 倍（n=2）,3 倍（n=3）などで指定する。 #@#

7.8 gsnd

Ghostscript（PostScript および PDF エンジン）を表示なしで実行する。このコマンドは、gs を -NODISPLAY フラグ付きで呼び出している。

実行例、オプション一覧は gs コマンド参照

7.9 gst-device-monitor-1.0

gst → GStreamer GStreamer は、マルチメディアデータの処理やストリーミングを行うためのフレームワークであり、様々なメディア関連のタスクに使用される。gst-device-monitor-1.0 は、GStreamer のデバイス監視機能を利用するためのコマンド。

実行例

```
$ gst-device-monitor-1.0
```

実行結果

```
  Probing devices...

  Device found

      name  : Monitor of 内部オーディオ　アナログステレオ
      class : Audio/Source
      caps  : audio/x-raw, format={ (string)S16LE, (string)S16BE, (string)F32LE, (string)F›
›32BE, (string)S32LE, (string)S32BE, (string)S24LE, (string)S24BE, (string)S24_32LE, (strin›
›g)S124_32BE, (string)UB }, layout=interleaved, rate=[ 1, 384000 ], channels=[ 1, 32 ]

          audio/x-alaw, rate=[ 1, 384000 ], channels=[ 1, 32 ]
          audio/x-mulaw, rate=[ 1, 384000 ], channels=[ 1, 32 ]
      properties:
          device.description = "Monitor\ of\ \345\206\205\351\203\250\343\202\252\343\203\›
›274\343\203\207\343\202\243\343\202\252\ \343\202\242\343\203\212\343\203\255\343\202\260\›
›343\202\271\343\203\205\343\2-3\254\343\202\252"
          device/class = monitor
          alsa.card = 0
          alse.card_name = "Intel\ 82801AA-ICH"
  ~~~~以下省略~~~~
```

♣ オプション一覧

-f もしくは --follow

デバイス・リストを表示した後は終了せず、デバイスの追加と削除を待つ。

実行例

```
$ gst-device-monitor-1.0 -f
```

実行結果

```
Probing devices...

Monitoring devices, waiting for devices to be removed or new devices to be added

~~~~以下実行例と同様~~~~
```

実行後はデバイスリストを現在のデバイスリストを表示後、待機する。

-i もしくは --include-hidden

非表示のデバイスを含めて表示する。

実行例

```
$ gst-device-monitor-1.0 -i
```

実行結果

```
実行例と同様のため省略
```

7.10 gst-discoverer-1.0

gst → GStreamer GStreamer は、マルチメディアデータの処理やストリーミングを行うためのフレームワークであり、様々なメディア関連のタスクに使用される。指定したファイルや URI に関するメタデータやプロパティ、ストリームの構造などを取得する。

実行例

```
$ gst-discoverer-1.0 video.mp4
```

実行結果

```
Analyzing file:///home/user/video/video.mp4
Done discovering file:file:///home/user/video/video.mp4
Missing plugins
  (gstreamer|1.0|gst-discoverer-1.0|H.264 (Main Profile) デコーダー|decoder-video/x=h264, ›
›level=(string)3.1, profile=(string)main)
  (gstreamer|1.0|gst-discoverer-1.0|MPEG-4 AAC デコーダー|decoder-audio/mpeg, mpegversion=›
›(int)4, level=(string)2, base-profile=(string)lc, profile=(string)lc)
```

♣ オプション一覧

-v、--verbose

入手可能な情報を全て出力する。

実行例

```
$ gst-discoverer-1.0 -v video.mp4
```

実行結果

```
実行例と同様のため省略
```

-t もしくは --timeout=\<T>

タイムアウトを秒単位で指定する。 (デフォルトは 10 秒)
実行後指定した秒数でタイムアウトする。

実行例

```
$ gst-discoverer-1.0 -t 20 video.mp4
```

実行結果

```
実行例と同様のため省略
```

-c もしくは --toc

可能な場合は目次 (章やチャプター) を出力する。

実行例

```
$ gst-discoverer-1.0 -c video.mp4
```

実行結果

```
実行例と同様のため省略
```

-a もしくは --async

非同期コードパスを使用する。
プログラムが非同期的に動作するようになる。

実行例

```
$ gst-discoverer-1.0 -a video.mp4
```

実行結果

```
実行例と同様のため省略
```

7.11　gst-inspect-1.0

gst → GStreamer GStreamer は、マルチメディアデータの処理やストリーミングを行うためのフレームワークであり、様々なメディア関連のタスクに使用される。GStreamer のプラグインに関する情報を取得するためのコマンド。

実行例

```
$ gst-inspect-1.0
```

実行結果

```
1394: dv1394src: Firewire (1394) DV video source
1394: hdv1394src: Firewire (1394) HDV video source
aasink: aasink: ASCII art video sink
aasink: aatv: aaTV effect
```

```
adder: adder: Adder
alaw: alawdec: A Law audio decoder
alaw: alawenc: A Law audio encoder
alpha: alpha: Alpha filter
alphacolor: alphacolor: Alphacolor filter
~~~~以下省略~~~~
```

♣ オプション一覧

\<PLUGIN|ELEMENT>

指定したプラグインまたは要素の詳細を表示する。

実行例

```
$ gst-inspect-1.0 audiotestsrc
```

実行結果

```
Factory Details:
    Rank                    none (0)
    Long-name               Audio test source
    Klass                   Source/Audio
    Description             Creates audio test signals of given frequency and volume
    Author                  Stefan Kost <ensonic@users.sf.net>

~~~~以下省略~~~~
```

-a もしくは --print-all

全てのプラグインと要素を全て出力する。

実行例

```
$ gst-inspect-1.0 -a
```

実行結果

```
dv1394src: Factory Details:
dv1394src:  Rank              none (0)
dv1394src:  Long - name       Firewire (1394) DV video source
dv1394srv:  Klass             Source/Video
dv1394src:  Description       Source for DV video data from firewire port
~~~~以下省略~~~~
```

-b もしくは --print-blacklist

ブラックリストを表示する。

実行例

```
$ gst-inspect-1.0 -b
```

実行結果

```
Blacklisted files:

Total count: 0 blacklisted files
```

--plugin

プラグインを表示する。

実行例

```
$ gst-inspect-1.0 --plugin
```

実行結果

```
dv1394src: Factory Details:
dv1394src:  Rank              none (0)
dv1394src:  Long - name       Firewire (1394) DV video source
dv1394srv:  Klass             Source/Video
dv1394src:  Description       Source for DV video data from firewire port
~~~~以下省略~~~~
```

--print-plugin-auto-install-info

指定されたプラグインが提供する機能の機械が可読なリストを表示します。外部の自動プラグインインストールメカニズムと連携する際に役立つ。

実行例

```
$ gst-inspect-1.0 --print-plugin-auto-install-info
```

実行結果

```
element-dv1394src
urisource-dv
element-hdv1394src
urisource-hdv
element-aasink
element-aatv
element-adder
~~~~以下省略~~~~
```

-t もしくは --types

指定した文字列が含まれるプラグインや要素のみを表示する。

実行例

```
$ gst-inspect-1.0  -t bin
```

実行結果

```
camerabin:  camerabin: Camera Bin
cluttergst3:  clutterautovideosink: Generic bin
encoding:  encodebin: Encoder Bin
encoding:  encodebin2: Encoder Bin
multifile:  splitmuxsink: Split Muxing Bin
multifile:  splitmuxsrc: Split File Demuxing Bin
~~~~以下省略~~~~
```

-u もしくは --uri-handlers

サポートされている URI スキームとそれを実装する要素を表示する。

実行例

```
$ gst-inspect-1.0 -u
```

実行結果

```
dv1394src (read, rank 0): dv
hdv1394src (rad, rank 0): hdv
appsink (write, rank 0): appsink
appsrc (read, rank 0): appsrc
cdparanoasrc (read, rank 128): cdda
~~~~以下省略~~~~
```

--no-colors

結果を表示する際に単色で表示する。（デフォルトは要素ごとに色分け）

実行例

```
$ gst-inspect-1.0 --no-colors
```

実行結果

```
1394: dv1394src: Firewire (1394) DV video source
1394: hdv1394src: Firewire (1394) HDV video source
aasink: aasink: ASCII art video sink
aasink: aatv: aaTV effect
adder: adder: Adder
alaw: alawdec: A Law audio decoder
alaw: alawenc: A Law audio encoder
alpha: alpha: Alpha filter
alphacolor: alphacolor: Alphacolor filter
~~~~以下省略~~~~
```

-C もしくは --color

表示される要素ごとに色をつけて表示する。

実行例

```
$ gst-inspect-1.0 -C
```

実行結果

```
1394: dv1394src: Firewire (1394) DV video source
1394: hdv1394src: Firewire (1394) HDV video source
aasink: aasink: ASCII art video sink
aasink: aatv: aaTV effect
adder: adder: Adder
alaw: alawdec: A Law audio decoder
alaw: alawenc: A Law audio encoder
alpha: alpha: Alpha filter
alphacolor: alphacolor: Alphacolor filter
~~~~以下省略~~~~
```

7.12　gst-launch-1.0

gst → GStreamer GStreamer は、マルチメディアデータの処理やストリーミングを行うためのフレームワークであり、様々なメディア関連のタスクに使用される。gst-launch-1.0 は、コマンドラインから GStreamer パイプラインを簡単に構築および実行するためのコマンド。GStreamer では Shell のパイプ "|" ではなく "!" を用いる。

実行例

```
$ gst-launch-1.0 audiotestsrc ! autoaudiosink
```

実行結果

```
パイプラインを一時停止 (PAUSED) にしています...
Pipeline is PREROLLING ...
Pipeline is PREROLLED ...
パイプラインを再生中 (PLAYING) にしています...
New clock: GstPulseSinkClock
Redistribute latency...
```

テスト音源の 440Hz が出力される。

♣ オプション一覧

-t もしくは --tags

タグ (メタデータ) を出力する。

実行例

```
$ gst-launch-1.0 -t audiotestsrc ! autoaudiosink
```

実行結果

```
   パイプラインを一時停止 (PAUSED) にしています...
   Pipeline is PREROLLING ...
   FOUND TAG      : found by element "autoaudiosink0-actual-sink-pilse".
           詳細: audiotest wave
   Pipeline is PREROLLED ...
   パイプラインを再生中 (PLAYING) にしています...
   New clock: GstPulseSinkClock
   Redistribute latency...
```

-c もしくは --toc

可能な場合は目次 (章やチャプター) を出力する。

実行例

```
   $ gst-launch-1.0 -c audiotestsrc ! autoaudiosink
```

実行結果

```
   実行例と同様のため省略
```

-v もしくは --verbose

ステータス情報とプロパティ通知を出力する。

実行例

```
   $ gst-launch-1.0 -v audiotestsrc ! autoaudiosink
```

実行結果

```
   パイプラインを一時停止 (PAUSED) にしています...
   Pipeline is PREROLLING ...
   /GstPipeline:pipeline0/GstAudioTestSrc:audiotestsrc0.GstPad:src: caps = audio/x-raw, for›
›mat=(string)S16LE, layout=(string)interleaved, rate=(int)44100, channels=(int)1
   /GstPipeline:pipeline0/GstAutoAudioSink:autoaudiosink0.GstGhostPad:sink.GstProxyPad:prox›
›ypad0: caps = audio/x-raw, format=(string)S16LE, layout=(string)interleaved, rate=(int)441›
›00, channels=(int)1
   Redistribute latency...
   ~~~~以下省略~~~~
```

-q もしくは --quiet

出力データの詳細などは表示せず、再生時間のみを表示する。

実行例

```
   $ gst-launch-1.0 -q audiotestsrc ! autoaudiosink
```

実行結果

```
   0:00:00.0 / 99:99:99.
```

-m もしくは --messages

パイプラインのバスに提示される出力メッセージを表示する。

実行例

```
  $ gst-launch-1.0 -q audiotestsrc ! autoaudiosink
```

実行結果

```
  パイプラインを一時停止 (PAUSED) にしています...
  Pipeline is PREROLLING ...
  Got message #9 from element "autoaudiosink0-actual-sink-pulse" (state-changed): GstMessa›
›geStateChanged, old-state=(GstState)null, new-state=(GstState)ready, pending-state=(GstSta›
›te)void-pending;
  Got message #10 from element "autoaudiosink0" (state-changed): GstMessageStateChanged, o›
›ld-state=(GstState)null, new-state=(GstState)ready, pending-state=(GstState)void-pending;
  ~~~~以下省略~~~~
```

7.13　gst-play-1.0

gst → GStreamer GStreamer は、マルチメディアデータの処理やストリーミングを行うためのフレームワークであり、様々なメディア関連のタスクに使用される。gst-play-1.0 を使用すると、コマンドラインから簡単にメディアファイルを再生することができる。

実行例

```
  $ gst-play-1.0 music.mp3
```

実行結果

```
  Press 'k' to see a list of keyboard shortcuts.
  Now playing /home/user/music.mp3
  Redistribute latency...
  Redistribute latency...

  Interactive mode - keyboard controls:

    space    : pause/unpause
    q or ESC : quit
    > or n   : play next
    < or b   : play previous
    →         : seek forward
    ←         : seek backward
    ↑         : volume up
    ↓         : volume down
    m        : toggle audio mute on/off
    +        : increase playback rate
    -        : decrease playback rate
    d        : change playback direction
    t        : enable/disable trick modes
```

```
    a        : change audio track
    v        : change video track
    s        : change subtitle track
    0        : seek to beginning
    k        : show keyboard shortcuts
```

♣ オプション一覧

-v もしくは --verbose

ステータス情報とプロパティ通知を出力する。

実行例

```
$ gst-play-1.0 -v music.mp3
```

実行結果

```
Press 'k' to see a list of keyboard shortcuts.
Now playing /home/user/music.mp3
/GstPlayBin:playbin/GstURIDecodeBin:uridecodebin0: ring-buffer-max-size = 0
/GstPlayBin:playbin/GstURIDecodeBin:uridecodebin0: buffer-size = -1
/GstPlayBin:playbin/GstURIDecodeBin:uridecodebin0: buffer-duration = -1
/GstPlayBin:playbin/GstURIDecodeBin:uridecodebin0: force-sw-decoders = false
/GstPlayBin:playbin/GstURIDecodeBin:uridecodebin0: use-buffering = false
/GstPlayBin:playbin/GstURIDecodeBin:uridecodebin0: download = false
/GstPlayBin:playbin/GstURIDecodeBin:uridecodebin0: uri = file:///home/user/music.mp3
/GstPlayBin:playbin/GstURIDecodeBin:uridecodebin0: connection-speed = 0
~~~~以下省略~~~~
```

--audiosink=\<SOMESINK>

autoaudiosink の代わりに指定した SOMESINK をオーディオ出力として使用する。

- 具体的なオーディオ出力の例

autoaudiosink: システムのデフォルトのオーディオ出力

alsasink: ALSA (Advanced Linux Sound Architecture) を使用したオーディオ出力

pulsesink: PulseAudio を使用したオーディオ出力

osssink: OSS (Open Sound System) を使用したオーディオ出力

openslessink: OpenSL ES を使用したオーディオ出力

実行例

```
$ gst-play-1.0 --audiosink=alsasink music.mp3
```

実行結果

```
Press 'k' to see a list of keyboard shortcuts.
Now playing /home/user/music.mp3
Redistribute latency...
Redistribute latency...
```

--volume=\<VOLUME>

初期再生音量を指定する。

1.0 がデフォルト。

実行例

```
$ gst-play-1.0 --volume=0.7 music.mp3
```

実行結果

```
Volume: 70%
Press 'k' to see a list of keyboard shortcuts.
Now playing /home/user/music.mp3
Redistribute latency...
Redistribute latency...
```

--shuffle

プレイリストからランダムに並び替えて再生する。

実行例

```
$ gst-play-1.0 --shuffle music.mp3
```

実行結果

```
最初の実行例と同様のため省略
```

--no-interactive

ターミナルでのキーボード操作による制御を無効にする。

実行例

```
$ gst-play-1.0 --no-interactive music.mp3
```

実行結果

```
Now playing /home/user/music.mp3
Redistribute latency...
Redistribute latency...
```

--gapless

メディアファイル間の無音部分などを無くして、連続して再生する。

このオプションを指定することで、異なるメディアファイルを連続して再生する際に、中断や無音の時間を最小限に抑えることができる。

実行例

```
$ gst-play-1.0 --gapless music1.mp3 music2.mp3
```

実行結果

```
実行例と同様のため省略
```

1 つ目の再生が終了後 2 つ目の再生が始まる

7.14 gst-typefind-1.0

gst → GStreamer GStreamer は、マルチメディアデータの処理やストリーミングを行うためのフレームワークであり、様々なメディア関連のタスクに使用される。gst-typefind-1.0 を実行すると、ファイルのメディアタイプを表示することができる。

実行例

```
$ gst-typefind-1.0 music.mp3
```

実行結果

```
music.mp3 - application/x-id3
```

7.15 gunzip

ファイルの伸張 (解凍) を行うための UNIX および Linux コマンド。gunzip は gzip コマンドと同じプログラムの一部であり、通常、gzip で圧縮されたファイルを解凍するために用いられる。

実行例

```
$ gzip -v text.txt.gz
```

実行結果

```
text.txt.gz:     68.6% -- replaced with text.txt
```

♣ オプション一覧

-f, --force

ファイルが複数のリンクを持っている場合や、対応するファイルが既に存在する場合、または圧縮データが端末から読み取られたり端末に書き込まれたりしている場合であっても、強制的に圧縮または解凍を行う。

-f が指定されず、同じファイルが存在する場合、既存のファイルを上書きするかどうかを確認するためにプロンプトを表示する。

実行例

```
$ gzip -v -f text.txt
```

実行結果

```
-fを指定した場合
text.txt:     68.6% -- replaced with text.txt.gz
```

```
-fを指定せず、同じファイルが存在する場合
gzip: text.txt.gz already exists; do you wish to overwrite (y or n)? y
text.txt:     68.6% -- replaced with text.txt.gz
```

-k, --keep

入力したファイルを消去しない。

デフォルトは消去。

実行例

```
$ gzip -v -k text.txt
```

実行結果

```
text.txt:   68.6% -- created text.txt.gz
```

-l, --list

圧縮ファイルに対して実行することで以下の内容を表示する。

- compressed size: 圧縮ファイルのサイズ
- uncompressed size: 非圧縮ファイルのサイズ
- ratio: 圧縮率 (不明な場合は 0.0%)
- uncompressed_name: 非圧縮ファイルの名前

実行例

```
$ gzip -l text.txt.gz
```

実行結果

```
compressed        uncompressed  ratio  uncompressed_name
       242                 685  68.6%  text.txt
```

-n, --no-name

圧縮時には、元のファイル名とタイムスタンプを保存しない。

解凍時には、元のファイル名を復元しない（圧縮されたファイル名から gzip の接尾辞だけを削除する）。また元のタイムスタンプも復元しない（圧縮されたファイルからコピーしない）。このオプションは、解凍時にデフォルトで適用される。

実行例

```
$ gzip -v -n text.txt
```

実行結果

```
実行例と同様のため省略
```

-N, --name

圧縮時には、常に元のファイル名とタイムスタンプを保存する。

解凍時には、元のファイル名とタイムスタンプを復元する（それらが存在する場合）。このオプションは、ファイル名の長さに制限があるか、またはファイルの転送後にタイムスタンプが失われた場合に有用。

実行例

```
$ gzip -v -N text.txt
```

実行結果

```
実行例と同様のため省略
```

-q, --quiet

実行内容や警告文などを表示しない

実行例

```
$ gzip -q text.txt
```

実行結果

```
無し
```

-r, --recursive
指定したディレクトリや複数のファイルを再帰的に処理する。

```
.
├──text
│    ├──text0.txt
│    ├──text1.txt
```

上記のディレクトリ構造に対して実行する

実行例

```
$ gzip -r -v text
```

実行結果

```
text/text0.txt:     3.3% -- replaced with text/text0.txt.gz
text/text1.txt:     5.0% -- replaced with text/text1.txt.gz
```

-v, --verbose
処理内容を出力する。

実行例

```
$ gzip -v text.txt
```

実行結果

```
text.txt:     68.6% -- replaced with text.txt.gz
```

7.16　gzip

ファイルの圧縮および伸張を行うための UNIX および Linux コマンド。gzip は、データの圧縮と解凍に広く使用されており、通常は.gz という拡張子がついたファイルに適用される。

実行例

```
$ gzip -v text.txt
```

実行結果

```
text.txt:     68.6% -- replaced with text.txt.gz
```

♣ オプション一覧

-d もしくは --decompress

指定したファイルを解凍する。
圧縮ファイルにのみ実行可能。

実行例

```
$ gzip -v -d text.txt.gz
```

実行結果

```
text.txt.gz:     68.6% -- replaced with text.txt
```

-f もしくは --force

ファイルが複数のリンクを持っている場合や、対応するファイルが既に存在する場合、または圧縮データが端末から読み取られたり端末に書き込まれたりしている場合であっても、強制的に圧縮または解凍を行う。
-f が指定されず、同じファイルが存在する場合、既存のファイルを上書きするかどうかを確認するためにプロンプトを表示する。

実行例

```
$ gzip -v -f text.txt
```

実行結果

```
-fを指定した場合
text.txt:     68.6% -- replaced with text.txt.gz
```

```
-fを指定せず、同じファイルが存在する場合
gzip: text.txt.gz already exists; do you wish to overwrite (y or n)? y
text.txt:     68.6% -- replaced with text.txt.gz
```

-k もしくは --keep

入力したファイルを消去しない。

デフォルトは消去。

実行例

```
$ gzip -v -k text.txt
```

実行結果

```
text.txt:   68.6% -- created text.txt.gz
```

-l もしくは --list

圧縮ファイルに対して実行することで以下の内容を表示する。

- compressed size: 圧縮ファイルのサイズ
- uncompressed size: 非圧縮ファイルのサイズ
- ratio: 圧縮率 (不明な場合は 0.0%)
- uncompressed_name: 非圧縮ファイルの名前

実行例

```
$ gzip -l text.txt.gz
```

実行結果

```
compressed        uncompressed  ratio  uncompressed_name
       242                 685  68.6%  text.txt
```

-n もしくは --no-name

圧縮時には、元のファイル名とタイムスタンプを保存しない。

解凍時には、元のファイル名を復元しない（圧縮されたファイル名から gzip の接尾辞だけを削除する）。また元のタイムスタンプも復元しない（圧縮されたファイルからコピーしない）。このオプションは、解凍時にデフォルトで適用される。

実行例

```
$ gzip -v -n text.txt
```

実行結果

```
実行例と同様のため省略
```

-N もしくは --name

圧縮時には、常に元のファイル名とタイムスタンプを保存する。

解凍時には、元のファイル名とタイムスタンプを復元する（それらが存在する場合）。このオプションは、ファイル名の長さに制限があるか、またはファイルの転送後にタイムスタンプが失われた場合に有用。

実行例

```
$ gzip -v -N text.txt
```

実行結果

```
実行例と同様のため省略
```

-q もしくは --quiet

実行内容や警告文などを表示しない

実行例

```
$ gzip -q text.txt
```

実行結果

```
無し
```

-r もしくは --recursive

指定したディレクトリや複数のファイルを再帰的に処理する。

```
.
├──text
│    ├──text0.txt
│    ├──text1.txt
```

上記のディレクトリ構造に対して実行する

実行例

```
$ gzip -r -v text
```

実行結果

```
text/text0.txt:     3.3% -- replaced with text/text0.txt.gz
text/text1.txt:     5.0% -- replaced with text/text1.txt.gz
```

-v もしくは --verbose

処理内容を出力する。

実行例

```
$ gzip -v text.txt
```

実行結果

```
text.txt:     68.6% -- replaced with text.txt.gz
```

-\<#> もしくは --fast もしくは --best

指定された数字 # を使用して圧縮の速度を調整する。

1 または--fast は最速の圧縮メソッド（圧縮が少ない）を示し、-9 または--best は最遅の圧縮メソッド（最大の圧縮）を示す。デフォルトの圧縮レベルは-6。

実行例

```
$ gzip -v -9 text.txt
```

実行結果

```
実行例と同様のため省略
```

第 8 章
頭文字が l のコマンド

8.1 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
```

-b もしくは --escape

C 言語の形式で非表示文字（エスケープシーケンス）を表示する。

実行例

```
ls -b
```

実行結果

```
// 比較のために改行をわかりやすくしている
auto_create_md.py
ls.md
man.md
one                         // ここが改行されている
two.txt
ping.md
```

実行結果

```
auto_create_md.py  ls.md  man.md  one\ntwo.txt  ping.md
```

--block-size=SIZE

SIZE の部分を変更することで -l オプションを使用したときに確認できる、ファイルのバイトサイズを任意のオーダーで確認できるようになる。

引数として利用できるのは SI 接頭辞と整数の掛け合わせ (例：10K = 10 * 1024) の形式に限られる。

- ここで使える SI 接頭辞は **キロ (K/k)、メガ (M/m)、ギガ (G/g)、テラ (T/t)、ペタ (P/p)、エクサ (E/e)、ゼタ (Z/z)、ヨタ (Y/y)** であり、それぞれそのまま使用すると、2 の 10 乗である 1024 の倍数として利用できる (例：G = 1024 * 1024 * 1024)。

- 直後に B をつけることで、10 の累乗による概念の SI 接頭辞として利用できる (例：G = 10^9)。

- バイナリ接頭辞も利用することができ、 キロ に相当するのが KiB(キビバイト), メガ に相当するのが MiB(メビバイト) などのように使用できる。

実行例

```
ls -l --block-size=K
```

実行結果

```
total 36
-rw-r--r-- 1 user user  4519 Dec 25 15:10   auto_create_md.py
-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
```

実行結果

```
total 36K
-rw-r--r-- 1 user user  5K Dec 25 15:10 auto_create_md.py
-rw-rw-r-- 1 user user 12K Dec 25 15:10 ls.md
-rw-rw-r-- 1 user user 11K Dec 11 03:49 man.md
-rw-rw-r-- 1 user user  1K Dec 11 04:23 ping.md
```

上記のように指定したオーダー表記にするために切り上げ処理が施される。

-B もしくは --ignore-backups

「~」で終わるファイル (バックアップファイル) を表示しない。

実行例

```
ls -B
```

実行結果

```
auto_create_md.py  hoge.txt  hoge.txt~  ls.md  man.md  ping.md
```

実行結果

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

-c

-lt オプションと併用すると、 **ctime** (ファイルステータス情報の最終変更時刻) に基づいてファイルを最新のものからソートして表示する。

-l オプションと併用すると、**ctime** を含むファイルの詳細情報をファイル名ソートして表示する。

単独で使用すると、**ctime** でソートして表示する。

実行例

```
ls -ltc
ls -lc
ls -c
```

実行結果

```
// ファイルの最終変更時刻順
total 40
-rw-rw-r-- 1 user user     9 Dec 26 01:05 hoge.txt
-rw-rw-r-- 1 user user 11692 Dec 25 15:10 ls.md
-rw-r--r-- 1 user user  4519 Dec 25 15:10 auto_create_md.py
-rw-rw-r-- 1 user user  1024 Dec 11 23:42 ping.md
-rw-rw-r-- 1 user user 10558 Dec 11 23:42 man.md
```

実行結果

```
// ファイルの名前順
total 40
-rw-r--r-- 1 user user  4519 Dec 25 15:10 auto_create_md.py
-rw-rw-r-- 1 user user     9 Dec 26 01:05 hoge.txt
-rw-rw-r-- 1 user user 11692 Dec 25 15:10 ls.md
-rw-rw-r-- 1 user user 10558 Dec 11 23:42 man.md
-rw-rw-r-- 1 user user  1024 Dec 11 23:42 ping.md
```

実行結果

```
// ファイルの最終変更時刻順
hoge.txt  ls.md  auto_create_md.py  ping.md  man.md
```

-C

出力を列にするオプション。

実行例

```
ls -C
```

実行結果

```
なんか全く変わらないんで,消す候補です
```

-d もしくは --directory

ディレクトリの内容ではなく、ディレクトリそのものをリストアップする

実行例

```
ls -d ~/directory
```

実行結果

```
/home/user/directory
```

-D もしくは --dired

Emacs の dired モード用にデザインされた出力を生成する。

実行例

```
実行例
```

実行結果

```
実行結果
```

-F もしくは --classify

エントリに対して特定のマーク（インディケータ）を追加します。各マークはエントリの種類を示し、ファイルがディレクトリであるか、実行可能であるか、シンボリックリンクであるかなどを示します。

以下は、使用可能なマークとその意味です：

- /: ディレクトリ
- *: 実行可能なファイル
- =: ソケットファイル
- \>: ファイルがスパーシャルブロックデバイス\
- |: ファイルがパイプ（FIFO）ファイル
- @: シンボリックリンク

実行例

```
実行例
```

実行結果

```
実行結果
```

--file-type

-f と同様にインジケータをつけるものの、ファイルを示す'*' だけ付けないオプション。

実行例

```
実行例
```

実行結果

```
実行結果
```

--full-time

-l オプションと同様に詳細な情報を表示します。また、--time-style=full-iso オプションも適用され、時刻の表示が完全な ISO 8601 形式で行われます。

実行例

```
ls --full-time
```

実行結果

```
total 40
-rw-r--r-- 1 user user  4519 2023-12-25 15:10:17.029008035 +0900 auto_create_md.py
-rwxrwxr-x 1 user user    63 2023-12-26 01:27:26.133960460 +0900 create_hoge_files.sh
-rw-rw-r-- 1 user user 11692 2023-12-25 15:10:20.957091058 +0900 ls.md
-rw-rw-r-- 1 user user 10558 2023-12-11 03:49:33.924170811 +0900 man.md
-rw-rw-r-- 1 user user  1024 2023-12-11 04:23:17.953652140 +0900 ping.md
```

-g

-l オプションと同様に詳細情報を表示するが、ファイルの所有者の情報のみ表示しないで出力するオプション。

実行例

```
ls -g
```

実行結果

```
total 40
-rw-r--r-- 1 user  4519 Dec 25 15:10 auto_create_md.py
-rwxrwxr-x 1 user    63 Dec 26 01:27 create_hoge_files.sh
-rw-rw-r-- 1 user 11692 Dec 25 15:10 ls.md
-rw-rw-r-- 1 user 10558 Dec 11 03:49 man.md
-rw-rw-r-- 1 user  1024 Dec 11 04:23 ping.md
```

-G もしくは --no-group

長いリストでは、グループ名を印刷しない

実行例

```
実行例
```

実行結果

```
実行結果
```

-h もしくは --human-readable

-l オプションや -s オプションと組み合わせて使用されます。このオプションを使用すると、ファイルやディレクトリのサイズが人間が理解しやすい形式で表示されます。サイズが大きい場合にはキロバイト（K）、メガバイト（M）、ギガバイト（G）などの単位が適切に変換されて表示されます。

実行例

```
ls -lh
```

実行結果

```
total 44K
-rw-r--r-- 1 user user 4.5K 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 4.0K Dec 28 00:20 dir
-rw-rw-r-- 1 user user  12K Dec 25 15:10 ls.md
-rw-rw-r-- 1 user user  11K Dec 11 03:49 man.md
-rw-rw-r-- 1 user user 1.0K Dec 11 04:23 ping.md
```

--si

-l オプションや -s オプションと組み合わせて使用されます。このオプションを使用すると、ファイルやディレクトリのサイズが 2 のべき乗ではなく、10 のべき乗で表現され、キロ（K）、メガ（M）、ギガ（G）などの単位が使われます。

実行例

```
ls -l --si
```

実行結果

```
total 46k
-rw-r--r-- 1 shuuto shuuto 4.6k Dec 25 15:10 auto_create_md.py
-rwxrwxr-x 1 shuuto shuuto   63 Dec 26 01:27 create_hoge_files.sh
drwxrwxr-x 2 shuuto shuuto 4.1k Dec 28 00:20 dir
-rw-rw-r-- 1 shuuto shuuto  12k Dec 25 15:10 ls.md
-rw-rw-r-- 1 shuuto shuuto  11k Dec 11 03:49 man.md
-rw-rw-r-- 1 shuuto shuuto 1.1k Dec 11 04:23 ping.md
```

-H もしくは --dereference-command-line

コマンドラインに記載されているシンボリックリンクをたどる

実行例

```
実行例
```

実行結果

```
実行結果
```

--dereference-command-line-symlink-to-dir

コマンドラインのシンボリックリンクディレクトリを指す

実行例

```
実行例
```

実行結果

```
実行結果
```

--hyperlink[=WHEN]

ファイル名をハイパーリンクとして表示します。=WHEN はオプションで、次のいずれかの値を取ります：

- always: 常にハイパーリンクとして表示する（デフォルト値）。

- auto: 標準出力が端末に接続されている場合のみ、ハイパーリンクとして表示する。

- never: ハイパーリンクとして表示しない。

実行例

```
実行例
```

実行結果

```
実行結果
```

--indicator-style=WORD

エントリ名にスタイル付きのインディケータを追加します。WORD には以下のいずれかの値を指定できます：

- none: インディケータなし（デフォルト値）。

- -p: ディレクトリにはスラッシュ (/)、実行可能なファイルにはアスタリスク (*) など、エントリの種類に合わせたインディケータが表示されます。

- --file-type: --classify オプションと同様に、エントリの種類に応じたインディケータが表示されます。

- -F: --classify オプションと同じく、エントリの種類に応じたインディケータが表示されます。

実行例

```
実行例
```

実行結果

```
実行結果
```

-i もしくは --inode

各ファイルのインデックス番号（inode 番号）を表示するオプション。インデックス番号とは、ファイルやディレクトリがファイルシステム内で一意に識別されるための番号のこと。

実行例

```
ls -i
```

実行結果

```
1973241 auto_create_md.py     1969394 dir    1973352 man.md
1994807 create_hoge_files.sh  1994891 ls.md  1973372 ping.md
```

-I もしくは --ignore=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もマッチしたものが表示されない。

実行例

```
ls -aI '*.md'
```

実行結果

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

-k もしくは --kibibytes

-s オプションとの併用でのみ使用でき、ディスク使用量を 1024 バイト (1 キビバイト) 単位で表示するオプション。

実行例

```
ls -sk
```

実行結果

```
total 44
8 auto_create_md.py      4 dir    12 man.md
4 create_hoge_files.sh  12 ls.md   4 ping.md
```

-L もしくは --dereference

シンボリックリンクのファイル情報を表示する場合は、リンクの情報を表示します。

実行例

```
実行例
```

実行結果

```
実行結果
```

-n もしくは --numeric-uid-gid

-l オプションのように詳細情報を表示するが、所有者を数値のユーザー ID で、所有グループをグループ ID で表示する。

実行例

```
ls -n
```

実行結果

```
total 44
-rw-r--r-- 1 1000 1000  4519 Dec 25 15:10 auto_create_md.py
-rwxrwxr-x 1 1000 1000    63 Dec 26 01:27 create_hoge_files.sh
drwxrwxr-x 2 1000 1000  4096 Dec 28 00:20 dir
-rw-rw-r-- 1 1000 1000 11692 Dec 25 15:10 ls.md
-rw-rw-r-- 1 1000 1000 10558 Dec 11 03:49 man.md
-rw-rw-r-- 1 1000 1000  1024 Dec 11 04:23 ping.md
```

-N もしくは --literal

特殊文字やスペースがファイル名に存在している時でもクォーテーションを使用しないで表示するオプション。

実行例

```
ls -N
```

実行結果

```
// ' one'$'\n''two.txt' はファイル名に改行が入っている
 auto_create_md.py      dir     man.md                ping.md
 create_hoge_files.sh   ls.md  ' one'$'\n''two.txt'
```

実行結果

```
// one?two.txt という表示になった
auto_create_md.py     dir    man.md        ping.md
create_hoge_files.sh  ls.md   one?two.txt
```

-o

-l オプションと同様に詳細なリスト形式でエントリを表示しますが、グループ情報を表示しません。通常、-l オプションを使用すると、エントリごとに所有者とグループが表示されますが、-o オプションを追加することでグループ情報を省略できます。

実行例

```
ls -o
```

実行結果

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

-q もしくは --hide-control-chars

制御文字などの特殊文字が表示される時に ? で置換されて表示されるオプション。

実行例

```
実行例
```

実行結果

実行結果

--show-control-chars

制御文字などの特殊文字をそのまま表示するオプション。標準出力がターミナルに接続されている時は ls コマンドは特殊文字を表示するようになっているため、 ssh 接続時などで利用することが多い。

実行例

実行例

実行結果

実行結果

--quoting-style=WORD

ファイル (ディレクトリ) 名の引用スタイルを指定する。引数 (=WORD) には以下の単語を指定できる。

- literal： ファイル (ディレクトリ) 名をそのまま表示する。
- locale： ロケールに基づいたスタイルで表示する。
- shell： シェルの解釈可能なスタイルで表示する。
- shell-always： シェルの解釈可能なスタイルかつ、特殊文字が含まれていなくても引用符で囲まれて表示する。
- shell-escape： シェルの解釈可能なスタイルで、特殊文字をエスケープして表示する。
- shell-escape-always： シェルの解釈可能なスタイルで、特殊文字をエスケープして、特殊文字が含まれていなくても引用符で囲まれて表示する。
- c： プログラムで利用可能なスタイルで表示する。
- escape： 特殊文字をエスケープして表示する。

実行例

実行例

実行結果

実行結果

-s もしくは --size

各ファイルの割り当てられたサイズをブロック単位で表示するオプション。(ファ

イルサイズは通常はバイト単位で表示される。)

実行例

```
ls -s
```

実行結果

```
total 44
8  auto_create_md.py      4  dir    12  man.md               4  ping.md
4  create_hoge_files.sh  12  ls.md   0 'one'$'\n''two.txt'
```

--time=WORD

ファイルのタイムスタンプに関する表示やソートの方法を引数 (=WORD) にて指定し出力するオプション。WORD には以下の単語を指定できる。

- atime, access, use : ファイルのアクセス時刻 (atime) を表示またはソートする。
- ctime, status : ファイルの変更時刻 (ctime) を表示またはソートする。
- birth, creation : ファイルの作成時刻 (birth time) を表示またはソートする。

実行例

```
実行例
```

実行結果

```
実行結果
```

--time-style=TIME_STYLE

-l オプションと併用することで、表示されるファイルの最終変更時刻のスタイルを変更できるオプション。引数 (TIME_STYLE) には次のスタイルが適用できる。

- full-iso : 完全な ISO 8601

例) '2023-12-25 10:00:00.000000000 +0000'

- long-iso : 長い ISO 8601

例) '2023-12-25 10:00'

- iso : ISO 8601

例) '2023-12-25'

- locale : ロケールに基づいた日時のスタイル
- +FORMAT : date(1) コマンドのスタイルで指定したカスタムフォーマット

例) '+%Y-%m-%d %H:%M:%S'

TIME_STYLE 環境変数を設定することでデフォルトのスタイルを指定することもできる。

実行例

```
実行例
```

実行結果

```
実行結果
```

-t

ファイル (ディレクトリ) 名を時間が新しい順でソートして表示する。
ls --sort=time と全く同じ。

実行例

```
実行例
```

実行結果

```
実行結果
```

-T もしくは --tabsize=COLS

タブ文字のサイズを変更できるオプション。

実行例

```
実行例
```

実行結果

```
実行結果
```

-Z もしくは --context

各ファイルのセキュリティコンテキストを表示するオプション。

実行例

```
ls -Z
```

実行結果

```
実行結果
```

第 9 章
頭文字が m のコマンド

9.1 man-recode

マニュアルページ（ man ページ ）を別のエンコーディングに変換するためのユーティリティ。ファイル名から各ページに対して適切な入力エンコーディングを推測して変換することもある。これは、レガシーな文字コードが使用されているマニュアルページの多くを共通のエンコード方法（ 一般に UTF-8 ）で変換する場合で有用である。このプログラムは各ページに対して man --recode や manconv を実行するよりもはるかに高速である。

マニュアルページの最初の行で以下のようなエンコーディング宣言が見つかると、その宣言が入力エンコーディングとして使われる。そうでない場合は、入力エンコーディングはファイル名から推測される。

マニュアルページの先頭に書かれるエンコーディング宣言は通常は以下の形式をとる：

```
'\"-*- coding: UTF-8 -*-
```

**マニュアルページプリプロセッサ* も宣言する場合：

```
'\" t -*- coding：ISO-8859-1 -*-
```

(* マニュアルページプリプロセッサ：マニュアルページ（man ページ）の作成や処理を補助するためのツールやプログラムのこと。ソースコードを構造化された形式に変換し、適切なマークアップやフォーマットを適用する役割を果たす。)

上記のエンコーディング宣言を変換したいマニュアルページの冒頭に書き加えたの

ち、man-recode コマンドを実行することでエンコーディング方法を変更してマニュアルページを作成できる。

実行例

```
man-recode -t shift_jis --in-place touch.1.gz
```

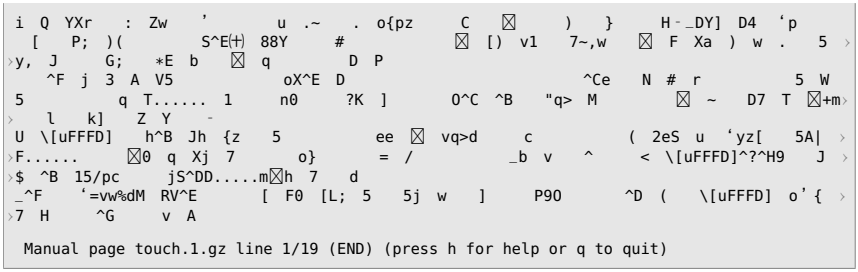
実行結果

```
// 変更前 touch.1.gz(UTF-8)
// user@localhost:/usr/share/man/man1$ man ./touch.1.gz

TOUCH(1)                          User Commands                          TOUCH(1)

NAME
       touch - change file timestamps

SYNOPSIS
       touch [OPTION]... FILE...

DESCRIPTION
       Update  the  access  and modification times of each FILE to the current
       time.

       A FILE argument that does not exist is created empty, unless -c  or  -h
       is supplied.

       A  FILE  argument  string of - is handled specially and causes touch to
       change the times of the file associated with standard output.

       Mandatory arguments to long options are  mandatory  for  short  options
       too.

       -a     change only the access time

       -c, --no-create
              do not create any files
...
```

以上のように正常なマニュアルページが表示される

実行結果

```
// 変更前 touch.1.gz(shift-jis)
// user@localhost:~/man-recode$ man ./touch.1.gz

^B^C  Vmo  6    _q    $N  u  Z      mb     "    (    D  "eG    ;  /
tb   ’        1
    "|  Gx^Fat       :  \[uFFFD]UU       y       ^C□    S    w  %          h’
\[uFFFD]  dp?                                ^L    gB  %  *    T!  Ei      q2  -□
|?xZ    ew:         l  O   0^DR",B     kG      ^F!&?H  .....  $  )  E    q  J:a    ^L    ,x(    L›
›    ci -
j%P  D  *|    ^D      ^Dy    X#    Jj  7B67    E              ^G  A}m  e(q%    (]□Z    y  ~\[uF›
›FFD]□    2<^Dx    e  +      u    k  +       I    m)c%    bQ[  ÷<U+0099>  m^E  □  a  A]  q  \[u›
›FFFD].□  0  +k     ’  3    ](drC  p         b^?/    lf*      !    ^B  WP  m‘Fc  G}
7  L]_    6(  ~b    ^
      A79  M    {    "     ’7^L         %@,(l    N      u       O  Z#;  7^Gy    ^Da  ^?:  E      y›
›    X  \[u0150]+      %      J^Bn     F      ^B  6x    |1R       @:  F      Q  ŏ z  B  W  i’    5›
›S^C         E  O      hp?cOOWW□         a    Sh    □  <    2  }TG4 ‘    H3    ^Gt      .  ^EA  .›
›  ^C    V         ^C~wj\[uFFFD]    l  G
\[uFFFD]                                                    i
3     - s    z                bCH    Z.  ^C  V  ^G^Hv4       N8  xO^G  t(  y  ^CL    V_T          ›
›
  d       3    W)K  ^D  I  +Qf  N  ~  PFK      %  Pr  .    ‘5  C0           }  4O
```



以上のように文字化けが発生してしまう。

♣ オプション一覧

-t もしくは --to-code=CODE

出力のエンコーディング方法を指定するオプション。この指定は必須であるため、忘れるとコマンドが実行できない。

上で既に実行例を出しているため実行例は省略する。

-d もしくは --debug

デバッグ情報を出すオプション。

実行例

```
man-recode -d -t shift_jis --in-place touch.1.gz
```

実行結果

```
loading seccomp filter (permissive: 0)
guessed input encoding UTF-8 for touch.1.gz
loading seccomp filter (permissive: 0)
seccomp already enabled
trying encoding UTF-8 -> shift_jis\/\/IGNORE
stem: touch.1, basename: touch.1.gz
```

この場合は、UTF-8 でエンコーディングされることが予想されるファイルであり、そのファイルを無理やり shift_jis でエンコーディングしようとしたことが読み取れる。

\/\/IGNORE は変換できない文字をそのままにしておくオプションであるらしい。

--in-place

入力ファイルをその場で上書きするオプション。圧縮拡張子が削除される。

上で既に実行例を出しているため実行例は省略する。

--suffix=SUFFIX

出力ファイル名に付加する接尾辞を引数（ SUFFIX ）として指定することで、その接尾辞でファイルを作成するオプション。

実行例

```
man-recode -t shift_jis --suffix=".txt" touch.1.gz
```

実行結果

```
// 実行前
// touchコマンドのマニュアルページのみが存在することを確認
user@localhost:~/man-recode$ ls
touch.1.gz

// 実行後
// touch.1.txt が存在することを確認
user@localhost:~/man-recode$ ls
touch.1.gz  touch.1.txt
```

-q もしくは --quiet

ページを変換できない場合にエラーメッセージを表示しないようにするオプション。

実行例

```
実行例
```

実行結果

```
実行結果
```

9.2 manpath

マニュアルページの検索パス（ manpath ）を決定、表示するための **ユーティリティ* 。

(* ユーティリティ：コンピュータの分析、構成、最適化、保守のためのソフトウェアの総称。ユーティリティソフトウェアとも呼ばれる。)

環境変数 $MANPATH が設定されている場合、manpath はその内容を表示して警告を発する。もし設定されていなければ、 manpath は適切なマニュアルページの階層検索パスを決定し、その結果を表示する。

パスはコロンで区切られ、/etc/manpath.config（ man-db の設定ファイル ）およびユーザーの環境から取得された情報を使用して決定される。

実行例

```
manpath
```

実行結果

```
/usr/local/man:/usr/local/share/man:/usr/share/man/cat1:/usr/share/man
```

♣ オプション一覧

-c もしくは --catpath

manpath が決定された後、各パス要素を相対的な catpath に変換するオプション。

実行例

```
manpath -c
```

実行結果

```
/var/cache/man/oldlocal:/var/cache/man/local:/usr/share/man/cat1:/var/cache/man
```

余談だが、/etc/manpath.config という設定ファイルの中に次のような記述がある。

```
#         *MANPATH*      ->       *CATPATH*
#
MANDB_MAP     /usr/man                        /var/cache/man/fsstnd
MANDB_MAP     /usr/share/man              /var/cache/man
MANDB_MAP     /usr/local/man              /var/cache/man/oldlocal
MANDB_MAP     /usr/local/share/man    /var/cache/man/local
MANDB_MAP     /usr/X11R6/man              /var/cache/man/X11R6
MANDB_MAP     /opt/man                        /var/cache/man/opt
MANDB_MAP     /snap/man                       /var/cache/man/snap
```

ここのコンフィグファイルに書かれているマッピングをもとに変換されることが最初に例示した実行例と上記の実行例からわかる。

-C FILE もしくは --config-file=FILE

このユーザー設定ファイルを使用し、デフォルトの ~/.manpath ではない manpath を決定するオプション。

実行例

```
manpath -C hoge.conf
```

実行結果

```
// hoge.conf というコンフィグファイルを用意
user@localhost:~/manpath$ ls
hoge.conf

// 内容は次のとおり
MANDATORY_MANPATH  /etc

// 実行結果
/usr/local/man:/usr/local/share/man:/usr/share/man/cat1:/usr/share/man:/etc
```

/etc ディレクトリが追加されていることがわかる。

-d もしくは --debug

デバッグ情報を表示するオプション。

実行例

```
manpath -d
```

実行結果

```
From the config file /etc/manpath.config:
Mandatory mandir `/usr/man'.
Mandatory mandir `/usr/share/man'.
Mandatory mandir `/usr/local/share/man'.
Path `/bin' mapped to mandir `/usr/share/man'.
Path `/usr/bin' mapped to mandir `/usr/share/man'.
Path `/sbin' mapped to mandir `/usr/share/man'.
Path `/usr/sbin' mapped to mandir `/usr/share/man'.
Path `/usr/local/bin' mapped to mandir `/usr/local/man'.
Path `/usr/local/bin' mapped to mandir `/usr/local/share/man'.
Path `/usr/local/bin' mapped to mandir `/usr/share/man/cat1'.
Path `/usr/local/sbin' mapped to mandir `/usr/local/man'.
Path `/usr/local/sbin' mapped to mandir `/usr/local/share/man'.
Path `/usr/X11R6/bin' mapped to mandir `/usr/X11R6/man'.
Path `/usr/bin/X11' mapped to mandir `/usr/X11R6/man'.
Path `/usr/games' mapped to mandir `/usr/share/man'.
Path `/opt/bin' mapped to mandir `/opt/man'.
Path `/opt/sbin' mapped to mandir `/opt/man'.
Global mandir `/usr/man', catdir `/var/cache/man/fsstnd'.
Global mandir `/usr/share/man', catdir `/var/cache/man'.
Global mandir `/usr/local/man', catdir `/var/cache/man/oldlocal'.
Global mandir `/usr/local/share/man', catdir `/var/cache/man/local'.
Global mandir `/usr/X11R6/man', catdir `/var/cache/man/X11R6'.
Global mandir `/opt/man', catdir `/var/cache/man/opt'.
Global mandir `/snap/man', catdir `/var/cache/man/snap'.
Added sections: `1', `n', `l', `8', `3', `0', `2', `3posix', `3pm', `3perl', `3am', `5',
› `4', `9', `6', `7'.
path directory /home/shuuto/.local/bin is not in the config file
path directory /usr/local/sbin is in the config file
  adding /usr/local/man to manpath
  adding /usr/local/share/man to manpath
path directory /usr/local/bin is in the config file
  adding /usr/share/man/cat1 to manpath
path directory /usr/sbin is in the config file
  adding /usr/share/man to manpath
path directory /usr/bin is in the config file
path directory /sbin is in the config file
path directory /bin is in the config file
path directory /usr/games is in the config file
path directory /usr/local/games is not in the config file
path directory /snap/bin is not in the config file
adding mandatory man directories
warning: /usr/man: No such file or directory
/usr/local/man:/usr/local/share/man:/usr/share/man/cat1:/usr/share/man
```

-g もしくは --global

man-db 設定ファイルで「global」として指定されたすべてのパスから構成される manpath を生成するオプション。

実行例

```
manpath -g
```

実行結果

```
/usr/man:/usr/share/man:/usr/local/man:/usr/local/share/man:/usr/X11R6/man:/opt/man:/sna›
›p/man
```

-m SYSTEM もしくは --systems=SYSTEM

このシステムが他のオペレーティングシステムのマニュアル階層にアクセスできる場合????、引数（ SYSTEM ）に指定することで、これを出力に含めるために使用できるオプション。

実行例

```
実行例
```

実行結果

```
実行結果
```

-q もしくは --quiet

警告を発しないオプション。

実行例

```
実行例
```

実行結果

```
実行結果
```

9.3　mcookie

**X 認証システム* で使用するための **マジッククッキー（魔法のクッキー）* を生成するための **ユーティリティ* 。

（* X 認証システム：X Window System という GUI を提供するためのプロトコルとディスプレイサーバに関するソフトウェアプロトコルで使用されるセキュリティ機構のこと。）

（* マジッククッキー：セッション間での認証トークンとして使用されるランダムデータのこと。）

（* ユーティリティ：コンピュータの分析、構成、最適化、保守のためのソフトウェアの総称。ユーティリティソフトウェアとも呼ばれる。）

具体的には、X 認証システムで使用する 128 ビットのランダムな 16 進数を生成する。

実行例

```
mcookie
```

実行結果

```
7f5666105aff03809c17dfb9f2d9ae12
```

♣ オプション一覧

-f もしくは --file FILE

引数（ FILE ）として指定したファイルをクッキーのシードとして使用するオプション。

実行例

```
  mcookie -m 1 -f hoge.txt
```

実行結果

```
  e9f79e7e1edc12fdda16fb3a4dc70e6b
```

-m もしくは --max-size NUM

シードファイルからの読み込み量を制限するオプション。引数（ NUM ）で指定したバイト数をマックスとする。何かしらのファイルやデバイスをシードにする際に指定しないと実行できないため、-f オプションと併用して使用する。

指定するバイトは当たり前だが、そのファイルのディスク使用量以下にする必要がある。以下の実行例で用いている hoge.txt は 5 バイトのファイルである。

実行例

```
mcookie -m 5 -f hoge.txt
```

実行結果

```
17b6822a593be8cab50b8e29cabab42a
```

-v もしくは --verbose

各ソースから読み取ったエントロピーの量とともにランダム性を出所を表示するオプション。

実行例

```
mcookie -m 5 -f create_hoge_files.sh -v
```

実行結果

```
Got 5 bytes from create_hoge_files.sh
Got 128 bytes from getrandom() function
f9370557c62ab6c3703b7b614d6ae34f
```

9.4　mesg

mesg コマンドは、システム上の他のユーザーが write コマンドまたは talk コマンドを使用して、 メッセージを送信できるかどうかを制御するコマンド。引数なしで呼び出されると、mesg コマンドは、ターミナルの現在のメッセージ許可設定を表示する。root 権限を持つユーザーは、メッセージ許可設定に関係なく、どのターミナルにでもメッセージを送信できる。

次に示すフラグによって制御する。

n：メッセージを許可しない。

y：メッセージの表示を許可する。

実行例

```
mesg
```

実行結果

```
is y
```

♣ オプション一覧

なし

-v もしくは --verbose

mesg コマンドは、端末で実行されていない場合にはエラーステータス 2 でサイレントに終了します。この場合、mesg を実行することは無意味です。バージョン 2.33 でこの挙動が導入され、--verbose オプションを使用すると、この状況で mesg が警告を表示するようになりました。

実行例

```
実行例
```

実行結果

```
実行結果
```

第 10 章
頭文字が p のコマンド

10.1 psicc(途中)

色管理プロファイルを適用するとコマンドツール。ICC（International Color Consortium）プロファイルを使用して、画像やグラフィックスの色空間を変更するために使用する。

♣ オプション一覧

-b

-b オプションを使用すると、プロファイルの変換を行う際に、画像やグラフィックスのバックアップファイルを作成できる。

実行例

```
psicc -b input.icc output.icc
```

実行結果

```
バックアップファイルが作成される: "input.icc"ファイルは、変換前の状態を保持したバックアップファイル
として保存される。バックアップファイルの名前は通常、元のファイル名に".bak"などの拡張子が追加されることが
ある。例えば、"input.icc"のバックアップファイルは"input.icc.bak"となる場合がある。
```

-c precision

文章説明

実行例

実行例

実行結果

実行結果

-i profile

文章説明

実行例

実行例

実行結果

実行結果

-n gridpoints

文章説明

実行例

実行例

実行結果

実行結果

-o profile

文章説明

実行例

実行例

実行結果

実行結果

-t intent

文章説明

実行例

```
実行例
```

実行結果

```
実行結果
```

10.2 pslog

pslog コマンドは、プロセスの現在の作業ログを報告する。

♣ オプション一覧

-V

バージョン情報を表示します

実行例

```
pslog -V
```

実行結果

```
pslog (PSmisc) 23.4
Copyright (C) 2015-2017 Vito Mule'.
PSmisc comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it underthe terms of the GNU ›
›General Public License.
For more information about these matters, see the files named COPYING.
```

オプション

文章説明

実行例

```
実行例
```

実行結果

```
実行結果
```

10.3 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

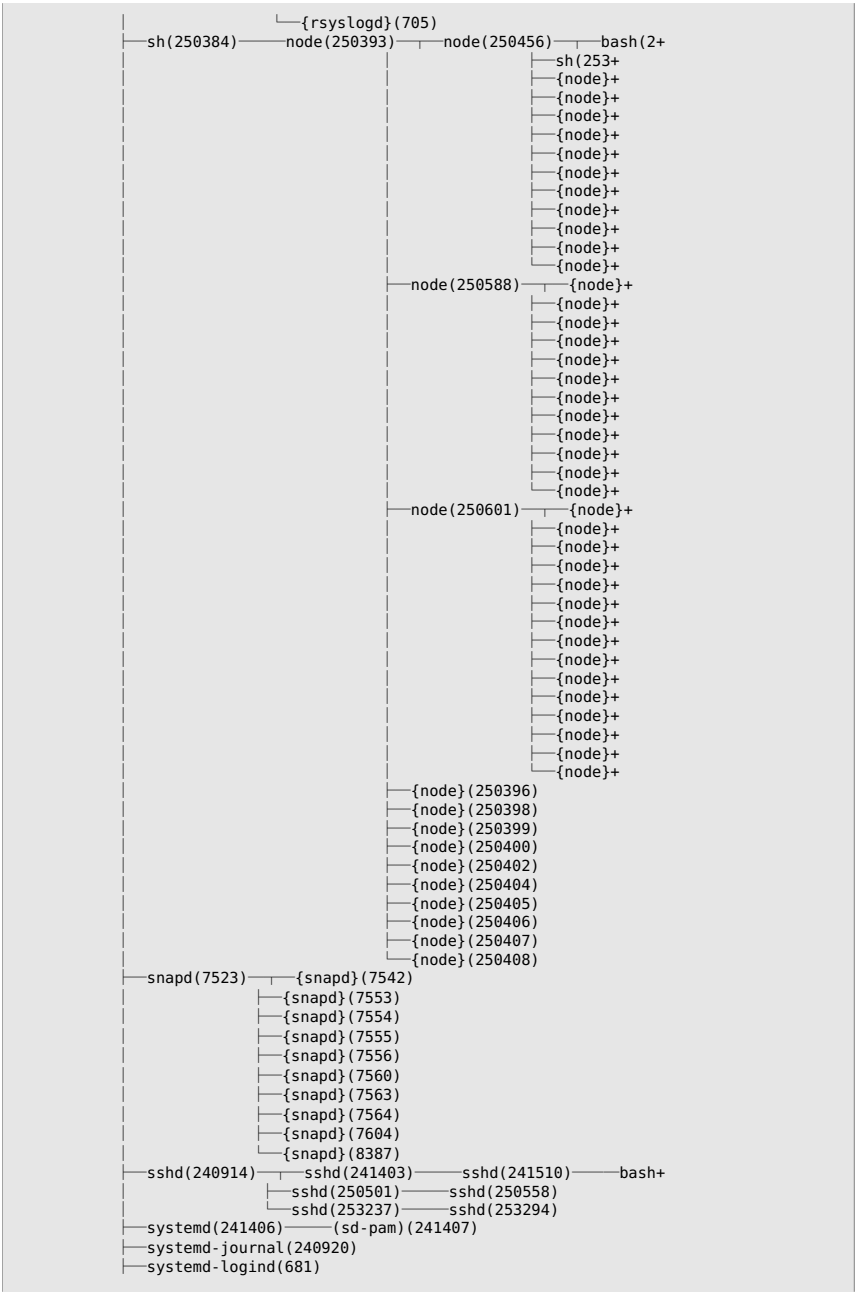
文章説明

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}+
├─node(250601)─┬─{node}+
{node}+
{node}+
{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}]
```

10.4 patar

Perl で書いた tar に似たコマンド。Perl モジュールである Archive::Tar を使用して、ファイルやディレクトリ構造を含めた tar アーカイブの抽出、作成、およびリスト表示を行うためのプログラム。

実行例

```
オプションなしでは実行できない
```

♣ オプション一覧

-c

tar アーカイブを作成する。

実行例

```
ptar -c  file1.text file2.txt
```

実行結果

```
$ ls
file1.text file2.txt //実行前

$ ptar -c  file1.text file2.txt

$ ls
default.tar  file1.text file2.txt/実行後
```

-f

ファイル名の指定ができる。

実行例

```
ptar -c -f test.tar test_1.txt test_2.txt
```

実行結果

```
$ls
test_1.txt  test_2.txt/実行前

$ls
test_1.txt  test_2.txt  test.tar/実行後
```

-t

tar ファイルの中身をリスト表示します。

実行例

```
ptar -t -f test.tar
```

実行結果

```
./test_1.txt
./test_2.txt
```

- **-x**

tar アーカイブは抽出されるを行う。

実行例

```
ptar -xf archive.tar
```

実行結果

```
$ ls //実行前
archive.tar

$ ptar -xf archive.tar

$ ls
archive.tar file1.txt file2.txt
```

-z

zlib 圧縮されたアーカイブファイルを読み書きするためオプション。zlib は、低レベルのデータ圧縮と解凍のためのライブラリおよびフォーマット。

実行例

```
ptar -czf test.tar.gz test_1.txt test_2.txt
```

実行結果

```
$ ls
test_1.txt  test_2.txt  test.tar /実行前

$ ptar -czf test.tar.gz test_1.txt test_2.txt
```

```
$ ls
test_1.txt  test_2.txt  test.tar  test.tar.gz/実行後
```

-v

tar アーカイブの作成または抽出されるファイル名が表示される。

実行例

```
ptar -cvf archive.tar test_1.txt test_2.txt
```

この実行例では、text_1.txt と text_2.txt を使用して archive.tar という名前の tar ファイルを作成している。v オプションを使用することで、上記のコマンドの実行後に text_1.txt と text_2.txt のファイル名が表示される。

実行結果

```
./test_1.txt
./test_2.txt
```

-T

tar ファイルの作成にテキストファイルを使用する。テキストファイルには、tar ファイルの作成に使用するファイルのパスが記載されている。あらかじめテキストファイルを作成することで多くのファイルを tar ファイルの作成に使用できる。

実行例

使用するファイル:filelist.txt

```
ptar -cvf archive.tar -T filelist.txt
```

実行結果

```
$ ls //実行前
filelist.txt

$ cat filelist.txt
/home/user/file1.txt
/home/user/file2.txt

//filelist.txtには、2つのディレクトリ構造が記載されている

$ ptar -cvf archive.tar -T filelist.txt
//filelist.txtに記載されているディレクトリ構造をもとにarchive.tarを作成。

/home/user/file1.txt
/home/user/file2.txt

$ ls //実行後
archive.tar  filelist.txt

$ tar -xf archive.tar
//作成されたtarファイルの確認のため、中身を抽出
```

```
$ ls
archive.tar  filelist.txt  home
// home/userのディレクトリが抽出される。
$ cd home/user
$ ls
file1.txt file2.txt
//その中にfile1.txt file2.txtが含まれる。
```

10.5 ptardiff

指定したアーカイブファイル内のファイルと、カレントディレクトリ内にある同じ名前のファイルとの差分を表示するコマンド。

実行例

```
実行例
```

実行結果

```
実行結果
```

♣ オプション一覧

オプション

文章説明

実行例

```
実行例
```

実行結果

```
実行結果
```

オプション

文章説明

実行例

```
実行例
```

実行結果

```
実行結果
```

オプション

文章説明

実行例

```
実行例
```

実行結果

```
実行結果
```

オプション

文章説明

実行例

```
実行例
```

実行結果

```
実行結果
```

10.6　ptargrep

tar アーカイブ内のファイルの内容にパターンマッチングを適用するためのツール

実行例

```
オプション無しで実行できない
```

♣ オプション一覧

--basename

アーカイブファイル内のファイル名に対して、指定したベース名（パターン）に一致するものを検索するためのコマンド。一致するファイルを抽出する場合、アーカイブからのディレクトリパスを無視し、現在のディレクトリにアーカイブ内のファイルのベース名を使用して書き込む。ただし、アーカイブ内の 2 つの一致するファイルが同じベース名を持つ場合、2 番目のファイルが最初のファイルを上書きする。

実行例

使用するアーカイブファイル

```
test.tar
└── home
    └── user
        ├──file1.txt
        └──file2.txt
```

```
$ ptargrep --basename "file" test.tar
```

実行結果

```
file1.txt
file2.txt
```

--ignore-case

--ignore-case オプションは、大文字と小文字を区別せずに検索を行うためのオプション

実行例

```
$ ptargrep --ignore-case "file" test.tar
```

実行結果

```
File.txt
file.jpg
```

--list-only

文章説明

実行例

```
$ ptargrep --list-only fruits.tar
```

実行結果

```
Apple.txt
apple.jpg
Orange.txt
orange.jpg
Pineapple.txt
```

オプション

文章説明

実行例

実行例

実行結果

実行結果

10.7 ptx

ptx コマンドは、与えられたテキストファイルを処理し、単語の順列索引を生成する。順列索引は、単語をアルファベット順にソートし、単語ごとにその出現箇所や参照情報を提供する。順列索引は、単語を検索したり、単語の出現パターンを分析したりするのに役立つ。

実行例

```
ptx file.txt
```

実行結果

```
$ cat file.txt /実行前

This is a sample file for testing the ptx command.
It contains multiple lines of text with various words.
The ptx command will generate a permuted index of these words.

$ ptx file.txt  /実行後

 with various words.                 It contains multiple lines of text
 permuted index of these/            The ptx command will generate a
 the ptx command.                    This is a sample file for testing
 command.                  This is   a sample file for testing the ptx
     The ptx command will generate   a permuted index of these words.
   sample file for testing the ptx   command.                  This is a
 index of these words/     The ptx   command will generate a permuted
 with various words.            It   contains multiple lines of text
                 This is a sample    file for testing the ptx command.
            This is a sample file    for testing the ptx command.
 words.       The ptx command will   generate a permuted index of these
        will generate a permuted     index of these words.       /command
 ptx command.                This     is a sample file for testing the
             It contains multiple     lines of text with various words.
 various words.        It contains    multiple lines of text with
       It contains multiple lines     of text with various words.
    will generate a permuted index    of these words.         /ptx command
   The ptx command will generate a    permuted index of these words.
     a sample file for testing the    ptx command.                 This is
 permuted index of these/      The    ptx command will generate a
 command.                 This is a   sample file for testing the ptx
         This is a sample file for    testing the ptx command.
     It contains multiple lines of    text with various words.
      is a sample file for testing    the ptx command.               This
      generate a permuted index of    these words.           /command will
       multiple lines of text with    various words.          It contains
 these words.       The ptx command   will generate a permuted index of
```

```
    contains multiple lines of text   with various words.              It
         lines of text with various   words.                    /multiple
          a permuted index of these   words.                /will generate
```

実行結果の中央を見ると、上から「It」,「The」,「This」〜のようにアルファベット順でキーワードの順列索引が生成されます。キーワードの両サイドには前後の文脈が表示される。

♣ オプション一覧

　このオプションを用いると、左の列にファイル名とキーワードが含まれている行番号を表示することができる。ただし、標準入力の場合ファイル名は表示されない。

-A

実行例.1

```
ptx file.txt -A
```

実行結果.1

```
$ cat file.txt /実行前

This is a sample file for testing the ptx command.
It contains multiple lines of text with various words.
The ptx command will generate a permuted index of these words.

$ ptx file.txt -A /実行後
file.txt:2:  of text with various/        It contains multiple lines
file.txt:3:  generate a permuted/         The ptx command will
file.txt:1:  testing the ptx/             This is a sample file for
file.txt:1:  the ptx/           This is   a sample file for testing
file.txt:3:      /command will generate   a permuted index of these/
file.txt:1:    file for testing the ptx   command.            /a sample
file.txt:3:  permuted/          The ptx   command will generate a
file.txt:2:  text with various/      It   contains multiple lines of
file.txt:1:           This is a sample    file for testing the ptx/
file.txt:1:       This is a sample file   for testing the ptx command/
file.txt:3:        The ptx command will   generate a permuted index/
file.txt:3:   /will generate a permuted   index of these words.
file.txt:1:  testing the ptx/      This   is a sample file for
file.txt:2:        It contains multiple   lines of text with various/
file.txt:2:  various/       It contains   multiple lines of
```

実行例.2

```
 cat file.txt | ptx -A
//catでfile.txtの中身を標準出力し、それをpxtで順列索引する。
```

実行結果.2

```
$ cat file.txt /実行前

This is a sample file for testing the ptx command.
It contains multiple lines of text with various words.
The ptx command will generate a permuted index of these words.


$ cat file.txt | ptx -A /実行後


:2:  text with various words.         It contains multiple lines of
:3:  permuted index of these/         The ptx command will generate a
:1:  testing the ptx command.         This is a sample file for
:1:  ptx command.           This is   a sample file for testing the
:3:      /ptx command will generate   a permuted index of these words/
:1:        file for testing the ptx   command.             /is a sample
:3:  permuted index of/     The ptx   command will generate a
:2:  with various words.          It   contains multiple lines of text
:1:  command.      This is a sample   file for testing the ptx
:1:           This is a sample file   for testing the ptx command.
:3:            The ptx command will   generate a permuted index of/
:3:        will generate a permuted   index of these words.    /command
:1:  the ptx command.          This   is a sample file for testing
:2:            It contains multiple   lines of text with various/
:2:  various words/     It contains   multiple lines of text with
:2:      It contains multiple lines   of text with various words.
:3:       generate a permuted index   of these words.     /command will
:3:    /ptx command will generate a   permuted index of these words.
:1:     sample file for testing the   ptx command.            This is a
:3:  permuted index of/         The   ptx command will generate a
:1:  command.             This is a   sample file for testing the ptx
:1:       This is a sample file for   testing the ptx command.
:2:      contains multiple lines of   text with various words.       It
:1:    is a sample file for testing   the ptx command.             This
:3:    generate a permuted index of   these words.                /will
:2:     multiple lines of text with   various words.        It contains
:3:  of these/      The ptx command   will generate a permuted index
:2:           multiple lines of text   with various words.     /contains
:2:      lines of text with various   words.                  /multiple
:3:       a permuted index of these   words.             /will generate
```

-W

正規表現を用いて、順列索引するキーワードを設定できる。

実行例

```
ptx file.txt -W 'It'
```

実行結果

```
$ cat file.txt /実行前

This is a sample file for testing the ptx command.
It contains multiple lines of text with various words.
The ptx command will generate a permuted index of these words.

$ ptx file.txt -W 'It' /実行後
with various words.                It contains multiple lines of text
```

-f

索引のソートや正規表現の大文字と小文字を区別する動作を変更する。索引のソートは大文字と小文字を区別しなくなり、正規表現の大文字と小文字の区別もなくなる

実行例

```
ptx file.txt -f
```

実行結果

```
$ cat file.txt /実行前

This is a sample file for testing the ptx command.
It contains multiple lines of text with various words.
The ptx command will generate a permuted index of these words.


$ ptx file.txt -f /実行後
    This is                        a sample file for testing the ptx
     The ptx command will generate  a permuted index of these words.
   sample file for testing the ptx  command.                   This is a
 index of these words/     The ptx  command will generate a permuted
 with various words.             It  contains multiple lines of text
                 This is a sample   file for testing the ptx command.
             This is a sample file  for testing the ptx command.
 words.       The ptx command will  generate a permuted index of these
         will generate a permuted   index of these words.       /command
 ptx command.                This    is a sample file for testing the
 with various words.                It contains multiple lines of text
             It contains multiple   lines of text with various words.
 various words.        It contains  multiple lines of text with
       It contains multiple lines   of text with various words.
    will generate a permuted index  of these words.         /ptx command
   The ptx command will generate a  permuted index of these words.
     a sample file for testing the  ptx command.                 This is
 permuted index of these/      The  ptx command will generate a
 command.                 This is a sample file for testing the ptx
        This is a sample file for   testing the ptx command.
     It contains multiple lines of  text with various words.
      is a sample file for testing  the ptx command.                This
 permuted index of these/           The ptx command will generate a
     generate a permuted index of   these words.          /command will
 the ptx command.                   This is a sample file for testing
      multiple lines of text with   various words.          It contains
 these words.      The ptx command  will generate a permuted index of
   contains multiple lines of text  with various words.            It
        lines of text with various  words.                   /multiple
         a permuted index of these  words.               /will generate
```

オプション無しの場合では大文字から小文字の順にソートされるが、f オプションを使うと、大文字小文字の区別なしにアルファベット順にソートされる。

-g

実行結果の中央部分にあるスペースを変更するオプション。g の後に数字を指定して幅を変える。デフォルトでは 3 になっている。

実行例

```
ptx -g 7 file.txt
```

実行結果

```
$ cat file.txt /実行前

This is a sample file for testing the ptx command.
It contains multiple lines of text with various words.
The ptx command will generate a permuted index of these words.



$ ptx -g 7 file.txt /実行後
     with various words.                It contains multiple lines of text
     permuted index of/                 The ptx command will generate a
     the ptx command.                   This is a sample file for testing
     command.              This is      a sample file for testing the ptx
        /ptx command will generate      a permuted index of these words.
          file for testing the ptx      command.            This is a sample
     index of/             The ptx      command will generate a permuted
     with various/              It      contains multiple lines of text
                  This is a sample      file for testing the ptx command.
            This is a sample file       for testing the ptx command.
             The ptx command will       generate a permuted index of these/
         /will generate a permuted      index of these words.
     ptx command.              This     is a sample file for testing the
              It contains multiple      lines of text with various words.
     various/          It contains      multiple lines of text with
        It contains multiple lines      of text with various words.
         generate a permuted index      of these words.        /command will
      /ptx command will generate a      permuted index of these words.
        sample file for testing the     ptx command.                This is a
     permuted index/            The     ptx command will generate a
     command.             This is a     sample file for testing the ptx
         This is a sample file for      testing the ptx command.
        contains multiple lines of      text with various words.          It
         a sample file for testing      the ptx command.             This is
                a permuted index of     these words.          /will generate
       multiple lines of text with      various words.           It contains
                    The ptx command     will generate a permuted index of/
             multiple lines of text     with various words.        /contains
         lines of text with various     words.                    /multiple
          a permuted index of these     words.                /will generate
```

-T

出力を TeX ディレクティブとして生成します。TeX ディレクティブは、ptx コマンドによって生成される出力形式の 1 つ。TeX は、文書作成システムであり、特に科学技術文書や数式を含む文書の作成に広く使用される。

実行例

```
ptx file.txt -T
```

実行結果

```
$ cat file.txt /実行前

This is a sample file for testing the ptx command.
It contains multiple lines of text with various words.
The ptx command will generate a permuted index of these words.

$ ptx file.txt -T /実行後
\xx {with various words.}{}{It}{ contains multiple lines of text}{}
\xx {permuted index of these}{}{The}{ ptx command will generate a}{}
\xx {the ptx command.}{}{This}{ is a sample file for testing}{}
\xx {command.}{This is}{a}{ sample file for testing the ptx}{}
\xx {}{The ptx command will generate}{a}{ permuted index of these words.}{}
\xx {}{sample file for testing the ptx}{command}{.}{This is a}
```

```
\xx {index of these words}{The ptx}{command}{ will generate a permuted}{}
\xx {with various words.}{It}{contains}{ multiple lines of text}{}
\xx {}{This is a sample}{file}{ for testing the ptx command.}{}
\xx {}{This is a sample file}{for}{ testing the ptx command.}{}
\xx {words.}{The ptx command will}{generate}{ a permuted index of these}{}
\xx {}{will generate a permuted}{index}{ of these words.}{command}
\xx {ptx command.}{This}{is}{ a sample file for testing the}{}
\xx {}{It contains multiple}{lines}{ of text with various words.}{}
\xx {various words.}{It contains}{multiple}{ lines of text with}{}
\xx {}{It contains multiple lines}{of}{ text with various words.}{}
\xx {}{will generate a permuted index}{of}{ these words.}{ptx command}
\xx {}{The ptx command will generate a}{permuted}{ index of these words.}{}
\xx {}{a sample file for testing the}{ptx}{ command.}{This is}
\xx {permuted index of these}{The}{ptx}{ command will generate a}{}
\xx {command.}{This is a}{sample}{ file for testing the ptx}{}
\xx {}{This is a sample file for}{testing}{ the ptx command.}{}
\xx {}{It contains multiple lines of}{text}{ with various words.}{}
\xx {}{is a sample file for testing}{the}{ ptx command.}{This}
\xx {}{generate a permuted index of}{these}{ words.}{command will}
\xx {}{multiple lines of text with}{various}{ words.}{It contains}
\xx {these words.}{The ptx command}{will}{ generate a permuted index of}{}
\xx {}{contains multiple lines of text}{with}{ various words.}{It}
\xx {}{lines of text with various}{words}{.}{multiple}
\xx {}{a permuted index of these}{words}{.}{will generate}
```

-S

行の終わりまたは文の終わりを指定するために使用される。具体的な正規表現を指定する必要がある。

実行例

```
ptx -S '\.|\?|!' file.txt
//文の終わりを .(ピリオド),?,!に設定。
```

実行結果

```
$ cat file.txt /実行前

This is a sample file for testing the ptx command.
It contains multiple lines of text with various words.
The ptx command will generate a permuted index of these words.


$ ptx -S '\.|\?|!' file.txt
     /for testing the ptx command.   It contains multiple lines of text/
      /of text with various words.   The ptx command will generate a/
 the ptx command. It/                This is a sample file for testing
 command. It contains/     This is   a sample file for testing the ptx
  /. The ptx command will generate   a permuted index of these words.
    sample file for testing the ptx  command. It contains multiple/    /a
       /with various words. The ptx  command will generate a permuted/
  /for testing the ptx command. It   contains multiple lines of text/
 It contains/     This is a sample   file for testing the ptx command.
             This is a sample file   for testing the ptx command. It/
      /words. The ptx command will   generate a permuted index of these/
          will generate a permuted   index of these words.       /command
 ptx command. It/            This   is a sample file for testing the
    /command. It contains multiple   lines of text with various words./
     /the ptx command. It contains   multiple lines of text with/
     /. It contains multiple lines   of text with various words. The/
    will generate a permuted index   of these words.         /ptx command
   The ptx command will generate a   permuted index of these words.    /.
    /a sample file for testing the   ptx command. It contains multiple/
  /of text with various words. The   ptx command will generate a/
 command. It/           This is a   sample file for testing the ptx
```

```
         This is a sample file for   testing the ptx command. It/
  /. It contains multiple lines of   text with various words. The ptx/
     /is a sample file for testing   the ptx command. It contains/
      generate a permuted index of   these words.          /command will
      /multiple lines of text with   various words. The ptx command/
   /various words. The ptx command   will generate a permuted index of/
   contains multiple lines of text   with various words. The ptx/   /. It
       /lines of text with various   words. The ptx command will/
         a permuted index of these   words.               /will generate
```

-r

各入力行の先頭の非空白文字を参照 ID として扱うために使用される。参照 ID は行のテキストとは別に扱われ、インデックス生成時にそれぞれの行に関連付けられる。

実行例

```
ptx -r file.txt
```

実行結果

```
$ cat file.txt /実行前

This is a sample file for testing the ptx command.
It contains multiple lines of text with various words.
The ptx command will generate a permuted index of these words.


$ ptx -r file.txt /実行後
This   ptx command.               is   a sample file for testing the
The        ptx command will generate   a permuted index of these/
This        file for testing the ptx   command.            is a sample
The    permuted index of/        ptx   command will generate a
It     text with various words/        contains multiple lines of
This   command.          is a sample   file for testing the ptx
This               is a sample file   for testing the ptx command.
The    these/       ptx command will   generate a permuted index of
The        /will generate a permuted   index of these words.
This   the ptx command.                is a sample file for testing
It     words.      contains multiple   lines of text with various
It     various words.       contains   multiple lines of text with
It           contains multiple lines   of text with various words.
The        generate a permuted index   of these words.    /command will
The      ptx command will generate a   permuted index of these words.
This     sample file for testing the   ptx command.                is a
The    permuted index of these/        ptx command will generate a
This   ptx command.             is a   sample file for testing the
This           is a sample file for   testing the ptx command.
It        contains multiple lines of   text with various words.
This       a sample file for testing   the ptx command.             is
The             a permuted index of   these words.      /will generate
It       multiple lines of text with   various words.          contains
The    of these/         ptx command   will generate a permuted index
It           multiple lines of text   with various words.     contains
It        lines of text with various   words.                /multiple
The        a permuted index of these   words.                /generate
```

-A オプションでは左端にファイル名と行番号が表現されるが、-r オプションでは各行の先頭に来る単語「This」,「The」,「It」が左端に表示される。

-w

出力行の長さを指定した数字によって設定することができるオプション。

実行例.1

```
ptx -w 20 file.txt
```

実行結果.1

```
$ cat file.txt /実行前

This is a sample file for testing the ptx command.
It contains multiple lines of text with various words.
The ptx command will generate a permuted index of these words.

$ ptx -w 20 file.txt /実行後

           It/
           The ptx/
 a/        This is
     /is   a sample/
        /   a/
    /ptx   command.
    /ptx   command/
      It   contains/
        /   file for/
   /file   for/
   /will   generate/
        /   index of/
    This   is a/
        /   lines of/
        /   multiple/
  /lines   of text/
  /index   of these/
      /a   permuted/
    /the   ptx/
     The   ptx/
   /is a   sample/
    /for   testing/
     /of   text/
        /   the ptx/
     /of   these/
   /with   various/
        /   will/
   /text   with/
        /   words.
  /these   words.
```

実行例.2

```
ptx -w 100 file.txt
```

実行結果.2

```
$ cat file.txt /実行前

This is a sample file for testing the ptx command.
It contains multiple lines of text with various words.
The ptx command will generate a permuted index of these words.

$ ptx -w 100 file.txt /実行後
```

```
      words.                                          It contains multiple lines of text wi›
›th various
    of these words.                                   The ptx command will generate a permu›
›ted index
    command.                                          This is a sample file for testing the›
› ptx
                                              This is   a sample file for testing the ptx com›
›mand.
                        The ptx command will generate   a permuted index of these words.
          This is a sample file for testing the ptx   command.
    words.                                    The ptx   command will generate a permuted inde›
›x of these
    words.                                         It   contains multiple lines of text with ›
›various
                                    This is a sample   file for testing the ptx command.
                                This is a sample file  for testing the ptx command.
                                 The ptx command will  generate a permuted index of these wo›
›rds.
           The ptx command will generate a permuted    index of these words.
                                               This    is a sample file for testing the ptx ›
›command.
                                It contains multiple   lines of text with various words.
                                        It contains    multiple lines of text with various w›
›ords.
                           It contains multiple lines  of text with various words.
         ptx command will generate a permuted index    of these words.                      ›
›          The
                     The ptx command will generate a   permuted index of these words.
              This is a sample file for testing the    ptx command.
    these words.                                  The   ptx command will generate a permuted ›
›index of
                                           This is a    sample file for testing the ptx comma›
›nd.
                          This is a sample file for    testing the ptx command.
                       It contains multiple lines of   text with various words.
                  This is a sample file for testing    the ptx command.
      ptx command will generate a permuted index of    these words.
                           The
            It contains multiple lines of text with    various words.
                                    The ptx command    will generate a permuted index of the›
›se words.
                   It contains multiple lines of text  with various words.
       contains multiple lines of text with various    words.
```

-b

単語の区切りをコンマ (,) やピリオド (.) などの文字で指定することができる。-b オプションを使用しない場合は、タブ文字（\t)、改行文字（\n)、およびスペースが単語の区切りとして扱われる。指定方法は、テキストファイルに記載された文字を参照する。

実行例

```
ptx -b separators.txt file.txt
```

実行結果

```
$ cat file.txt /実行前

This is a sample file for testing the ptx command.
It contains multiple lines of text with various words.
```

```
  The ptx command will generate a permuted index of these words.

  $ cat  separators.txt /単語を区切る文字の出力
  .
  ,
  ;


  $ ptx -b separators.txt file.txt /実行後

   .                                  It contains multiple lines of text with various wor›
›ds
   .                                  The ptx command will generate a permuted index of t›
›hese words
   .                                  This is a sample file for testing the ptx command
```

上の例ではピリオド (.) がヒットし、file.txt 内の各 1 文が単語として区切られている。

-o

順序索引するワードの指定を、ファイル指定によって行うオプション。

実行例

```
  ptx -o keywords.txt file.txt
```

実行結果

```
  $ cat file.txt /実行前

  This is a sample file for testing the ptx command.
  It contains multiple lines of text with various words.
  The ptx command will generate a permuted index of these words.

  $ cat keywords.txt /順序索引するワード表示
  It
  The
  This
  a
  command


  $  ptx -o keywords.txt file.txt /実行後
   with various words.                It contains multiple lines of text
   permuted index of these/           The ptx command will generate a
   the ptx command.                   This is a sample file for testing
   command.                 This is   a sample file for testing the ptx
       The ptx command will generate   a permuted index of these words.
     sample file for testing the ptx   command.                   This is a
   index of these words/     The ptx   command will generate a permuted
```

-i

順序索引しないワードをテキストファイルで指定する。

実行例

```
  ptx -i ignore.txt file.txt
```

実行結果

```
$ cat file.txt /実行前

This is a sample file for testing the ptx command.
It contains multiple lines of text with various words.
The ptx command will generate a permuted index of these words.

$ cat ignore.txt /順序索引しないワードの表示
It
The
This
a
command

$ ptx -i ignore.txt file.txt
with various words.            It    contains multiple lines of text
                  This is a sample   file for testing the ptx command.
             This is a sample file   for testing the ptx command.
 words.       The ptx command will   generate a permuted index of these
         will generate a permuted    index of these words.       /command
 ptx command.                 This   is a sample file for testing the
              It contains multiple   lines of text with various words.
 various words.        It contains   multiple lines of text with
        It contains multiple lines   of text with various words.
    will generate a permuted index   of these words.         /ptx command
   The ptx command will generate a   permuted index of these words.
     a sample file for testing the   ptx command.                 This is
 permuted index of these/      The   ptx command will generate a
 command.               This is a    sample file for testing the ptx
         This is a sample file for   testing the ptx command.
     It contains multiple lines of   text with various words.
      is a sample file for testing   the ptx command.               This
      generate a permuted index of   these words.           /command will
       multiple lines of text with   various words.           It contains
 these words.      The ptx command   will generate a permuted index of
   contains multiple lines of text   with various words.               It
        lines of text with various   words.                   /multiple
         a permuted index of these   words.                /will generate
```

-R

参照行を右に配置する。

実行例

```
ptx -RA file.txt
```

実行結果

```
$ cat file.txt /実行前

This is a sample file for testing the ptx command.
It contains multiple lines of text with various words.
The ptx command will generate a permuted index of these words.

$ ptx -RA file.txt
with various words.                 It contains multiple lines of text     file.txt:2
permuted index of these/            The ptx command will generate a        file.txt:3
the ptx command.                    This is a sample file for testing      file.txt:1
command.                  This is   a sample file for testing the ptx      file.txt:1
    The ptx command will generate   a permuted index of these words.       file.txt:3
  sample file for testing the ptx   command.                  This is a   file.txt:1
index of these words/     The ptx   command will generate a permuted       file.txt:3
with various words.             It  contains multiple lines of text        file.txt:2
                 This is a sample   file for testing the ptx command.      file.txt:1
```

```
            This is a sample file   for testing the ptx command.           file.txt:1
words.       The ptx command will   generate a permuted index of these     file.txt:3
        will generate a permuted    index of these words.       /command    file.txt:3
ptx command.                 This   is a sample file for testing the       file.txt:1
            It contains multiple    lines of text with various words.       file.txt:2
various words.        It contains   multiple lines of text with             file.txt:2
       It contains multiple lines   of text with various words.            file.txt:2
   will generate a permuted index   of these words.         /ptx command   file.txt:3
  The ptx command will generate a   permuted index of these words.         file.txt:3
    a sample file for testing the   ptx command.                  This is   file.txt:1
permuted index of these/      The   ptx command will generate a            file.txt:3
command.                This is a   sample file for testing the ptx        file.txt:1
        This is a sample file for   testing the ptx command.               file.txt:1
    It contains multiple lines of   text with various words.               file.txt:2
    is a sample file for testing   the ptx command.
This   file.txt:1
     generate a permuted index of   these words.           /command will   file.txt:3
      multiple lines of text with   various words.           It contains   file.txt:2
these words.      The ptx command   will generate a permuted index of      file.txt:3
  contains multiple lines of text   with various words.
  It   file.txt:2
```

オプション-A で表示されるファイル名と行番号が右に表示される

-O

出力を ROFF ディレクティブとして生成する。ROFF ディレクティブは、テキストのフォーマットやスタイルを指定するためのコマンドセットである。

実行例

```
ptx -O file.txt > output.roff
```

実行結果

```
$ cat file.txt /実行前

This is a sample file for testing the ptx command.
It contains multiple lines of text with various words.
The ptx command will generate a permuted index of these words.

$ ptx -O file.txt > output.roff

$ cat output.roff /実行後
.xx "with various words." "" "It contains multiple lines of text" ""
.xx "permuted index of these/" "" "The ptx command will generate a" ""
.xx "the ptx command." "" "This is a sample file for testing" ""
.xx "command." "This is" "a sample file for testing the ptx" ""
.xx "" "The ptx command will generate" "a permuted index of these words." ""
.xx "" "sample file for testing the ptx" "command." "This is a"
.xx "index of these words/" "The ptx" "command will generate a permuted" ""
.xx "with various words." "It" "contains multiple lines of text" ""
.xx "" "This is a sample" "file for testing the ptx command." ""
.xx "" "This is a sample file" "for testing the ptx command." ""
.xx "words." "The ptx command will" "generate a permuted index of these" ""
.xx "" "will generate a permuted" "index of these words." "/command"
.xx "ptx command." "This" "is a sample file for testing the" ""
.xx "" "It contains multiple" "lines of text with various words." ""
.xx "various words." "It contains" "multiple lines of text with" ""
.xx "" "It contains multiple lines" "of text with various words." ""
.xx "" "will generate a permuted index" "of these words." "/ptx command"
.xx "" "The ptx command will generate a" "permuted index of these words." ""
.xx "" "a sample file for testing the" "ptx command." "This is"
.xx "permuted index of these/" "The" "ptx command will generate a" ""
```

```
.xx "command." "This is a" "sample file for testing the ptx" ""
.xx "" "This is a sample file for" "testing the ptx command." ""
.xx "" "It contains multiple lines of" "text with various words." ""
.xx "" "is a sample file for testing" "the ptx command." "This"
.xx "" "generate a permuted index of" "these words." "/command will"
.xx "" "multiple lines of text with" "various words." "It contains"
.xx "these words." "The ptx command" "will generate a permuted index of" ""
.xx "" "contains multiple lines of text" "with various words." "It"
.xx "" "lines of text with various" "words." "/multiple"
.xx "" "a permuted index of these" "words." "/will generate"
```

-F

行の切り詰めを示すためのフラグを指定するために使用さる。行の切り詰めとは、テキストが行の幅に収まらずに途中で切り捨てられた場合に、それを示すための特定のフラグや記号を挿入することを指す。

実行例

```
ptx -F "/---" input.txt > output.txt
```

実行結果

```
$ cat input.txt /実行前
This is a sample text that is too long to fit within a single line. /---

$ ptx -F "/---" input.txt > output.txt /実行後

$ $ cat output.txt
   is too long to fit/---             This is a sample text that
 long to fit/---           This is   a sample text that is too
        is too long to fit within   a single line. /---         /---that
     /---text that is too long to   fit within a single line. /-/---
 long to fit/---              This   is a sample text that is too
        /---is a sample text that   is too long to fit within a/---
           to fit within a single   line. /---           /---is too long
    /---a sample text that is too   long to fit within a single/---
 to fit/---              This is a   sample text that is too long
         too long to fit within a   single line. /---        /---that is
                This is a sample    text that is too long to fit/---
            This is a sample text   that is too long to fit/---
        /---text that is too long   to fit within a single line./---
     /---is a sample text that is   too long to fit within a/---
      /---that is too long to fit   within a single line. /---
```

-G

System V の ptx コマンドと同様の動作をするように ptx コマンドを設定する。

実行例

```
$ ptx -G input.txt > output.txt
```

実行結果

```
$ cat input.txt /実行前
This is line 1.
This is line 2.
This is line 3.

$  ptx -G input.txt > output.txt

$ cat output.txt
.xx "" "This is line" "1." ""
.xx "" "This is line" "2." ""
.xx "" "This is line" "3." ""
.xx "" "" "This is line 1." ""
.xx "" "" "This is line 2." ""
.xx "" "" "This is line 3." ""
.xx "" "This" "is line 1." ""
.xx "" "This" "is line 2." ""
.xx "" "This" "is line 3." ""
.xx "" "This is" "line 1." ""
.xx "" "This is" "line 2." ""
.xx "" "This is" "line 3." ""
```

10.8 pwck

Linux システムでユーザやグループのパスワードファイルをチェックするためのコマンド。パスワードファイル内のエントリに関する問題を検出し、修正するのに役立つ。このコマンドを実行するには root ユーザまたは sudo 権限を持つユーザである必要がある。

実行例

```
pwck
```

実行結果

```
user 'lp': directory '/var/spool/lpd' does not exist
user 'news': directory '/var/spool/news' does not exist
user 'uucp': directory '/var/spool/uucp' does not exist
user 'www-data': directory '/var/www' does not exist
user 'list': directory '/var/list' does not exist
user 'irc': directory '/run/ircd' does not exist
user 'gnats': directory '/var/lib/gnats' does not exist
user '_apt': directory '/nonexistent' does not exist
user 'nobody': directory '/nonexistent' does not exist
pwck: no changes
```

♣ オプション一覧

-r

読み込み専用（書き込みできない）でコマンドを実行する

実行例

```
sudo pwck -r
```

実行結果

```
user 'john' has an invalid home directory '/home/john'
user 'jane' has an invalid shell '/bin/false'
user 'testuser' is missing a password entry
```

この実行結果のから、ユーザー「john」のホームディレクトリが無効であること、ユーザー「jane」のシェルが無効であること、そして「testuser」のパスワードエントリが存在しないことが確認できる。

-q

システムやグループ設定に関するエラーや警告を表示するオプション。

実行例

```
pwck -q
```

実行結果

```
user 'john' has an invalid home directory '/home/john'
user 'jane' has an invalid shell '/bin/false'
```

この実行結果から、ユーザー「john」のホームディレクトリが無効であること、ユーザー「jane」のシェルが無効であることがわかる

-R

指定された chroot ディレクトリ内の設定ファイルを使用して、pwck コマンドを実行するためのオプション。chroot とは、プロセスが利用できるルートディレクトリを変更することを指す。

実行例

```
pwck -R /mnt/chroot
```

実行結果

```
user 'john': directory '/mnt/chroot/home/john' does not exist
user 'jane': directory '/mnt/chroot/home/jane' does not exist
```

-s

パスワードファイル (/etc/passwd) と shadow ファイル (/etc/shadow) のエントリを UID (ユーザー識別子) の昇順でソートするオプション。

実行例

```
pwck -s
```

実行結果

```
表示なし
```

10.9 pwconv

etc/passwd（パスワードファイル）をシャドウパスワードファイルへ変換するコマンド。シャドウパスワードファイルとは、パスワードのデータを一般ユーザから保護するために使用するファイルである。このコマンドを実行するには、管理者権限を持つユーザが使用する必要がある。

実行例

```
pwconv
```

実行結果

```
 $ sudo cat passwd | grep user  /実行前のパスワードファイル確認

 user:$6$tR9u4.BHS〜:1000:1000:user:/home/user:/bin/bash

ここでは、パスワードファイルの中にあるユーザエントリを表示している。
表示されたエントリは:(コロン)によって区切られている。

 ・ユーザ名：user
 ・パスワード：$6$tR9u4.BHS〜 (本来なら長めの文字列が表示される。)
 ・ユーザID：1000
 ・グループID：1000
 ・ユーザ情報：user
 ・ホームディレクトリ：/home/user
 ・ログインシェル：/bin/bash
 ここに表示されているパスワードはLinuxで使用されているハッシュ関数に
 よってハッシュ化された文字列である。

 $ pwconv  //実行

 $ cat passwd | grep user
 user:x:1000:1000:user:/home/user:/bin/bash

   pwconvを実行してからもう一度パスワードファイルを確認すると、
 パスワードのフィールドを確認するとxとなっており、ファイルに
 パスワードが記載されていないことが確認できる。

 $ sudo cat shadow- | grep user /シャドウパスワードファルの確認。
user:$6$tR9u4.BHS〜:19635:0:99999:7:::
```

シャドウパスワードファルを確認すると、ユーザのエントリが記載されており、\$6\$tR9u4.BHS〜のようにパスワード情報が記載されている。パスワードのほかに

も、最終パスワード変更日やパスワード有効期限などが記載されている。このシャドウパスワードファル確認するためには、管理者権限を持つユーザである必要がある。

10.10 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
```

10.11　pwd

PWD(1) ユーザーコマンド PWD(1)
名前 pwd - 現在の作業ディレクトリの名前を表示する
概要 pwd
説明現在の作業ディレクトリの完全なファイル名を表示します。
-L もしくは --logical
環境変数の PWD を使用し、シンボリックリンクが含まれていても使用する
-P もしくは --physical
すべてのシンボリックリンクを避ける
--help ヘルプを表示して終了する
--version
バージョン情報を出力して終了する
オプションが指定されていない場合、-P が仮定されます。
注意: シェルには通常、ここで説明されているバージョンよりも優先される独自の pwd がある可能性があります。オプションの詳細については、シェルのドキュメントを参照してください。

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。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

```
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
```

10.12 pwdx

プロセスの作業ディレクトリを表示するコマンド

実行例

```
pwdx [PSID]
```

実行結果.1

```
$ ps //プロセスの確認
  PID TTY          TIME CMD
1071025 pts/0    00:00:00 bash
1072332 pts/0    00:00:00 ps
```

```
$ pwdx 1071025 //bashを対象にpwdxを実行
1071025: /home/user/
```

実行結果

```
$ ps aux //Linuxのシステムが動かしているプロセスを確認
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0 167728 13464 ?        Ss   Oct05   1:44 /lib/systemd/systemd ›
›--system --deserialize 71
root           2  0.0  0.0      0     0 ?        S    Oct05   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   Oct05   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   Oct05   0:00 [rcu_par_gp]
root           5  0.0  0.0      0     0 ?        I<   Oct05   0:00 [slub_flushwq]
 ・
 ・
 ・


$ sudo pwdx 1
1: /
 $ sudo pwdx 2
2: /
 $ sudo pwdx 3
3: /
 $ sudo pwdx 4
4: /
 $ sudo pwdx 5
5: /
 ・
 ・
 ・
```

この例では、Linux のシステムが自動的に実行しているプロセスに対して pwdx を行った。このことからシステムが動かしているプロセスは/(ルート) で動かしていることがわかる。#@#

10.13 pwunconv

pwconv コマンドでシャドウ化した etc/passwd を元に戻すコマンド。シャドウ化とは、パスワードファイルを一般ユーザから閲覧されないようにするための工夫。

実行例

```
pwunconv
```

実行結果

```
$ cat passwd | grep user //実行前のパスワードファイルを確認
user:x:1000:1000:user:/home/user:/bin/bash
ここでは、パスワードファイルの中にあるユーザエントリを表示している。
表示されたエントリは：(コロン)によって区切られている。
・ユーザ名：usre
・パスワード：x
・ユーザID：1000
```

```
・グループID：1000
・ユーザ情報：user
・ホームディレクトリ：/home/user
・ログインシェル：/bin/bash
パスワードフィールドがxとなっている。
これは、パスワードがシャドウ化により閲覧ができなくなっているためである。

$ pwunconv //実行
$ cat passwd | grep user //実行後のパスワードファイルを確認
user:$6$tR9u4.BHS～:1000:1000:user:/home/user:/bin/bash
```

エントリの中にパスワードが表示された。6tR9u4.BHS～は Linux のシステムで使用されるハッシュ関数によって生成される。

10.14　py3compile

python3 のソースファイルをバイトコンパイルするためのコマンド。Python の実行には通常、ソースコードをもとに仮想マシンからバイトコードに変化し、そのバイトコードを用いて PC の機械言語に翻訳する必要がある。ソースコードからバイトコードへ変換することをバイトコンパイルという。py3compile の使用目的は、あらかじめソースコドをバイトコンパイルすることで、プログラムの実行までの時間を短くするために使われることである。

実行例

```
py3compile test.py
```

実行結果

```
$ ls  //実行前
test.py

$ py3compile test.py // 実行

$ ls  //実行後
__pycache__  test.py  //__pycache__ディレクトリが作成される

$ cd __pycache__

$ ls test.cpython-310.pyc //.pycが作成される。
```

♣ オプション一覧

-f

タイムスタンプに関係なく常にバイトコードファイルを再構築するためのオプション。つまり、ソースファイル（.py ファイル）の最終更新日時とバイトコードファイル（.pyc ファイル）の最終更新日時を比較し、ソースファイルが変更されていなくて

もバイトコードファイルを再構築することを指す。

実行例

```
py3compile -f test.py
```

実行結果

```
$ pwd
/home/user/py3compile/__pycache__
$ ls
test.cpython-310.pyc //.pycファイルがすでにある状態

$cd ..
$pwd
home/user/py3compile
$ ls
__pycache__  test.py

$ py3compile -f test.py //実行
$ cd __pycache__
$ ls
test.cpython-310.pyc // 同じ.pyが生成されている。
```

-O

pyo ファイルにコンパイル

実行例

```
py3compile -O test.py
```

実行結果

```
$ ls
test.py

$ py3compile -O test.py

$ ls
__pycache__  test.py

$ cd __pycache__
$ ls
test.cpython-310.opt-1.pyc //.pyoファイルが生成される
```

-v

コンパイルの際に、詳細な実行情報を表示

実行例

```
$ py3compile -v test.py
```

実行結果

```
  D: py3compile:253: argv: ['/usr/bin/py3compile', '-vf', 'test.py']
  D: py3compile:254: options: {'verbose': True, 'force': True, 'optimize': False, 'package
›': None, 'vrange': None, 'regexpr': None}
  D: py3compile:255: args: ['test.py']
```

10.15 pydoc3

Python のモジュール、クラス、メソッド等のドキュメントを生成するためのツール。ドキュメントを作成することで、モジュールに含まれる関数やクラスを確認できる。

実行例

```
pydoc3 math
```

実行結果

```
Help on built-in module math:

  NAME
      math

  DESCRIPTION
      This module provides access to the mathematical functions
      defined by the C standard.

  FUNCTIONS
  acos(x, /)
      Return the arc cosine (measured in radians) of x.

      The result is between 0 and pi.

  acosh(x, /)
      Return the inverse hyperbolic cosine of x.

  asin(x, /)
      Return the arc sine (measured in radians) of x.

      The result is between -pi/2 and pi/2.

  asinh(x, /)
      Return the inverse hyperbolic sine of x.

  atan(x, /)
      Return the arc tangent (measured in radians) of x.

      The result is between -pi/2 and pi/2.

  atan2(y, x, /)
      Return the arc tangent (measured in radians) of y/x.

      Unlike atan(y/x), the signs of both x and y are considered.

  atanh(x, /)
      Return the inverse hyperbolic tangent of x.

  ceil(x, /)
      Return the ceiling of x as an Integral.

      This is the smallest integer >= x.
```

```
        .
        .
        .

DATA
  e = 2.718281828459045
  inf = inf
  nan = nan
  pi = 3.141592653589793
  tau = 6.283185307179586
```

ドキュメントには、「DESCRIPTION」、「FUNCTIONS」、「DATA」の 3 つのタグがある。DESCRIPTION：モジュールの説明文 FUNCTIONS：モジュールに含まれている関数を表示 DATA：モジュール内で使用されている定数の表示

上の例では、math モジュールのドキュメントを表示している。

実行結果から math モジュールには、acos() や asin() などの関数が含まれていることが確認できる。また、自然対数（e）や演習率などの定数も確認できる。

♣ オプション一覧

-w

指定したモジュールのドキュメントを、HTML ドキュメントとして現在いるディレクトリに作成する。

実行例

```
pydoc3 -w math
```

実行結果

```
$ ls //現在のディレクトリにあるファイルを確認
file.txt
$ pydoc3 -w math //コマンドの実効
wrote math.html
$ ls  //実効後のファイルを確認
file.txt math.html
```

-b

pydoc を使用して任意の未使用ポート上で HTTP サーバーをローカルホストで起動し、ウェブブラウザにウェブサイトが自動的に開かれる。このサイトでは、モジュールを検索しドキュメントを表示することができる。

実行例

```
$ pydoc3 -b
```

実行結果

```
Server ready at http://localhost:44161/
Server commands: [b]rowser, [q]uit
server>
```

実行すると、サーバが起動して表示された URL からアクセスできる。また、

-n

指定されたホスト名で HTTP サーバを起動して、ドキュメントを表示するウェブサイトを表示する。

実行例

```
pydoc3 -n 10.20.30.40
```

実行結果

```
Server ready at http://10.20.30.40:41009/
Server commands: [b]rowser, [q]uit
server>
```

-p

指定したポート番号で HTTP サーバを起動して、ドキュメントを表示するウェブサイトを表示する。

実行例

```
pydoc3 -p 5000
```

実行結果

```
Server ready at http://localhost:5000/
Server commands: [b]rowser, [q]uit
server>
```

-k

キーワードを指定して、それに関するモジュールを検索する

実行例

```
pydoc3 -k math
```

実行結果

```
cmath - This module provides access to mathematical functions for complex
math - This module provides access to the mathematical functions
numpy._core._multiarray_umath
numpy._core.umath
```

```
   numpy.core._multiarray_umath
   numpy.core._umath_tests
   numpy.core.tests.test_scalarmath
   numpy.core.tests.test_umath
   numpy.core.tests.test_umath_accuracy
   numpy.core.tests.test_umath_complex
   numpy.core.umath - Create the numpy.core.umath namespace for backward compatibility. In ›
›v1.16
   numpy.core.umath_tests - Shim for _umath_tests to allow a deprecation period for the new›
› name.
   numpy.lib.scimath - Wrapper functions to more user-friendly calling of certain math func›
›tions
   numpy.linalg._umath_linalg
```

上の例では、キーワード「math」を含んでいるモジュールの検索を行っている。

実行結果から"cmath"、"math"、"numpy"などのモジュールが表示されている。

10.16　python3

python

DESCRIPTION Python は、解釈型で、対話的に使えるオブジェクト指向のプログラミング言語です。Python は、非常にわかりやすい構文と強力な機能を備えています。Python には Python チュートリアルがあり、Python のプログラミング入門に役立ちます。Python ライブラリリファレンスでは、組み込みおよび標準の型、定数、関数、およびモジュールについて説明しています。また、Python リファレンスマニュアルでは、コア言語の構文と意味論について詳細に説明しています。Python の基本的な機能は、C や C++ で書かれた独自のモジュールで拡張することもできます。ほとんどのシステムでは、このようなモジュールを動的にロードすることができます。Python は既存のアプリケーションの拡張言語としても適応可能です。インストールされた Python モジュールやパッケージのドキュメントは、pydoc プログラムを実行することで表示することができます。COMMAND LINE OPTIONS

-B .pyc ファイルをインポート時に書き込まない。PYTHONDONTWRITEBYTECODE も参照してください。

-b str(bytes_instance)、str(bytearray_instance)、および bytes もしくは bytearray と str の比較について警告を表示する（-bb: エラーを発生させる）。

-c command

実行するコマンドを指定する（次のセクションを参照）。これにより、オプションリストが終了し、以降のオプションはコマンドの引数として渡されます。

--check-hash-based-pycs mode

ハッシュベースの .pyc ファイルの最新情報の評価方法を設定する。

-d パーサーデバッグ出力を有効にする（専門家向け、コンパイルオプションに

依存)。

-E PYTHONPATH や PYTHONHOME のような環境変数を無視する。これらはインタプリタの動作を変更するために使用されます。

-h もしくは -? もしくは --help

インタプリタの実行ファイルの使用方法を表示し、終了します。

-i スクリプトが最初の引数として渡された場合、または -c オプションが使用された場合、スクリプトまたはコマンドの実行後に対話モードに入る。$PYTHONSTARTUP ファイルは読み込まれません。スクリプトが例外を発生させた場合、グローバル変数やスタックトレースを調査するのに便利です。

-I Python を隔離モードで実行する。これには -E と -s も含まれます。隔離モードでは、sys.path にはスクリプトのディレクトリやユーザーの site-packages ディレクトリは含まれません。また、PYTHON* 環境変数も無視されます。さらに、悪意のあるコードの注入を防ぐために追加の制限が課される場合があります。

-m module-name

名前付きモジュールを sys.path で検索し、対応する .py ファイルをスクリプトとして実行する。これにより、オプションリストが終了し、以降のオプションはモジュールの引数として渡されます。

-O assert 文と __debug__ の値に基づく条件付きコードを削除する；コンパイル（バイトコード）ファイルの拡張子の前に .opt-1 を追加してファイル名を変更する。

-OO -O を行い、また docstrings を破棄する；コンパイル（バイトコード）ファイルの拡張子の前に .opt-2 を追加してファイル名を変更する。

-q バージョンと著作権メッセージを表示しない。これらのメッセージは対話モード以外でも抑制されます。

-s ユーザーサイトディレクトリを sys.path に追加しない。

-S モジュール site のインポートとそれに関連する sys.path の操作を無効にする。site が後で明示的にインポートされた場合も、これらの操作は無効になります。

-u 標準出力と標準エラー出力ストリームを非バッファリングにする。このオプションは標準入力ストリームには影響しません。

-v モジュールが初期化されるたびにメッセージを表示し、それがロードされる場所（ファイル名または組み込みモジュール）を示す。2 回指定すると、モジュールを検索する際にチェックされる各ファイルに対してメッセージを表示します。また、終了時のモジュールのクリーンアップに関する情報も提供します。

-V もしくは --version

実行可能ファイルの Python バージョン番号と終了メッセージを表示します。2 回指定すると、ビルドに関する詳細情報も表示されます。

実行例

```
python3 hello.py
```

実行結果

```
$ cat hello.py
print("Hello World")
$ python3 hello.py
Hello World
```

♣ オプション一覧

-c

コマンドラインから python のスクリプトを直接実効するためのオプション

実行例.1

```
python3 -c "result = 2 + 3; print(result)"
```

実行結果.1

```
5
```

実行例.2

```
python3 -c "import math; result = math.sqrt(25); print(result)"
```

実行結果.2

```
5.0
```

実行例 2 では、モジュールをインポートしている。

-i

python の実行後に対話モードに入る。対話モードに入ると、式の演算や変数の定義を操作などを行うことができる。

実行例

```
python3 -i hello.py
```

実行結果

```
$ cat hello.py
print("Hello World")
$ python3 -i hello.py
Hello World
>>>
対話モードに入ると「>>>」が表示されるようになり、
ここから入力ができる。

$ python3 -i hello.py
Hello World
>>> 2 + 3                 // 2+3の計算
5
>>> "Hello, " + "World!" //Hello World!の表示
'Hello, World!'
>>> 10 > 5                //10>5が正しいかどうかを判断する
True
>>> len([1, 2, 3])        //配列の長さを表示
3
```

-m

Python のモジュールを直接実効するためのオプション

実行例.1

```
python3 -m mymodule
```

実行結果.1

```
$ cat mymodule.py
# mymodule.py

def main():
  print("This is the main function.")

if __name__ == '__main__':
  main()
$ python3 -m mymodule
This is the main function.
```

実行例.2

```
python3 -m http.server
```

実行結果.2

```
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

"http://0.0.0.0:8000/" にアクセスすると、ウェブサイトがブラウザで開かれる。このサイトには、現在いるディレクトリのファイル情報が記載されている。

-O

Python の最適化モードを有効するためのオプション。最適化モードにすることで、実効速度が向上する。

実行例

```
python3  -O time.py
```

実行結果

```
$ cat time.py
import time

def calculate_sum(n):
    start_time = time.time()
    total = 0
    for i in range(1, n+1):
        total += i
    end_time = time.time()
    execution_time = end_time - start_time
    print(f"Sum of numbers from 1 to {n} is: {total}")
    print(f"Execution Time: {execution_time} seconds")

if __name__ == "__main__":
    calculate_sum(10000000)

このコードは、calculate_sum関数が与えられた範囲の数値の総和を計算する。
計算が終わると、プロセスの実行時間が表示される。


$ python3 time.py  //最適化モードなし
Sum of numbers from 1 to 10000000 is: 50000005000000
Execution Time: 0.5327153205871582 seconds
$ python3 -O time.py //最適化モードを適用
Sum of numbers from 1 to 10000000 is: 50000005000000
Execution Time: 0.5239436626434326 seconds
```

オプション無しで実行した場合、時間が 0.532s であるのに対し、

-O オプションありで実行すると時間が 0.523s となり、実行時間が短くなっていることが確認できる。

-B

Python 実行時に.pyc ファイルを生成しないようにするためのオプション。

実行例

```
python3 -Bc "import mymodule"
```

実行結果

```
$ cat mymodule.py //使用するモジュールの表示
# mymodule.py

def main():
    print("This is the main function.")

if __name__ == '__main__':
    main()


$ ls //ディレクトリにあるファイルを確認
mymodule.py
$ python3 -c "import mymodule" //-Bオプション無しで実行
$ ls
mymodule.py  __pycache__
```

```
__pycache__ディレクトリが作成されており、その中には.pycファイルが存在する。

$ rm -r  __pycache__ // __pycache__の削除
$ python3 -Bc "import mymodule"  //-Bオプションありで実行
$ ls
mymodule.py  //__pycache__が作成されていない
```

-v

Python が実行される際に行われる様々な処理や読み込まれるモジュールなどの情報が表示される。

実行例

```
python3 -v hello.py
```

実行結果

```
import _frozen_importlib # frozen
import _imp # builtin
import '_thread' # <class '_frozen_importlib.BuiltinImporter'>
import '_warnings' # <class '_frozen_importlib.BuiltinImporter'>
import '_weakref' # <class '_frozen_importlib.BuiltinImporter'>
import '_io' # <class '_frozen_importlib.BuiltinImporter'>
import 'marshal' # <class '_frozen_importlib.BuiltinImporter'>
import 'posix' # <class '_frozen_importlib.BuiltinImporter'>
import '_frozen_importlib_external' # <class '_frozen_importlib.FrozenImporter'>
# installing zipimport hook
import 'time' # <class '_frozen_importlib.BuiltinImporter'>
import 'zipimport' # <class '_frozen_importlib.FrozenImporter'>
# installed zipimport hook
# /usr/lib/python3.10/encodings/__pycache__/__init__.cpython-310.pyc matches /usr/lib/py›
›thon3.10/encodings/__init__.py
# code object from '/usr/lib/python3.10/encodings/__pycache__/__init__.cpython-310.pyc'
# /usr/lib/python3.10/__pycache__/codecs.cpython-310.pyc matches /usr/lib/python3.10/cod›
›ecs.py
# code object from '/usr/lib/python3.10/__pycache__/codecs.cpython-310.pyc'
import '_codecs' # <class '_frozen_importlib.BuiltinImporter'>
import 'codecs' # <_frozen_importlib_external.SourceFileLoader object at 0x7f7e6fdd32e0>

 .
 .
 .

# cleanup[3] wiping builtins
# destroy _thread
# destroy posix
# destroy _frozen_importlib_external
# destroy _imp
# destroy io
# destroy marshal
# destroy time
# destroy _warnings
# destroy os
# destroy stat
# destroy genericpath
# destroy _abc
# destroy _frozen_importlib
# destroy codecs
# destroy sys
# destroy encodings.aliases
# destroy encodings.utf_8
# destroy _codecs
# destroy builtins
# clear sys.audit hooks
```

10.17 python3

python のプログラムを動かすためのコマンド

実行例

```
python3 hello.py
```

実行結果

```
$ cat hello.py
print("Hello World")
$ python3 hello.py
Hello World
```

♣ オプション一覧

-c

コマンドラインから python のスクリプトを直接実効するためのオプション

実行例.1

```
python3 -c "result = 2 + 3; print(result)"
```

実行結果.1

```
5
```

実行例.2

```
python3 -c "import math; result = math.sqrt(25); print(result)"
```

実行結果.2

```
5.0
```

実行例 2 では、モジュールをインポートしている。

-i

python の実行後に対話モードに入る。対話モードに入ると、式の演算や変数の定義を操作などを行うことができる。

実行例

```
python3 -i hello.py
```

実行結果

```
$ cat hello.py
print("Hello World")
$ python3 -i hello.py
Hello World
>>>
対話モードに入ると「>>>」が表示されるようになり、
ここから入力ができる。

$ python3 -i hello.py
Hello World
>>> 2 + 3                 // 2+3の計算
5
>>> "Hello, " + "World!" //Hello World!の表示
'Hello, World!'
>>> 10 > 5                //10>5が正しいかどうかを判断する
True
>>> len([1, 2, 3])       //配列の長さを表示
3
```

-m

Python のモジュールを直接実効するためのオプション

実行例.1

```
python3 -m mymodule
```

実行結果.1

```
$ cat mymodule.py
# mymodule.py

def main():
  print("This is the main function.")

if __name__ == '__main__':
  main()
$ python3 -m mymodule
This is the main function.
```

実行例.2

```
python3 -m http.server
```

実行結果.2

```
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

"http://0.0.0.0:8000/"にアクセスすると、ウェブサイトがブラウザで開かれる。このサイトには、現在いるディレクトリのファイル情報が記載されている。

-O

Python の最適化モードを有効するためのオプション。最適化モードにすることで、実効速度が向上する。

実行例

```
python3  -O time.py
```

実行結果

```
$ cat time.py
import time

def calculate_sum(n):
    start_time = time.time()
    total = 0
    for i in range(1, n+1):
        total += i
    end_time = time.time()
    execution_time = end_time - start_time
    print(f"Sum of numbers from 1 to {n} is: {total}")
    print(f"Execution Time: {execution_time} seconds")

if __name__ == "__main__":
    calculate_sum(10000000)

このコードは、calculate_sum関数が与えられた範囲の数値の総和を計算する。
計算が終わると、プロセスの実行時間が表示される。


$ python3 time.py  //最適化モードなし
Sum of numbers from 1 to 10000000 is: 50000005000000
Execution Time: 0.5327153205871582 seconds
$ python3 -O time.py //最適化モードを適用
Sum of numbers from 1 to 10000000 is: 50000005000000
Execution Time: 0.5239436626434326 seconds
```

オプション無しで実行した場合、時間が 0.532s であるのに対し、
-O オプションありで実行すると時間が 0.523s となり、実行時間が短くなっていることが確認できる。

-B

Python 実行時に.pyc ファイルを生成しないようにするためのオプション。

実行例

```
python3 -Bc "import mymodule"
```

実行結果

```
$ cat mymodule.py //使用するモジュールの表示
# mymodule.py

def main():
    print("This is the main function.")
```

```
if __name__ == '__main__':
    main()


$ ls //ディレクトリにあるファイルを確認
mymodule.py
$ python3 -c "import mymodule" //-Bオプション無しで実行
$ ls
mymodule.py  __pycache__

__pycache__ディレクトリが作成されており、その中には.pycファイルが存在する。


$ rm -r  __pycache__ // __pycache__の削除
$ python3 -Bc "import mymodule"  //-Bオプションありで実行
$ ls
mymodule.py  //__pycache__が作成されていない
```

10.18　pzstd

ファイルの圧縮を目的としたコマンド。写真や動画などを圧縮することができる。

実行例

```
pzstd file.txt

pzstd -9 file.txt //圧縮レベルの指定

$ pzstd --ultra -22 file1.txt  //より高い圧縮レベルを指定できる
```

実行結果

```
$ ls  //ディレクトリにあるファイルを確認
file1.txt
$ pzstd file1.txt

file1.txt            :2500.00%   (     0 =>     25 bytes, file1.txt.zst)
$ ls  //実行後のディレクトリにあるファイルの確認
file1.txt file1.txt.zst
```

上の例では、1 つのファイルを指定して圧縮ファイルを作成しているが、

複数のファイルを指定して圧縮ファイルを作成できる。例えば、「file1.txt」、「file2.txt」を指定すると、「file1.txt.zst」、「file2.txt.zst」の 2 種類圧縮ファイルが作成される。

ハイフン (-) の後に数字を入れることで、圧縮レベルを指定できる。圧縮レベルは 1 から 19 まである。また、「--ultra」を付けることで圧縮レベルを 22 まで指定できる

♣ オプション一覧

-d

.zst 形式の圧縮ファイルを解凍する

実行例

```
pzstd -d file1.txt.zst
```

実行結果

```
$ ls  //ディレクトリにあるファイルの確認
file1.txt.zst
$ pzstd -d file1.txt.zst  //実行

file1.txt.zst       : 0 bytes
$ ls  //実行後のディレクトリにあるファイルの確認
file1.txt file1.txt.zst
```

-p

圧縮および解凍に使用するスレッド数を指定する。スレッドとは、プロセス内で実行される個々のタスクや処理の単位である。

実行例

```
pzstd -p 4 file1.txt
```

実行結果

```
$ ls //ディレクトリ内のファイルを確認
file1.txt
$ pzstd -p 4 file1.txt
file1.txt            :2500.00%   (     0 =>     25 bytes, file1.txt.zst)
$ ls  //実行後のディレクトリ内のファイルを確認
file1.txt  file1.txt.zst  man.txt
```

-o

出力ファイルの名前を指定する。

実行例

```
pzstd -o test.zst file1.txt
```

実行結果

```
$ ls
file1.txt  file1.txt.zst
$ pzstd -o test.zst file1.txt  // 出力ファイルの名前を"test.zst"に指定
file1.txt            :2500.00%   (     0 =>     25 bytes, test.zst)
$ pzstd -o output.zst file1.txt  // 出力ファイルの名前を"output.zst"に指定
file1.txt            :2500.00%   (     0 =>     25 bytes, output.zst)
$ ls
file1.txt  file1.txt.zst output.zst  test.zst
```

-f

圧縮されたファイルを強制的に上書きする。通常、出力ファイルと同名のファイル

が存在する場合、上書きするかどうかユーザに確認するメッセージが表示されるが、このオプションを使用するとメッセージが表示されなくなる。

実行例

```
pzstd -f file1.txt
```

実行結果

```
$ ls  //ディレクトリ内のファイルを確認
file1.txt file1.txt.zst
$ pzstd file1.txt  //オプション無しで実行
pzstd: file1.txt.zst already exists; do you wish to overwrite (y/n) ? y
//上書きするかどうかの確認メッセージが表示される
file1.txt            :2500.00%   (     0 =>     25 bytes, file1.txt.zst)

$ pzstd -f file1.txt  //-fオプションで実行
file1.txt            :2500.00%   (     0 =>     25 bytes, file1.txt.zst)
//上書きするかどうかの確認メッセージが表示されない
```

-v

処理の詳細を表示する

実行例

```
pzstd -v file1.txt
```

実行結果

```
$ pzstd -v file1.txt
Chosen frame size: 8388608
file1.txt            :2500.00%   (     0 =>     25 bytes, file1.txt.zst)
```

"Chosen frame size" は圧縮に使用されるフレームサイズである。上の例では、フレームサイズが 8388608 バイト (8 メガバイト) であることがわかる。

-r

ディレクトリを指定して、そのディレクトリ内のファイルを圧縮する。圧縮されたファイルは指定したディレクトリ内にある

実行例

```
pzstd -r test
```

実行結果

```
$ ls //現在いるディレクトリ(/test)内のファイルを確認
file1.txt      file2.txt      file3.txt
```

```
$ cd ..
$ pzstd -r test
$ cd test
$ ls //実行後のtestディレクトリ内のファイル確認
file1.txt      file2.txt      file3.txt
file1.txt.zst  file2.txt.zst  file3.txt.zst
```

第 11 章
頭文字が r のコマンド

11.1 rbash

Bash を制限付きモードで起動する。制限付きモードで使用すると、cd コマンドでのディレクトリ移動や、シェルスクリプトの実行などの行動が制限される。「rbash」とターミナルに入力すれば制限モードで Bash が起動する。また、「exit」か「bash」をターミナルで入力することで制限モードを解除することができる。

実行例

```
rbash

または

rbash -r
```

実行結果.1

```
$ rbash
$ cd
rbash: cd: restricted
```

cd コマンドが制限されている。

実行結果.2

```
$ rbash
$ cat test.sh //シェルスクリプトの内容を表示
#!/bin/bash

# 引数が与えられているかチェック
if [ $# -eq 0 ]; then
echo "Usage: $0 [name]"
exit 1
```

```
fi

# 引数の取得とメッセージの表示
name=$1
echo "Hello, $name!"

$ ./test.sh  //シェルスクリプトの実効
rbash: ./test.sh: restricted: cannot specify `/' in command names
```

シェルスクリプトの実行が制限されている。

実行結果.3

```
$ rbash
$ man rbash > output.txt
rbash: output.txt: restricted: cannot redirect output
```

"＞、＞|、＜＞、＞&、＞＞‘などのリダイレクション演算子を使用して出力をリダイレクトすることが制限されている。

11.2 realpath

指定したパスを絶対パスに変換する。

実行例

```
realpath test.txt
```

実行結果

```
/home/user/Prmn
```

♣ オプション一覧

-e

指定したパスが存在すれば、絶対パスが表示される。
存在しない場合はエラーメッセージが表示される。

実行例

```
realpath -e aaa.txt
```

実行結果

```
$ ls    //ディレクトリ内に存在するファイルを確認
test.txt
$ realpath -e test.txt  //存在するファイルパスを指定
/home/user/Prmn
$ realpath -e aaa.txt   //存在しないファイルパスを指定
realpath: aaa.txt: No such file or directory
```

-m

指定したパスが存在しない場合でも絶対パスを表示する。

実行例

```
realpath -m aaa.txt
```

実行結果

```
$ ls    //ディレクトリ内に存在するファイルを確認
test.txt
$ realpath -m aaa.txt  // 存在しないファイルパスを指定
/home/user/Prmn/aaa.txt
```

-s

指定したパスがシンボリックリンクの場合、シンボリックリンクを展開せずにそのままのパスで表示する。シンボリックリンクとは、ファイルやディレクトリを参照するファイルである。

実行例

```
realpath -s read_test/file1.txt
```

実行結果

```
$ ln -s /home/user/test  read_test  //「/home/user/test」を参照するシンボリックリンクを作成
$ cd read_test  //シンボリックリンク内のファイル確認
$ ls
file1.txt  file2.txt
$ cd ..
$ realpath read_test/file1.txt  //オプション無しで実行
/home/user/test/file1.txt    //参照元のパスを表示
$ realpath -s read_test/file1.txt  //オプションありで実行
/home/user/Prmn/read_test/file1.txt //シンボリックリンクを展開せずにパスを表示する
```

第 12 章
頭文字が s のコマンド

12.1 sort

ファイルの内容をソートし、標準出力に出力する。1 文字目が同値だった場合、2 文字目で判断する。3 文字目以降も同様にソートしていく。また、ソートの結果をファイルに出力するためには、リダイレクトを行う。以下の実行例にリダイレクトの書式を示す。実行結果には出力先のファイルである output1.txt の内容を記述する。

実行例

```
sort ex1.txt > output1.txt
```

実行結果

```
1
22
24
477
5666
677
999
```

♣ オプション一覧

-n

数値として並び替える。

実行例

```
sort ex1.txt -n
```

実行結果

```
1
22
24
477
677
999
56666
```

-r

降順でソートする。

実行例

```
sort ex1.txt -r
```

実行結果

```
999
677
56666
477
24
22
1
```

-t

項目の区切り文字を指定する。後述の-k オプションと組み合わせて使用されることが多い。

実行例

```
sort ex2.txt -t :
```

実行結果

```
goto:50
kawai:19
kudo:44
ohira:300
```

-k

ソートの基準となる項目を指定してソートする。前述の-t オプションで区切り文字を指定し、このオプションで区切られた項目を指定してソートする。

実行例

```
sort ex2.txt -t ":" -k 2
```

実行結果

```
kawai:19
ohira:300
kudo:44
goto:50
```

-u

重複したデータを削除して出力する。

実行例

```
sort ex3.txt -u
```

実行結果

```
11
111
222
333
444
555
```

-f

大文字と小文字を区別せずにソートする。

実行例

```
sort ex4.txt -f
```

実行結果

```
nagoya
OSAKA
osaka
TOKYO
tokyo
```

-c

指定したファイルの内容がソートされているかどうかチェックする。正しくソートされている場合は何も出力されないが、正しくソートされていない場合はエラーを出力する。

実行例

```
sort ex1.txt -c
```

実行結果

```
sort: ex1.txt:4: 順序が不規則: 477
```

-m

ソートは行わず、指定した 2 個のファイルをつなげて出力する。

実行例

```
sort ex1.txt -m ex2.txt
```

実行結果

```
1
24
56666
477
999
677
22
kawai:19
ohira:300
goto:50
kudo:44
```

-o

パスで指定したファイルに出力する。実行結果には出力先のファイルの内容を記述する。

実行例

```
sort ex1.txt -o output2.txt
```

実行結果

```
1
22
24
477
56666
677
999
```

-R

ランダムにソートする。

実行例

```
sort ex1.txt -R
```

実行結果

```
56666
24
677
1
999
477
22
```

--debug

ソートの際に注釈をつける。

実行例

```
sort ex1.txt --debug
```

実行結果

```
sort: `ja_JP.UTF-8' ソートルールを使用してテキストの並び替えを実行しています
1
_
22
__
24
__
477
___
56666
_____
677
___
999
___
```

第 13 章
頭文字が t のコマンド

13.1 timedatectl

大まかな文章説明

♣ オプション一覧

-a

文章説明

実行例

```
ls -a
```

実行結果

```
Downloads Music ......................
```

オプション

文章説明

実行例

```
実行例
```

実行結果

```
実行結果
```

13.2 timeout

timeout コマンドは、指定された時間内にコマンドやプロセスを実行し、制限時間が経過すると実行を中断する。※ timeout コマンドは、コマンドがタイムアウトした場合にはステータスコード 124 で終了し、--preserve-status オプションが設定されている場合には実行中のコマンドの終了ステータスを保持する。タイムアウト時には TERM シグナルが送信され、キャッチできない場合には KILL（9）シグナルが使用される。これにより、timeout コマンドはタイムアウト制御とプロセス終了の管理を行う。

実行例

```
timeout 1 ping localhost
```

実行結果

```
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=128 time=0.136 ms
```

♣ オプション一覧

-s もしくは --signal=SIGNAL

タイムアウト時に送信するシグナルを指定。デフォルトでは TERM シグナルを使用。

実行例

```
timeout -s USR1 -k 2 3 ./test-u.sh
```

実行結果

```
test
test
test
User defined signal 1
USER1 signal
test
test
Killed
```

-k もしくは --kill-after=DURATION

初期のシグナルが送信されてから指定した時間が経過してもコマンドが実行中の場合には、KILL シグナルも送信することで確実にコマンドのキルをする。

実行例

```
timeout -k 2 3 ./test-u.sh
```

実行結果

```
test
test
test
Terminated
TERM signal
test
test
Killed
```

13.3 tipc

TIPC（Transparent Inter-Process Communication）は、Linux 上のプロトコルであり、クラスタ化されたコンピュータ環境内のアプリケーション間での通信を可能にする。TIPC コマンドは、TIPC プロトコルの設定と管理を行うためのツールである。

実行例

```
timeout 1 ping localhost
```

実行結果

```
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=128 time=0.136 ms
```

♣ オプション一覧

-j もしくは -json

文章説明

実行例

```
ls -a
```

実行結果

```
Downloads Music ......................
```

-p もしくは pretty

文章説明

実行例

```
実行例
```

実行結果

```
実行結果
```

13.4 tload

現在のシステムの負荷 (load average) を指定された tty にグラフで表示する。

実行例

```
tload
```

実行結果

```
 0.52, 0.58, 0.59
*
*
*
*
*
```

♣ オプション一覧

-s

画面上の縦軸を設定する。load average 1 に対応するグラフのひと目盛が、指定した文字分になる。したがって小さい値を設定するとスケールは大きくなる。逆も同じ。

実行例

```
tload -s 10
```

実行結果

```
0.52, 0.58, 0.59
----------
**********
**********
```

```
**********
**********
**********
**********
```

-d

グラフの更新間隔を秒単位で指定する。

実行例

```
tload -d 10
```

実行結果

```
 0.52, 0.58, 0.59
*
*
*
*
*
```

13.5 man

大まかな文章説明

実行例

```
実行例
```

実行結果

```
実行結果
```

♣ オプション一覧

オプション

文章説明

実行例

```
実行例
```

実行結果

実行結果

オプション

文章説明

実行例

実行例

実行結果

実行結果

13.6 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
```

13.7 touch

各ファイルのタイムスタンプを変更する。主にファイル内容の変更日時 (ctime:Change Time) やファイルへの書き込みや権限などのパラメータ情報の修正日時 (mtime:Modify Time) を変更する。

指定のファイルがない場合は作成される

実行例

```
touch hoge.txt
stat hoge.txt(更新日時を確認)
```

実行結果

```
File: hoge.txt
Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: 802h/2050d      Inode: 5806706     Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/shunsuke)   Gid: ( 1000/shunsuke)
Access: 2023-12-11 00:57:09.763776785 +0900
Modify: 2023-12-11 00:57:09.763776785 +0900
Change: 2023-12-11 00:57:09.763776785 +0900
Birth: 2023-12-11 00:07:45.595626196 +0900
```

♣ オプション一覧

-a

アクセス日時 (atime) と ctime を変更する。

実行例

```
touch -a hoge.txt
stat hoge.txt(更新日時を確認)
```

実行結果

```
File: hoge.txt
Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: 802h/2050d      Inode: 5806706     Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/shunsuke)   Gid: ( 1000/shunsuke)
Access: 2023-12-11 01:00:24.002451121 +0900
Modify: 2023-12-11 00:57:09.763776785 +0900
Change: 2023-12-11 01:00:24.002451121 +0900
Birth: 2023-12-11 00:07:45.595626196 +0900
```

-c もしくは --no-create

touch 実行時、指定ファイルが無い際にファイルを作成しない。

実行例

```
touch -c hoge1.txt
ls
```

実行結果

```
hoge.txt      snap      ダウンロード  ドキュメント  ミュージック
テンプレート  ビデオ      公開          user          デスクトップ  ピクチャ
```

-d もしくは --date=STRING

STRING(日時) を解析して、現在の時刻の代わりに使用する。

実行例

```
touch -d "2002-05-0210:00:00" hoge.txt  (2002年5月2日は筆者の誕生日)
stat hoge.txt
```

実行結果

```
File: hoge.txt
Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: 802h/2050d      Inode: 5806706     Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/shunsuke)   Gid: ( 1000/shunsuke)
Access: 2002-05-02 10:00:00.000000000 +0900
Modify: 2002-05-02 10:00:00.000000000 +0900
Change: 2023-12-11 01:15:56.288362108 +0900
Birth: 2023-12-11 00:07:45.595626196 +0900
```

-f

既存のファイルと同じ名前のファイルが存在していても上書きする。しかし動作的には通常の touch と変わりはない。

明示的にファイルを強制的に上書きするという意味合いが込められているのみ。

実行例

```
同様のため省略
```

実行結果

```
同様のため省略
```

-h もしくは --no-dereference

シンボリックリンクの場合、リンク先ではなくシンボリックそのもののタイムスタンプを変更する。シンボリックリンクとはリンクしているファイルのパス自体のこと。

実行例

```
touch -h link (linkとhoge.txtがつながっている)
stat link (シンボリックリンク自体のタイムスタンプを確認)
stat hoge.txt (sourceのタイムスタンプを確認)
```

実行結果

```
$touch -h link
$stat link
  File: hoge.txt
  Size: 9               Blocks: 8          IO Block: 4096   regular file
  Device: 802h/2050d      Inode: 5806905     Links: 1
  Access: (0664/-rw-rw-r--)  Uid: ( 1000/shunsuke)   Gid: ( 1000/shunsuke)
  Access: 2023-12-12 02:16:22.142100484 +0900
  Modify: 2023-12-12 02:16:22.142100484 +0900
  Change: 2023-12-12 02:16:22.142100484 +0900
  Birth: 2023-12-12 01:42:56.393351377 +0900

$stat hoge.txt
  File: link -> /home/shunsuke/hoge.txt
  Size: 23              Blocks: 0          IO Block: 4096   symbolic link
  Device: 802h/2050d      Inode: 5806911     Links: 1
  Access: (0777/lrwxrwxrwx)  Uid: ( 1000/shunsuke)   Gid: ( 1000/shunsuke)
  Access: 2023-12-12 02:16:44.658122212 +0900
  Modify: 2023-12-12 02:16:44.658122212 +0900
  Change: 2023-12-12 02:16:44.658122212 +0900
  Birth: 2023-12-12 02:08:13.509685780 +0900
```

シンボリックリンクと元ファイルの hoge.txt のタイムスタンプを比較すると違う。

touch link をすれば元の hoge.txt のタイムスタンプも変更される。

-m

変更時刻（mtime）のみ変更する。

実行例

```
touch -m hoge.txt
stat hoge.txt
```

実行結果

```
File: hoge.txt
Size: 9               Blocks: 8          IO Block: 4096   regular file
Device: 802h/2050d      Inode: 5806905     Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/shunsuke)   Gid: ( 1000/shunsuke)
Access: 2023-12-12 02:25:55.299271733 +0900
Modify: 2023-12-12 02:47:38.463970752 +0900
Change: 2023-12-12 02:47:38.463970752 +0900
Birth: 2023-12-12 01:42:56.393351377 +0900
```

-r もしくは --reference=FILE

現在の時刻の代わりに、指定したファイルのタイムスタンプを使用する。

実行例

```
touch -r hoge_source.txt hoge.txt
stat hoge.txt
```

実行結果

```
File: hoge.txt
Size: 9               Blocks: 8          IO Block: 4096   regular file
Device: 802h/2050d      Inode: 5806905     Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/shunsuke)   Gid: ( 1000/shunsuke)
Access: 2023-12-12 02:25:55.299271733 +0900
Modify: 2023-12-12 02:47:38.463970752 +0900
Change: 2023-12-12 02:47:38.463970752 +0900
Birth: 2023-12-12 01:42:56.393351377 +0900
```

-r もしくは --reference=FILE

タイムスタンプで現在の時刻の代わりに、

実行例

```
touch -t 202312111234.56 hoge.txt
stat hoge.txt
```

実行結果

```
File: hoge.txt
Size: 9               Blocks: 8          IO Block: 4096   regular file
Device: 802h/2050d      Inode: 5806905     Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/shunsuke)   Gid: ( 1000/shunsuke)
Access: 2023-12-11 12:34:56.000000000 +0900
Modify: 2023-12-11 12:34:56.000000000 +0900
Change: 2023-12-12 03:13:33.025723661 +0900
Birth: 2023-12-12 01:42:56.393351377 +0900
```

--time=WORD

タイムスタンプの変更するタイムを指定できる。しかし WORD が acces,atime,use の場合は-a のオプションと同様である。また WORD が modify,mtime のときは-m と同様である。

実行例

```
touch --time=aime
stat hoge.txt
```

実行結果

```
File: hoge.txt
Size: 9               Blocks: 8          IO Block: 4096   regular file
Device: 802h/2050d      Inode: 5806905     Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/shunsuke)   Gid: ( 1000/shunsuke)
Access: 2023-12-12 03:19:11.378151886 +0900
Modify: 2023-12-11 12:34:56.000000000 +0900
Change: 2023-12-12 03:19:11.378151886 +0900
Birth: 2023-12-12 01:42:56.393351377 +0900
```

13.8　tr

標準入力から文字を変換、スクイーズ、削除し、標準出力に書き出す。

実行例

```
cat file1.txt | tr a-z a  (file1の中身を出力し、aからzまでの文字を全てaに置き換えるという意味)
```

実行結果

```
aaaaa,aaaaa!!
I aa Laaaa.
```

♣ オプション一覧

-d もしくは --delete

「-d」オプションで、指定した文字を削除することができる。一例として、Windows 環境で作成したファイルの改行コードを、Linux 環境向けに置き換えるといった用途に使用できる。

実行例

```
cat file1.txt | tr -d ami
```

実行結果

```
hello,world!!
I  Lnux.
```

-s もしくは --squeeze-repeats

指定した文字が 1 回だけ出現するように置き換える。一例として、複数改行して間が大きく空いている文書などの間も縮めることができる。

実行例

```
cat file1.txt | tr -s a
```

実行結果

```
hello,world!!
I am Linux.
```

-t もしくは --truncate-set1

置換する文字列の長さや個数に合わせて置換が行われる。連続している文字列は一つに集約されて出力される。

実行例

```
cat file1.txt | tr  -s hel! HEL?
```

実行結果

```
HELo,worLd?
I am Linux.
```

13.9　tzselect

本体に保存されているそれぞれの地域のタイムゾーンを調べるコマンドである。タイムゾーンを設定する際に用いる文字列を確認することもできる。他の国の時刻を知りたいときや、どんなタイムゾーンがあるのか知りたいときに便利である。

tzselect はシェルからパラメータなしで呼び出され、地域のリストが表示される。数字で地域を選ぶと、その地域にある国と都市のリストが表示される。Enter キーを押せば、リストを再表示できる。タイムゾーンを選ぶには、その左の数字を押す。Ctrl-C を押すと、いつでもスクリプトを中断できる。tzselect は実際にはタイムゾーンを変更しない。変更するには"dpkg-reconfigure","tz-date"を用いる。

実行例

```
tzselect
```

実行結果

```
Please identify a location so that time zone rules can be set correctly.
Please select a continent, ocean, "coord", or "TZ".
 1) Africa
 2) Americas
 3) Antarctica
 4) Asia
 5) Atlantic Ocean
 6) Australia
 7) Europe
 8) Indian Ocean
 9) Pacific Ocean
10) coord - I want to use geographical coordinates.
11) TZ - I want to specify the timezone using the Posix TZ format.
    #? 4(入力)
Please select a country whose clocks agree with yours.
 1) Afghanistan              20) Iran                    39) Palestine
 2) Antarctica               21) Iraq                    40) Philippines
 3) Armenia                  22) Israel                  41) Qatar
 4) Azerbaijan               23) Japan                   42) Russia
 5) Bahrain                  24) Jordan                  43) Ré union
 6) Bangladesh               25) Kazakhstan              44) Saudi Arabia
 7) Bhutan                   26) Korea (North)           45) Seychelles
 8) Brunei                   27) Korea (South)           46) Singapore
 9) Cambodia                 28) Kuwait                  47) Sri Lanka
10) China                    29) Kyrgyzstan              48) Syria
11) Christmas Island         30) Laos                    49) Taiwan
12) Cocos (Keeling) Islands  31) Lebanon                 50) Tajikistan
13) Cyprus                   32) Macau                   51) Thailand
14) East Timor               33) Malaysia                52) Turkmenistan
15) French S. Terr.          34) Mongolia                53) United Arab Emirates
16) Georgia                  35) Myanmar (Burma)         54) Uzbekistan
17) Hong Kong                36) Nepal                   55) Vietnam
18) India                    37) Oman                    56) Yemen
19) Indonesia                38) Pakistan
    #? 23(入力)

The following information has been given:

        Japan

Therefore TZ='Asia/Tokyo' will be used.
Selected time is now:   Tue Jan  9 18:45:08 JST 2024.
Universal Time is now:  Tue Jan  9 09:45:08 UTC 2024.
Is the above information OK?
1) Yes
2) No
   #? 1(入力)

You can make this change permanent for yourself by appending the line
        TZ='Asia/Tokyo'; export TZ
to the file '.profile' in your home directory; then log out and log in again.

Here is that TZ value again, this time on standard output so that you
can use the /usr/bin/tzselect command in shell scripts:
Asia/Tokyo
```

♣ オプション一覧

-c

コード

大陸、国、都市の順に入力する代わりに
緯度経度を打ち込み、最も近い都市を含むタイムゾーンから選択する。
COORD は ISO 6709 表記を使用する。
日本なら北緯 35 度 41 分、東経 139 度 41 分であるため、「+3541+13941」であり、ブエノスアイレスならば「-35-058」or「-3436-05826」である。
「緯度経度だけでも理解してくれる。

実行例

```
tzselect -c +35+139
```

実行結果

```
Please identify a location so that time zone rules can be set correctly.
Please select one of the following timezones, echo listed roughly in increasing order of distance from +35+139.
 1) Japan                                      6) Guam, Northern Mariana Islands
 2) Russia - MSK+08 - Sakhalin Island          7) Russia - MSK+06 - Tomponsky, Ust-Maysky
 3) Russia - MSK+07 - Amur River               8) China - Beijing Time
 4) Korea (South)                              9) Russia - MSK+07 - Oymyakonsky
 5) Korea (North)                             10) Palau
    #? 1

The following information has been given:

        coord +35+139
        Japan

Therefore TZ='Asia/Tokyo' will be used.
Selected time is now:   Tue Jan  9 19:00:09 JST 2024.
Universal Time is now:  Tue Jan  9 10:00:09 UTC 2024.
Is the above information OK?
1) Yes
2) No
   #? 1

You can make this change permanent for yourself by appending the line
        TZ='Asia/Tokyo'; export TZ
to the file '.profile' in your home directory; then log out and log in again.

Here is that TZ value again, this time on standard output so that you
can use the /usr/bin/tzselect command in shell scripts:
Asia/Tokyo
```

-n

c を使用したとき、地域リスト表示の最大値を設定する（デフォルトは 10）。

実行例

```
tzselect -c +35+139 -n 2
```

実行結果

```
tzselect -c +35+139 -n 2
Please identify a location so that time zone rules can be set correctly.
Please select one of the following timezones, echo listed roughly in increasing order of distance from +35+139.
```

```
1) Japan
2) Russia - MSK+08 - Sakhalin Island
#? 1

The following information has been given:

        coord +35+139
        Japan

Therefore TZ='Asia/Tokyo' will be used.
Selected time is now:   Tue Jan  9 19:01:59 JST 2024.
Universal Time is now:  Tue Jan  9 10:01:59 UTC 2024.
Is the above information OK?
1) Yes
2) No
#? 1

You can make this change permanent for yourself by appending the line
        TZ='Asia/Tokyo'; export TZ
to the file '.profile' in your home directory; then log out and log in again.

Here is that TZ value again, this time on standard output so that you
can use the /usr/bin/tzselect command in shell scripts:
Asia/Tokyo
```

第 14 章
頭文字が u のコマンド

14.1 uniq

ファイル同士を比較し、連続して重複している部分を削るコマンドである。オプションなしだと、指定したファイルに上書きされる。長いオプションに必須の引数は、短いオプションにも必須である。基本ファイル内は sort 済みであることを想定されているため、Sort は"必須"である。

実行例

```
sort File2.txt -o File2.txt  (uniqにはsortが必要なため)
cat File2.txt  (catはそれぞれのtxtを確認するため)
uniq FIle2.txt
```

実行結果

```
$cat File2.txt
    Blossom
    blossom
    blooooooooooossom
    Elephant
    Radiant
    Serendipity
    Serendipity
$uniq FIle2.txt
    Blossom
    blossom
    blooooooooooossom
    Elephant
    Radiant
    Serendipity
```

♣ オプション一覧

-c もしくは --count

出現回数を単語の行ごと先頭に明記する。

実行例

```
uniq -c File2.txt
```

実行結果

```
   1 Blossom
   1 blossom
   1 blooooooooooossom
   1 Elephant
   1 Radiant
   2 Serendipity
```

-d もしくは --repeated

グループごとに 1 行ずつ、重複している部分を出力する。

実行例

```
uniq -d File2.txt
```

実行結果

```
Serendipity
```

-D もしくは --all-repeated[=METHOD]

重複する行を全て出力する。

「--all-repeated=none」->重複行のグループに区切りをいれない。(デフォルト)

「--all-repeated=prepend」->重複行のグループの手前に区切りをいれる。

「separate」で、グループの間に区切りの空白行をいれる。

実行例

```
uniq -D File2.txt
```

実行結果

```
Serendipity
Serendipity
```

-f もしくは --skip-fields=N

行全体ではなく、スペースやタブ文字で区切られた項目で重複を判断する。指定する際は一番左の項目から 0,1 の順で指定する。

実行例

```
cat File1.txt
uniq -f 2 File1.txt
```

実行結果

```
$cat File1.txt
  1 Blossom 95
  2 Blossom 20
  3 Harmony 90
  4 Radiant 21
  5 Sunshine 21
  6 Whisper 21

$uniq -f 2 File1.txt
  1 Blossom 95
  2 Blossom 20
  3 Harmony 90
  4 Radiant 21
```

--group[=METHOD]

すべての項目を表示し、グループを空行で区切る。METHOD に関してはオプション-D で記述したものと同様である。

しかし上記 3 つに加えて"both"が存在する。これは先頭と最後それぞれに空行を入れる。

実行例

```
cat File2.txt
uniq --group File2.txt
```

実行結果

```
$cat File2.txt
  Blossom
  blossom
  blooooooooooossom
  Elephant
  Radiant
  Serendipity
  Serendipity
$uniq --group File2.txt
  Blossom

  blossom

  blooooooooooossom

  Elephant

  Radiant

  Serendipity
  Serendipity
```

-i もしくは --ignore-case

比較の際、大文字と小文字の違いは無視する。

実行例

```
uniq -i File2.txt
```

実行結果

```
Blossom
blooooooooooossom
Elephant
Radiant
Serendipity
```

-s もしくは --skip-chars=N

文頭から N 文字までを比較対称としない。

実行例

```
uniq -s 1 File2.txt
```

実行結果

```
Blossom (Blossomは頭文字以外同じため消えている)
blooooooooooossom
Elephant
Radiant
Serendipity
```

-u もしくは --unique

重複していない行だけを出力する。

実行例

```
uniq -u  -i File2.txt (今回は小文字、大文字のの違いを無視する)
```

実行結果

```
blooooooooooossom
Elephant
Radiant
```

-z もしくは --zero-terminated

行の区切り記号は改行ではなく NULL であるゼロバイト文字にする。

実行例

```
uniq -z File2.txt
```

実行結果

```
最後に改行されない他に変化しないため記載しない。
```

-w もしくは --check-chars=N

N 文字までを比較する。

実行例

```
cat File2.txt
uniq -w 3 File2.txt
uniq -w 4 File2.txt
```

実行結果

```
$cat File2.txt
  Blossom
  blossom
  blooooooooooossom
  Elephant
  Radiant
  Serendipity
  Serendipity
$uniq -w 3 File2.txt
  Blossom
  blossom
  Elephant
  Radiant
  Serendipity
$uniq -w 4 File2.txt
  Blossom
  blossom
  blooooooooooossom
  Elephant
  Radiant
  Serendipity
```

第 15 章
頭文字が w のコマンド

15.1 wc

テキストファイルの行数や単語数、文字数、バイト数を表示するコマンド
単語は、空白や改行文字で区切られたものを数える
ファイルがない場合は標準入力を読み込む
オプションにより改行、単語、文字、バイト、最大行長の順に選択することができる

wc

実行例

```
wc file.txt
```

実行結果

```
1 1 6 file.txt
```

♣ オプション一覧

-c もしくは --bytes

バイト数を出力する

実行例

```
wc -c file.txt
```

実行結果

```
6 file.txt
```

-m もしくは --chars

文字数を出力する

実行例

```
wc -m file.txt
```

実行結果

```
6 file.txt
```

-l もしくは --lines

行数を出力する

実行例

```
wc -l file.txt
```

実行結果

```
1 file.txt
```

--files0-from=F

ファイルに入力されている NULL 文字で区切られたファイル名のリストを指定する

"--files0-from=-"とした場合、ファイルを標準入力から読み込む

sort や find と併用して使い、ファイルサイズなどを出力できる

実行例

```
wc -m -files0-from=filename.txt
```

実行結果

```
6 /home/ubuntu/file.txt
```

-L もしくは --max-line-length

最大の表示幅を出力する

実行例

```
wc -L file.txt
```

実行結果

```
5 file.txt
```

-w もしくは --words

単語数を出力する

実行例

```
wc -w file.txt
```

実行結果

```
1 file.txt
```

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 v0.3.0

Selected Unix Commands for Beginners

2024 年 2 月 29 日　v0.3.0

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)