ファイルの所有者とパーミッション

◎アクセス権

```
r(4) 読み取り
w(2) 書き込み
x(1) 実行
```

```
chmod u+r #所有ユーザに読み込み権限を与える
chmod g+w #所有グループに書き込み権限を与える
chmod o+r #そのほかユーザに実行権限を与える
chmod 111 #所有ユーザに1、所有グループに1、そのほかのユーザに1の権限を与える
```

```
ls -l
drwxr-xr-x. 3 root root   45 12月  7 10:12 GCC_TEST
(d)rwx  所有ユーザのアクセス権(u)
   r-x  所有グループのアクセス権(g)
   r-x  その他のアクセス権(0)

   chmod ug+X dir1
   chmod g-wr dir1

groups  #現在のユーザが所属しているグループを表示する
usermod -g グループ名  #現在のユーザが所属しているグループを変更できる
```

```
* SUID (SUIDが設定されているファイルが実行されると所有ユーザの権限で実行される)
    * chmod u+s Filename
    * chmod 4XXX Filename
* SGID (SGIDが設定されているファイルが実行されると所有グループの権限で実行される)
    * chmod g+s Filename
    * chmod 2XXX Filename
* スティッキービット : ディレクトリに設定すると、他のユーザがファイルを削除できなくなる
    * chmod o+t Dirname
    * chmod 1XXX Dirname
```

- chmod
- umask
- chown
- chgrp
- ls
    - ls -1 縦に並べて表示
    - ls -l　詳細（所有者、アクセス権など）も含んだ情報を表示

    - ls -a　.から始まるファイル（隠しファイル）も表示
    - ls -r　逆順に表示
    - ls -R　ディレクトリ内を再帰的にすべて表示
    - ls -t　更新日時でソートして表示

◎ファイル・ディレクトリの操作と管理

- ls
- file
- touch
    - touch -t CCYYMMDDhhmm Filename
- cp
    - cp -f(--force)
    - cp -i(--interactive)
    - cp -p(--preserくえうくえう
    - cp -r(--recursive)
- mv
    - mv -f(--force)
    - mv -i(--interactive)
- rm
    - rm -f(--force)
    - rm -i(--interactive)
    - rm -r(--recursive)
- mkdir
    - mkdir -m(--mode)
    - mkdir -p(--parents)
- rmdir
    - rmdir -p
- tar
    - -c(--create) : 圧縮する
    - -x(--file) : アーカイブファイル名を指定
    - -f(--list) : アーカイブファイルの内容を表示
    - -v(--verbose) : 処理したファイルの一覧を詳細に出力
    - -x(--extract) : 解凍する
    - -z(--gzip) : gzipで圧縮、解凍
    - -j(--bzip2) : bzip2で圧縮、解凍
    - -J(--xz) : xzで圧縮、解凍
- gzip
    - gzip -d
    - gzip -r
- gunzip
- bzip2
    - bzip2 -d
- xz
    - xz -d
- ファイル名のパターンマッチ
    - * (アスタリスク) : 任意の0文字以上の文字列

- ? (クエスチョンマーク) : 任意の1文字
- [0-9] : 0-9の数字
- [a-z] : a-zのアルファベット

◎ファイルとディレクトリ操作と管理（ハードリンク）

```
* Linuxのファイルシステムでは「ファイルの中身」「ファイルの属性や管理情報」は別に保存されていて、iノードは「ファイルの属性や管理情報」が保存されている
* iノードには、以下の情報が記載されている
    * ファイルの種別
    * ファイルサイズ
    * アクセス権
    * 所有者
    * リンク
    * ディスク上の物理的な保存場所（ブロック番号）
iノードを確認するには
ls -i

ファイルに対してリンクを作成する場合、リンクにはハードリンク/シンボリックリンクがある
ln Filename リンク先 （ハードリンクの作成）
ln -s Filename リンク先（シンボリックリンクの作成）
```

```
[root@localhost link]# ls -li
9552719 -rw-r--r--. 1 root root 5  4月  7 18:30 test.cpp.txt
9552718 -rw-r--r--. 2 root root 5  4月  7 18:30 test.hardlink.txt
9552720 lrwxrwxrwx. 1 root root 8  4月  7 18:32 test.symbolic.txt -> test.txt
9552718 -rw-r--r--. 2 root root 5  4月  7 18:30 test.txt
[root@localhost link]# mkdir next
[root@localhost link]# mv test.txt next/
[root@localhost link]# ls -li
13070473 drwxr-xr-x. 2 root root 22  4月  7 18:46 next
 9552719 -rw-r--r--. 1 root root  5  4月  7 18:30 test.cpp.txt
 9552718 -rw-r--r--. 2 root root  5  4月  7 18:30 test.hardlink.txt
 9552720 lrwxrwxrwx. 1 root root  8  4月  7 18:32 test.symbolic.txt -> test.txt
[root@localhost link]# cd next/
[root@localhost next]# ls -li
9552718 -rw-r--r--. 2 root root 5  4月  7 18:30 test.txt
[root@localhost next]# mv test.txt /boot/
[root@localhost next]# cd ..
[root@localhost link]# ls -li
13070473 drwxr-xr-x. 2 root root 6  4月  7 18:47 next
 9552719 -rw-r--r--. 1 root root 5  4月  7 18:30 test.cpp.txt
 9552718 -rw-r--r--. 1 root root 5  4月  7 18:30 test.hardlink.txt
 9552720 lrwxrwxrwx. 1 root root 8  4月  7 18:32 test.symbolic.txt -> test.txt
[root@localhost link]#[root@localhost link]# ls -li
9552719 -rw-r--r--. 1 root root 5  4月  7 18:30 test.cpp.txt
9552718 -rw-r--r--. 2 root root 5  4月  7 18:30 test.hardlink.txt
9552720 lrwxrwxrwx. 1 root root 8  4月  7 18:32 test.symbolic.txt -> test.txt
9552718 -rw-r--r--. 2 root root 5  4月  7 18:30 test.txt
```

※ハードリンクを作ったのちコピー元のファイルを移動した場合、(つまりファイルシステム間を移動した場合、)コピー元のiノードが変わる。そして、ハードリンクの内容を書き換えるとハードリンクも新しいiノードが採番される。

```
[root@localhost link]# vi test02.txt
[root@localhost link]# ls -li
 9552722 -rw-r--r--. 1 root root 5  4月  7 18:53 test02.txt
[root@localhost link]# ln test02.txt test02.link
[root@localhost link]# ls -li
 9552722 -rw-r--r--. 2 root root 5  4月  7 18:53 test02.link
 9552722 -rw-r--r--. 2 root root 5  4月  7 18:53 test02.txt
[root@localhost link]# mv test02.txt /boot
[root@localhost link]# ls -li
 9552722 -rw-r--r--. 1 root root 5  4月  7 18:53 test02.link
[root@localhost link]# vi test02.link
[root@localhost link]# ls -li
 9552723 -rw-r--r--. 1 root root 12  4月  7 18:54 test02.link
```

※シンボリックリンクは、iノードではなくファイル名に紐づくので、ファイル名を変えると機能しなくなる
※シンボリックリンクは、ディレクトリにも適応できる
※ハードリンクは別のファイルシステムには作れない（シンボリックリンクは作れる）

◎ファイルの配置と検索

- find
  - -name
  - -atime　（最終アクセス日 -2 -3 +1 など）
    - find ./ -atime -2 -type f
  - -u
  - -l
  - -mtime　(最終更新日時)
  - -perm
  - -size
  - -user
  - -type　（fならファイル）
  - -maxdepth
  - -mindepth
  - -exec コマンド {} ;
    - find ./ -mtime +1 -type f -exec rm {} ;

```
現在から3日前まで（現在時間～72時間前）
    # find ./ -mtime -3
3日前（72時間前～96時間前）
    # find ./ -mtime 3
過去から3日前まで（72時間前～過去）
    # find ./ -mtime +2
```

- which
- locate (findより高速に検索が可能：updatedbコマンドを更新する必要がある)
    - /etc/updatedb.conf
        - 検索対象から外す
            - PRUNE_BIND_MOUNT
            - PRUNEFS
            - PRUNENAMES
            - PRUNEPATHS
- whereis (コマンドの場所やマニュアルの場所を示す)
- type (コマンドの情報を表示する)
    - -a
    - -p
    - -t

```
代表的なファイルシステムに格納されているもの
/dev : ハードディスク、DVD-ROMなどのデバイスファイル
/etc : システム、コマンドなどの設定ファイル
/lib : 共有ライブラリやモジュール、/bin、/sbinにあるコマンドが利用するライブラリ
/opt : 追加パッケージや追加プログラム
/root : rootユーザのホームディレクトリ
/bin : 一般ユーザが実行できるシステムコマンド(ls,catなど)
/sbin : rootユーザの使用するシステムコマンド(rebootなど)
/usr : ユーザが共有するコマンド
/usr/bin : 一般ユーザとrootユーザが使用する基本コマンド
/usr/sbin : rootユーザが使用する基本コマンド
/usr/lib : プログラムの共有ライブラリ
/usr/local : 個人で作成したコマンド
/usr/src : Linuxのカーネルソースなどのソースコード
/media : DVD-ROMなどのリムーバブルメディアのマウントポイント
/mnt : 一時的にマウントするファイルシステムのマウントポイント
/proc : カーネル内部の情報にアクセスする
/tmp : 一時ファイルが配置され、すべてのユーザーが読み書きできる
/home : 各ユーザーが利用する専用のホームディレクトリ
/boot : 起動に必要なカーネルイメージ
/var : ログファイルなどの頻繁に書き込まれるファイル
/var/cache : 一時的なキャッシュファイル
/var/lock : アプリケーションを制御するためのロックファイル
/var/log : ログファイル
/var/run : システム状態を示すファイル
/var/spool : 印刷待ちのデータ、予約されたジョブ
```

◎コマンドラインの操作

- bash
- sh

- csh
- シェル
    - 環境変数
    - シェル変数
- echo $変数名　[環境変数、シェル変数を表示]
- printenv [環境変数一覧を表示]
- env　[環境変数一覧を表示。環境変数を指定してコマンドを実行]
    - env LANG=C date [英語表記で日時を表示]
    - env user=Jiro age=16 ./sample.sh (exportで環境変数を設定していてもこちらを利用する)
- set [環境変数、シェル変数一覧を表示]
- unset [環境変数、シェル変数を削除] ($はつけない)
    - unset 変数名
- export [シェル変数を環境変数として設定して別のシェルからも利用できるようにする] ($はつけない)
    - export 変数名

代表的な環境変数一覧 HOME : ホームディレクトリ LANG : 使用言語 PATH : 実行できるコマンドの検索パス PWD : カレントディレクトリ USER : ログインユーザ名

```
引用符(シングルクォーテーションとダブルクォーテーションの違い)
シングルクォーテーションの場合、中の文字はすべて文字列として解釈される
    VAR=apple
    echo '$VAR'  # $VARと表示され、変数は展開されない
ダブルクォーテーションの場合、中の変数は展開される
    echo "I like $VAR"  # I like appleと$VARが展開される
    echo "I like \$VAR"  # I like $VARと表示される
```

- man
    - -a すべてのセクションのマニュアルを表示
    - -f (whatis) 簡単な説明を表示
    - -k (apropos) 指定されたキーワードを含むコマンドの説明を表示
    - -w マニュアルの置かれたディレクトリを表示
        - manコマンドのマニュアルはセクションで分けられている
        - /usr/share/man/man5 ・・・例えば5は設定ファイル系のマニュアル
- uname
    - -a　(すべて)
    - -m　(CPUのアーキテクチャ)
    - -r　(カーネルバージョン)
- history
    - /home/user/.bash_history
    - -c
    - -d <行弁豪>
    - -a FILE_NAME
    - -r FILE_NAME

    - -w FILE_NAME
    - -s <文字列>

- 絶対パス、相対パス

- bash_profile

    - ユーザがログインした時に環境変数をexportする

フィルタを使ったテキストストリーム

- cat
    - -n 行数も表示
- cut ファイルから一部を抽出して出力するコマンド
    - cut -c 1 sample.txt (CSVの一列目を抽出)
    - cut -c 2-4 txt
    - cut -c 3- txt
    - cut -c -4 txt
    - cut -c 2,4 txt
    - cut -d , -f 1 txt (この区切り文字で各行を分割して一行目を取り出す)
    - cut -d : -f 1-4 /etc/passwd
- expand タブをスペースに変換する
    - expand -t 4 txt (タブを4つのスペースに変換する)
- unexpand スペースをタブに変換する
    - unexpand -t 8 txt (8つのスペースを1つのタブに変換する)
- less(more)
- fmt テキストを決められた桁に成型する
    - fmt -w 7 txt (1行あたり、7文字で出力)
    - fmt -u TXT (文字間あたりのスペースを1つにする)
- pr 印刷用に整形する
    - pr -h "パスワード" -l 15 +3:4 /etc/passwd | less
- head
    - head (-n) 2 /etc/passwd
- tail (-F,-f) -Fはファイルが一度なくなっても、同名のファイルが新しく作られるとそちらをフォローする
    - tail (-n) 2 /etc/passwd
    - tail -F txt
- od ファイルの形式を八進数や十六進数にダンプする (例えば画像ファイル(2進数)をキャラクターに)
    - od -t c sample.png
- sed
    - sed s/apple/banana/ txt (appleをbananaに変換する)
    - sed s/a/b/g (すべてのaをbに変換する)
    - sed -n 1,3p txt (1行目と3行目を表示)
    - sed 1,3d txt (1行目と3行目を削除して表示)
    - sed y/1p/2b txt (1を2に、pをbに変更する)
    - sed s/[0-9]/・/g txt (0-9の数字が・に変換される)
    - sed s/^[0-9]*/・ txt (10などの2桁の場合はこんな感じで)
    - sed -i s/apple/banana/g txt はファイルを直接書き換えてしまう

- tr 文字列の変換、消去
    - cat txt | tr -d ・ (あるいは　「tr -d ・ < txt」)
    - tr -s 'a' < txt (複数の同じ文字列を1文字で)
    - tr [:lower:] [:upper] < txt (小文字を大文字に変換できる)
- sort
    - 数字としてsortするなら　sort -n txt
    - 逆順なら　sort -r txt
    - スペースを無視　sort -b txt
    - cut -d : -f 1 /etc/passwd | sort
    - sort -t , -k2 txt (カンマで区切って、2つめの列をキーにsortする)
- uniq　sort済みの文字列から重複を削除
    - -d　(重複のみ)
    - -u　(非重複のみ)
- split ファイルを分割するコマンド
    - split -100 txt bk/sample.　(100行ずつ取り出して、bkフォルダにsamploe.**でファイルを作成)
    - cat bk/sample.a* > sample_splite.txt　(分割したファイルを統合)
    - split -b 20 txt　(20バイトで分割)
- join ファイル1 ファイル2　ファイル1とファイル2から重複する行を表示する
    - join -j 1 txt1 txt2　(1行目をもとにファイルを統合し、重複する行だけ出力)
- paste ファイルを行ごとに連結する
    - paste -d ":" ファイル1 ファイル2　(:で連結させる)
- nl ファイルの行頭に番号
    - nl は cat -n と同じ
- wc
    - wc -c
    - wc -w
    - wc -l
        - 3 sample.txt
    - wc -l sample.txt | cut -d ' ' -f 1
        - 3

◎ストリーム、パイプ、リダイレクト

ストリーム：データ入出力の流れのこと
標準入力（0）
標準出力（1）
標準エラー出力（2）

- パイプ
- /dev/null　スペシャルファイル。そこに書き込まれたデータはすべて削除される
- tee (ファイルと画面、両方に出力)
    - -a (追記オプション)
    - echo Hello | tee -a sample.txt
- リダイレクトなど

```
コマンド > ファイル名　#標準出力をファイルに上書きして書き込み（リダイレクト）
コマンド >> ファイル名　#標準出力をファイルに追記（リダイレクト）
```

```
コマンド < ファイル名　#ファイルを入力としてコマンドに渡す
コマンド << 文字列　#指定した文字列が入力されるまで、標準入力を行う
コマンド 2> ファイル名　#標準エラー出力をファイルに書き込む
コマンド 2>> ファイル名　#標準エラー出力をファイルに追記

2>&1　#標準エラー出力を標準出力に渡す
1>&2　#標準出力を標準エラー出力に渡す
コマンド > ファイル 2>&1　#標準出力と標準エラー出力をファイルに書き込む
コマンド >> ファイル 2>&1　#標準出力と標準エラー出力をファイルに追記する
コマンド1 | コマンド2　#コマンド1の標準出力をコマンド2の入力にする
コマンド1 | tee ファイル名 | コマンド2　#コマンド1の標準出力をコマンド2の標準入力にするとともにファイルに書き込む
```

```
$ grep hello << EOF
> goodbye
> hello
> bye
> EOF
hello
```

- xargs　標準入力から読み込んだ内容をコマンドの引数に変換して渡す
    - cat remove_file.txt | xargs rm
    - | xargs -I引数 コマンド 引数に入れる値
        - cat remove_file.txt | xargs -Ivar rm /var

```
$ ls | xargs -I{} wc -l {} 2>/dev/null
0 next
1 test.cpp.txt
2 test.hardlink.txt
2 test02.linkls /etc/ | xargs -I{} wc -l /etc/
```

◎正規表現

- 正規表現チェッカー　https://weblabo.oscasierra.net/tools/regex/

```
\d          数字1文字。[0-9]と同じ。
^           文字列の先頭。[]内で使った場合、[]内以外の文字。
$           行末
-           []内で範囲を表す。
\           直後の文字をエスケープする(メタ文字をリテラルとしてマッチさせる)。
.           改行文字を除く任意の1文字。
[]          []の任意の1文字。
\D          数字以外の1文字。[^0-9]と同じ。
\w          アルファベット又は数字(単語)。[a-zA-Z_0-9]と同じ。
\W          アルファベットと数字以外の1文字。[^a-zA-Z_0-9]と同じ。
\s          空白文字、タブ文字又は改行文字。[\n\r\f\t]
```

```
\S          空白文字、タブ文字及び改行文字以外。[^\n\r\f\t]
pattern1 | pattern2      論理和(OR)。pattern1又はpattern2のいずれかにマッチ。
(string)    文字列をグループ化

?           直前の要素が0個か1個
*           直前の要素が0個以上
+           直前の要素が1個以上
{ m , }     直前の要素がm個以上
{, n }      直前の要素がn個以下
{m,n}       直前の要素がm個以上n個以下
```

```
これ便利
(.*?)       特定文章の特定部分を抽出したい
```

- grep　文字列と正規表現を用いて検索するコマンド
  - -v　特定の文字列を含まない行を抽出
  - -E　正規表現を用いて文字を抽出

```
[root@localhost work]# cat grep
is test file
Hi
iiiiiiiiddddddd
[root@localhost work]# grep -E 'is(.*?)file' grep
is test file
```

  - -F　正規表現を利用しない
    - grep \[abcd\] txt
    - grep -F [abcd] txt
  - -n　行数をふやして抽出
  - -i　文字列の大文字と小文字を区別しない
- egrep　正規表現を用いて文字列の検索をする（grep -Eと同じ）
  - -n　行数も表示
  - -v　検索文字列を含まない行を抽出
  - -i　文字列の大文字と小文字を区別しない
- fgrep　正規表現を利用しない（grep -Fと同じ）
  - -n　行数も表示
  - -v　検索文字列を含まない行を抽出
  - -i　文字列の大文字と小文字を区別しない

◎エディタを使ったファイル編集

- vi
- vim
  - vimdiff text1 text2　（Ctrl+wで異なるファイルに移動）
- nano　Ubuntsで標準搭載

- EDITOR　環境変数

◎仮想マシン・コンテナ

Docker Client　　コマンド
Docker Server　　コンテナの作成、削除、起動
Docker images　　コンテナのソース
Docker Hub　　イメージを共有するリポジトリ
Docker Compose　　複数コンテナの管理
Docker Machine　　仮想環境上のDockerをインストールするツール

- Dockerイメージ
  - イメージは修正して保存することはできず、イメージを修正して新たなイメージとして保存。
  - イメージはimage layerという階層構造になっており、イメージの内容を修正したら階層のトップに加える
    - Linuxカーネル -> Ubuntuイメージ -> VIM追加 -> Apache追加
- Dockerコンテナ
  - イメージを実体化したもの
- カーネル
  - OSの中核
- ハイパーバイザー
  - 仮想化技術を用いてOS上の仮想マシンを作成するためのプログラム
- ネイティブハイパーバイザー
  - ハードウェア上にハイパーバイザーが直接動作し、様々なOSがハイパーバイザー上に動作する
    - vSphere / Hyper-V / KVM など
- ホスト型仮想化
  - ホストOSの上に新たなOSを作る
    - Vmware / VirtualBox
- コンテナ型仮想化
  - ゲストOSは必要なくホストOSのコア部分（カーネル）を共有して使用する
  - 原則、LinuxOS上でWindowsコンテナを使用することはできず、WindowsOS上でLinuxコンテナを使用することはできない
- Docker
  - コンテナ管理ソフトウェア
  - GO言語

◎Docker install

```
yum install -y yum-utils
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
yum-config-manager --enable docker-ce-nightly
yum install -y docker-ce docker-ce-cli containerd.io

systemctl start docker
systemctl status docker
```

```
yum-config-manager --disable docker-ce-nightlyyum-config-manager --eable docker-ce-nightly
```

- docker pull
- docker rmi
- docker create
- docker start (-i)
- docker run (pull - create -start まとめて)
    - docker run イメージ名
    - docker run -it: -i
- exit
    - コンテナを停止して抜け出す
- ctrl+P -> ctrl+Q
    - コンテナを停止せずに抜け出す
- docker run -d
- docker tag イメージ名 新しいイメージ名
    - イメージを別名であたら悪しく作成する
- docker rmi (-f) イメージ名(イメージID)
    - ローカル上に存在するDockerイメージを削除する
- docker commit コンテナID イメージ名(:タグ)
    - コンテナから新しいイメージを作成する
- docker images
    - ダウンロードしたDockerイメージを確認する
- docker inspect
    - イメージの詳細情報を表示する
- docker ps
    - 動作中のコンテナを表示する
- docker ps -a
    - 終了したコンテナも含めて、コンテナをすべて表示する
- docker rm コンテナID

    - 指定したコンテナIDのコンテナを削除する
- docker start コンテナID or コンテナ名
    - コンテナの開始
- docker stop コンテナID or コンテナ名
    - 動作中のコンテナを終了させる（SIGTERM）
- docker kill コンテナID or コンテナ名
    - 動作中のコンテナを終了させる（SIGKILL）
- docker build -t タグ名 .
    - 現在のフォルダ上のDockerfileからイメージをビルドする
    - -f ファイル名　Dockerfile以外のファイルを指定する
- docker system prune
    - コンテナ全削除
- docker image prune
    - イメージ全削除

コマンド実行

```
docker pull centos:6

docker images
    REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
    centos       6         5bf9684f4720   18 months ago   194MB

docker create 5bf9684f4720 echo 'Hello'
    85bfa0293771e9b1b80831f671b03a45cb644e9085efcd89cf1a43f4bc340620

docker ps -a
    CONTAINER ID   IMAGE          COMMAND        CREATED          STATUS    PORTS
NAMES
    85bfa0293771   5bf9684f4720   "echo Hello"   45 seconds ago   Created
loving_mayer

docker start 85bfa0293771
    85bfa0293771

docker ps -a
    CONTAINER ID   IMAGE          COMMAND        CREATED         STATUS
PORTS     NAMES
    85bfa0293771   5bf9684f4720   "echo Hello"   2 minutes ago   Exited (0) 19
seconds ago             loving_mayer
```

```
docker start -a 85bfa0293771
    Hello


docker run hello-world
docker images
    REPOSITORY    TAG       IMAGE ID       CREATED                  SIZE
    hello-world   latest    9c7a54a9a43c   Less than a second ago   13.3kB
    centos        6         5bf9684f4720   18 months ago            194MB

docker ps -a
    CONTAINER ID   IMAGE          COMMAND        CREATED              STATUS
PORTS     NAMES
    db0275b3475a   hello-world    "/hello"       About a minute ago   Exited (0)
About a minute ago             flamboyant_wozniak
    85bfa0293771   5bf9684f4720   "echo Hello"   6 minutes ago        Exited (0) 2
minutes ago                  loving_mayer

docker start -a db0275b3475a
docker ps -a
    CONTAINER ID   IMAGE          COMMAND        CREATED         STATUS
PORTS     NAMES
    db0275b3475a   hello-world    "/hello"       2 minutes ago   Exited (0) 12
seconds ago             flamboyant_wozniak
    85bfa0293771   5bf9684f4720   "echo Hello"   8 minutes ago   Exited (0) 3
minutes ago              loving_mayer
docker rm db0275b3475a
docker rmi 9c7a54a9a43c
```

◎コンテナを起動したままにする

```
docker run -it centos
[root@28c6446adc1e /]#
exit

docker start -i 28c644

Ctrl+p -> Ctrl+Q
docker exec -it 28c644 bash
exit   (execの場合は、exitでも落ちない)

docker stop 28c
    28c

docker run -d cetos sleep 100   (バックグラウンドで実行)
docker kill 上のdockerID   (強制終了)

 docker system prune
```

- docker commitについて

1. ubuntuのコンテナを作成
2. コンテナにa.txtを作成
3. そのコンテナからイメージを作成して
   docker commit
4. イメージからコンテナを作成するとa.txt

◎docker file

```
vim Dockerfile
    FROM ubuntu:bionic

    RUN apt update -y && apt install -y python3-pip python-dev

docker build -t my_flask .

docker images

docker run -it my_flask

# 以下を加える
# Flaskのインストール
COPY ./\requirements.txt requirements.txt
RUN pip3 install -r requirements.txt

docker build -t my_flask .
```

```
# app.py
from flask import Flask

app = Flask(__name__)

@app.route(' /hello')
def hello_world():
    return 'Hello World'

if __name__ == '__main__' :
    app.run(host='0000')

http://localhost/hello
```

◎virshについて

```
virt-manager  GUIで利用できる仮想マシン管理用ソフト
virt-install  CUIでの仮想マシン作成用のコマンド
libvirt  仮想化機構（KVMなど）と連携するライブラリ
```

```
virsh               仮想マシンとハイパーバーザーのリソースを管理するコマンド
virsh list          ドメインの一覧を表示
virsh console VM    virshコマンドを用いてVMに接続する
virsh shutdown VM   仮想マシンを停止する
virsh destroy        仮想マシンを削除する
virsh dominfo VM    仮想マシンのドメインに関する基本的な情報を表示する
virsh domid VM      仮想マシンのドメインIDを表示する
virsh domstate VM   仮想マシンの実行状態（シャットオフ、実行中などの状態）
```

◎ブートプロセスとsystemd

- ここは後ほど改めて

```
systemctl list-unit-files --type=service
```

```
wall  #全ユーザのにメッセージを送る
```

- ssh

```
~/.ssh/known_hosts     一度もssh接続をしていないサーバにssh接続した際に接続したサーバの情報が保存されるファイル
~/.ssh/authorized_keys  公開鍵認証でクライアントの公開鍵を登録するファイル
~/.ssh/id_rsa           公開鍵認証で利用する秘密鍵
~/.ssh/id_rsa.pub        公開鍵認証で利用する公開鍵
```

```
#接続先
/etc/ssh/sshd_config
systemctl restart sshd

#接続元
ssh-keygen
ls ~/.ssh/
cat ~/.ssh/id_rsa/pub

ssh-copy-id -i ~/.ssh/id_rsa.pub root@接続先

#接続先
cat ~/.ssh/authorized_keys
```

◎プロセスの生成、監視、終了

```
コマンド &     コマンドをバックグラウンドプロセスとして実行
jobs           ログインユーザの実行中と停止中のジョブを表示するコマンド
    ctrl + z       実行中のコマンドを停止する

bg             一時停止中のジョブをバックグラウンドで再開する
fg             バックグラウンド実行中のジョブをフォアグラウンドで実行

kill %〇       〇番目のジョブを削除する
kill -19 %〇   〇番目のジョブを停止する

top  実行中のプロセスのCPU使用率などを継続で表示する

ps  実行中のプロセスを表示する

pstree  親プロセスとその親プロセスから派生した子プロセスの階層構造を表示する

uptime  システムの稼働時間を表示する

pgrep  プロセス名を指定してプロセスIDを検索する

kill  プロセスにシグナルを送信する
    kill -1 PID     再起動
    kill -2 PID     割り込み（Ctrl+C）
    kill -9 PID     強制終了
    kill -15 PID     終了
    kill -19         一時停止

pkill  プロセス名を指定してプロセスにシグナルを送信する
killall  プロセスの停止などシグナルを送れる。killと似ているがコマンド名を指定できる

nohup     ユーザがログアウトした後も、指定したコマンドの標準出力をバックグラウンドで継続して処理させ、
$(HOME)/nohup.outファイルに出力する

tmux     ターミナル上に複数のターミナルを立ち上げる
    tmux ls
    tmux new-session -s **
    tmux kill-session -t **
    tmux attach -t **
    tmux rename-session -t **
```

◎デスクトップ環境の利用

- ここは後ほど改めて

◎RPMパッケージ管理、yum

- ここは後ほど改めて
- RPM
- rpm　（依存関係を解決することはできない。強制的に単体でインストールすることはできる）

- rpm -q　（パッケージ情報検索）
- yum
    - /etc/yum.conf
    - /etc/yum.repo.d/

◎Debianパッケージ管理、apt

- ここは後ほど改めて
- dpkg　（依存関係を解決することはできない）
- apt　（apt-getは古い）
    - /etc/apt/sources.list
- apt update

◎ハードウェアの基礎知識と設定

- ここは後ほど改めて
  94-99

◎ハードディスクのレイアウトとパーテーション、ファイルシステムの作成と管理、マウント

- パーテーション
    - 基本パーテーション　（大元。最大4個作成することができる。/dev/ada1 - sda4）
    - 拡張パーテーション　（基本パーテーションのうち1つを拡張パーテーションにできる）
        - 論理パーテーション　（sda5以降作成することができる）
- EFIシステムパーテーション
- パーテーション管理方式
    - MBR (Master Boot Record)
        - 基本パーテーションは4個まで
        - 扱えるディスクドライブは2TBまで
    - GPT (GUID Partition Table)
        - パーテーション区別はなくなる（基本や拡張などはない）
        - 128個までパーテーションを作成できる
        - 容量は8zbまで
- LVM (Logical Volume Manager)
    - 物理ボリューム (sda sdb)
    - ボリュームグループ (物理ボリュームを束ねる)
    - 論理ボリューム　（ボリュームグループを切り出す）
- 以下のことが可能

    - 動的な論理ボリュームの拡張/縮小/追加/削除
    - スナップショットの作成
    - ディスク間をまたがったボリュームの作成
    - ボリュームグループのサイズ変更

- fdisk　MBRのパーテーション管理をするコマンド

- parted　MBR,GPTのパーテーション管理をするコマンド

ハードディスクのレイアウトとパーテーション

- パーテーション分割

    - ディスクの領域に分割すること。データを整理し安全に利用したりパフォーマンスを口上させるために用いる

- (root)ファイルシステム

    - ルートディレクトリと呼ばれる最上位のディレクトリ。すべてのディレクトリの親ディレクトリをたどるとルートディレクトリに達する

- スワップ領域

    - 物理メモリの秋容量が少なくなった時にディスクの一部を仮想的にメモリとして使う領域（物理メモリの1~2倍の容量を確保する）
    - ルートとswap領域をLinuxをインストールする際に最低限必要なパーテーション

- /boot

    - Linux起動時に必要なファイルを配置するファイルシステム。古いシステムだと容量が足りないと起動できないこともありパーテーションを分ける

- /home

    - ユーザのホームディレクトリが置かれる。ユーザごとにディレクトリが作成されユーザ個人のファイルを配置する。ユーザの利用するファイルの増大があり、パーテーションを分ける

- /var

    - システムのログなどに動的なファイルがこのディレクトリに作成される。Webサーバなど大量にログが出力されるシステムではパーテーションをわける

- /tmp

    - 誰も書き込み可能なディレクトリで一時的に使用するファイルを配置する。一時ファイルの増加することが考えられるため、パーテーション分割する

- /usr

    - OSのプログラムやライブラリがこのディレクトリにインストールされる。性能の観点からパーテーション分割する

- ファイルシステム：ファイルとしてディスク上のデータを扱う仕組み

- ファイルシステムがないとき：1882128ブロックから1882150までのデータを読み込むといった処理をする
- ファイルシステムがあるとき：/dataディレクトリ内のsample.txtを読み込むといった処理をする
    - ext2
        - Linuxの標準ファイルシステム
    - ext3
        - ext2にジャーナリング機能(ファイルシステムへの操作をログに記録する仕組み)を備えたもの
    - ext4
        - ext3を昨日拡張したファイルシステム
    - XFS
        - RHEL/CentOSの標準ファイルシステム
    - JFS
        - IBM社が開発したジャーナルファイルシステム
    - Btrfs
        - 最近登場したファイルシステムで現在開発中のもの
    - iso9660
        - CD-ROMのファイルシステム
    - UDF
        - DVD-ROMのファイルシステム
    - Msdos
        - MS-DOSのファイルシステム
    - VFAT
        - SDカード、USBメモリなどのフラッシュメモリで利用されているファイルシステム
    - tmpfs
        - メモリ上に作成される仮想的なファイルシステム
- mkfs　（論理 or 物理）パーテーション上にファイルシステムを作成するコマンド
    - -t　　作成するファイルシステムの種類をして（ext2(デフォルト)/ext3/ext4など）
    - -c　　作成する前に不良プログラムの有無の検査を行う
- mke2fs
    - -t ファイルシステム
    - -j
    - -c
- Btrfs
- mkfs.btrfs　or　mkfs -t btrfs
- f -T
- fsck -N パーテーション名
- XFS　RHEL/CentOSでの標準のファイルシステム。最大ファイルサイズは8Eバイトで、大容量にも対応している

    - mkfs.xfs　　XFSファイルシステムを作成する
    - xfs_info　　XFSファイルシステムの情報を表示する
    - xfs_db　　XFSファイルシステムをデバッグする
    - xfs_check　　XFSファイルシステムをチェックする
    - xfs_admin　　XFSファイルシステムのパラメータを変更する
    - xfs_fsr　　XFSファイルシステムのデフラグをする
    - xfs_repair　XFSファイルシステムを修復する

- スワップ領域　システムの物理メモリが足りなくなった場合に、メモリの一部を書き出す領域
- mkswap　　指定したパーテーションにスワップ領域を作成するコマンド
- mkswap /dev/sda1　　/dev/sda1にスワップ領域を作成する
- swapon　　スワップ領域を有効にする
    - -s　スワップ領域の一覧を確認する
- cat /proc/swaps　スワップ領域の一覧を確認する
- swapoff　　スワップ領域を無効にする
- /etc/fstab　　マウントする際に参照されるマウントの初期値を設定する設定ファイル

```
cat /etc/fstab
＝＝＝＝1＝＝＝＝＝＝＝2＝＝＝3＝＝＝4＝＝＝＝＝5＝＝＝6＝＝＝
/dev/mapper/cl-root    /    xfs    defaults    0    0
１：デバイスファイル名 or UUID（デバイスを識別するためのユニークのID）
２：ファイルシステムのマウント先のディレクトリ（マウントポイント）
３：ファイルシステムの種類
４：マウントオプション
５：dumpフラグ（1はdumpコマンドのバックアップ対象になる）
６：ブート時にfsckがチェックする順番

・オプション
    async　　非同期入出力を設定する
    auto　　-aオプションでmountを実行したときにマウント
    noauto　　-aオプションでmountを実行したときにマウントしない
    defaults　デフォルトのオプションを設定
    exec　　バイナリの実行を設定
    noexec　　バイナリの実行を許可しない
    ro　　読み取り専用でマウント
    rw　　読み書きを許可してマウント
    unhide　　隠しファイルも表示
    suid　　SUIDとSGIDを有効にする
    user　　一般ユーザでもマウント可能にする
    users　　マウントユーザ以外にも案マウントできる
    nouser　　一般ユーザのマウントを許可しない
```

```
mount USB,DVDなどのフォーマット済みの領域をLinux内のディレクトリと結び付けてアクセスできるようにする
    -a
    -o
    -t

async  入出力を非同期で実行
ro     読み取り専用モードでマウント
rw     読み書き可能モードでマウント
user   一般ユーザにマウントを許可
```