基本コマンド

Linux (Unix)の基本コマンドを説明します。Linux は web server などで最も多く使われるオープンソースの OS で、商用ソフト Unix と互換性のあるコマンドが使用できます。Macintosh は Unix をベースにしているので、Mac のコマンド操作は Linux/Unix にも応用することができます。

本資料においてコマンドの前の ＄ マークはプロンプト（コマンド入力が待機中であることを意味する記号）を示しますので、入力は不要です

誤った操作などによって、入力が受け付けなくなってしまった（プロンプトが表示されなくなった）場合には、
ctrl-C を押すことで、処理をキャンセルし入力待機状態に戻ることができる場合があります。
q を押すことで処理を中断させられる場合もあります。

また、タブキーを素早く２度押すと、利用可能なコマンドやファイル名が補完されます（候補が多すぎて表示しきれない場合 q を押すと途中でとめられます）
矢印キーの上下で過去に入力したコマンドの履歴が表示できます。
これらをなるべく活用することで手入力による打ち間違いを避けるのがコツです。

---

1. 現在の状況を把握する（pwd と ls）

pwd: 現在自分がいるディレクトリ（フォルダ）を表示します。(print working directory)
$ pwd
ログイン直後には各ユーザーのホームディレクトリ /Users/userXXX/ にいるはずです。

ls: 現在のディレクトリ内にあるファイルおよびディレクトリの一覧（list）を表示します。
$ ls

不可視ファイル（. で始まるもの）を含めて表示させます。
$ ls -a
このコマンドによって表示されたディレクトリのうち、
. は現在のディレクトリ、.. は一つ上の階層のディレクトリ（親ディレクトリ）を示します。
また、-a はコマンドラインオプションのひとつです。
コマンドの多くはコマンドラインオプションを使うことで、既定の動作を変更させることができます。

ファイルの更新日時やファイルサイズも含めて表示させます。
$ ls -a -l
上記は ls -al としても同様の結果が得られます。
また、多くのシステムでは ll が "ls -l" のショートカットとして利用できます。

$ ls --help
多くのコマンドでは -h -help --help などのオプションをつけることでヘルプを表示できます。

---

2. Linux のディレクトリの基本構造

/（ルートディレクトリ）
　/bin（おもにシステムの基本動作に最低限必要な実行ファイルを格納する）
　/Applications（各種アプリケーションと、それに付随するファイルを格納する）
　/Users 各ユーザーごとのホームディレクトリが格納されている
　　/Users/userXXX/　userXXX のホームディレクトリ
　　/Users /userYYY/　userYYY のホームディレクトリ
　　/Users /userZZZ/　userZZZ のホームディレクトリ

Mac/UNIX/Linux ではファイルやディレクトリごとに所有者や書き込み・読み込み権限（パーミッション）が設定されており、通常は他のユーザーからはファイルが見られないようになっています。
同じグループに属するユーザーであれば一般的には閲覧可能です。
自分が所有するファイルであれば権限を変更することも可能です（chmod コマンド）

上記は@tomoya 氏の Blog を参考にしました
https://qiita.com/tomoya1993/items/c0f1592faa12478ffefa

---

3. ディレクトリの基本操作

ディレクトリの新規作成
mkdir コマンド（make directory)を使い “test"という名称のディレクトリを作ります。
$ mkdir test

ディレクトリの移動
cd コマンド（change directory）を使い作成したディレクトリに移動します
$ cd test

元の（一つ上の）ディレクトリに戻ります。
$ cd ..
.. は親ディレクトリを示します

Applications ディレクトリに移動します。
$ cd /Applications
/ （ルートディレクトリ）を起点にした場所の示し方を「絶対パス」と呼びます。

元に戻ります。
$ cd

別のディレクトリを作成します。
$ mkdir test2
$ cd test2

test2 から test ディレクトリに移動します。
$ cd ../test
現在いる場所を起点にした場所の示し方を「相対パス」と呼びます。

ディレクトリを削除します。
$ rmdir test
このコマンドは空のディレクトリに対してしか使用できません。
ディレクトリの中身も含めて削除する場合には代わりに
$ rm -r test2
コマンドを使用してください

ホームディレクトリに戻ります
$ cd
または
$ cd ~
~ はホームディレクトリを示す記号です。

4. touch コマンドと rm コマンド(ファイル作成と消去)

touch コマンドを使い空のテキストファイル（a.txt）を作成します
$ touch a.txt
なおテキストファイルであることが目で見てわかりやすいように拡張子 .txt をつけていますが、
Windows とは異なり、Linux では拡張子によってファイルの種類を認識することはありません。
したがって、拡張子は必須ではありません。（touch a で a という名称の空のファイルが作られます）

ll コマンド（ls -l）でファイルが作成できていることを確認します。
$ ll

なお、touch コマンドは本来はファイルのタイムスタンプを更新するためのコマンドです。
動作確認のため、再び touch コマンドを用いた後、タイムスタンプを見てみます。
$ touch a.txt
$ ll

ファイルの消去
$ rm a.txt

rm a.txt b.txt c.txt というように複数のファイルを指定して削除することも可能です。
また、ワイルドカード（後述）を利用してパターンに合うファイルを一括削除することもできます。
-r オプション（recursive）をつけて実行することでフォルダ内のファイルすべてを再帰的にすべて消去することができます。
例）rm -r test　← test ディレクトリを中身を含めて消去します。

---

5. echo コマンド、cat コマンド、リダイレクト

echo コマンド：文字列を画面に出力します
$ echo OK

> を使うと画面の出力をテキストファイルに書き込むことができます（リダイレクト）。
$ echo OK > a.txt
a.txt がすでに存在していた場合、上書きされますので気をつけてください。

>> で追加書き込みできます
$ echo OK2 >> a.txt

他のコマンドの出力結果もファイルに書き込めます。
$ ls -l > b.txt

cat コマンド：ファイルの中身を画面に出力します
$ cat a.txt
cat コマンドは本来は複数のファイルを連結（concatenate）するコマンドです。
$ cat a.txt b.txt > c.txt
で a.txt と b.txt を連結し、その結果を c.txt に書き込みます。

---

6. ワイルドカード

.txt で終わるファイルの一覧を表示
$ ls *.txt
*は 0 文字以上の任意の文字列を示します

$ ls ?.txt
?は任意の 1 文字を示します
$ ls ??.txt
と打った場合には ab.txt や XY.txt にマッチしますが、a.txt にはマッチしません

これまでに作成した .txt ファイルを削除します
$ rm *.txt
(単に rm * と打つと全てのファイルが消えてしまいますのでご注意ください。
また、-r オプションをつけて rm -r * を実行すると、カレントディレクトリ以下にある
ファイルがすべて消えます。linux では一度削除したファイルを
復活させることはできませんので、使用する場合には注意してください)

---

7. cp コマンド（ファイルのコピー）と mv コマンド（ファイルの移動）

これまでの復習を兼ねて練習用のファイルとディレクトリを作ります
$ cd # ホームディレクトリに移動
$ mkdir dirA
$ mkdir dirB
$ echo "File A" > a.txt

$ echo "File B" > b.txt

a.txt を A ディレクトリに移動します。
$ mv a.txt dirA/
(最後の/はなくても構いません)

b.txt を B ディレクトリにコピーします。
$ cp b.txt dirB/
(最後の/はなくても構いません)

ファイルを別名で複製することもできます。
b.txt を b2.txt として複製します
$ cp b.txt b2.txt
(最後の/はなくても構いません)

mv コマンドはファイル名を変更するときにも使用します。
b.txt を c.txt に名前を変えます
$ mv b.txt c.txt

dirB に移動します
$ cd dirB
一つ上のディレクトリにある c.txt をカレントディレクトリに移動します。
$ mv ../c.txt ./
結果的に b.txt と b2.txt と c.txt の 2 つがカレントディレクトリに存在するはずです。

b.txt と c.txt の両方を dirA にコピーします。
$ cp b.txt c.txt ../dirA/
このように複数のファイルを指定してまとめてコピーすることもできます。
mv コマンドも同様です。

一つ上の階層に戻ります
$ cd ..
dirB を dirA の中にコピーします。
ディレクトリをコピーするときは -r オプションを加えます
$ cp -r dirB dirA

最後に余分なファイルを削除します
$ rm -r b2.txt dirA dirB

8. less コマンド と more コマンド

cat コマンドはファイルの中身を一気に表示させるのに対し、
大きなファイルを部分ごとに表示させるのが less や more コマンドです。

less と more は多少の違いはありますが、less は more の拡張版ですので、
less を使うことの方が多いです。

$ cd bacteria_analysis
$ less read.fastq
で起動します。
矢印キーの上下左右でスクロール。
f と b で 1 画面ずつ上下にスクロールします。

/ で前方（現在の位置からファイル末端に向けて）への検索
? で後方（現在の位置からファイル末端に向けて）への検索
繰り返し検索する場合には n または N で次候補または前候補を表示します。

q で less を終了します。

h でヘルプが表示できます。

参考）
less コマンドは、デフォルトでは画面の右端で自動的に折り返されて表示されます。
この設定を無視して画面 1 行にファイルの内容を 1 行ずつ表示させる場合には
-S オプションを使用します。
$ less -S read.fastq

---

8. head コマンドと tail コマンド, wc

それぞれファイルの先頭および末端部分を表示します。

$ head -100 read.fastq > head100.txt
で先頭から 100 行を抽出し、ファイルに書き込みます。
$ tail -100 read.fastq > tail100.txt

で末端から 100 行を抽出し、ファイルに書き込みます。

-100 の部分を省くと 10 行表示されます。

wc で行数を数えて見ます
$ wc read.fastq
表示される数値は順に、行数、単語数、文字数を表します。
文字数には改行コードも含まれます。

-l をつけると行数のみ表示します
$ wc -l read.fastq

---

9. パイプ

“ | “ を使うとコマンドの出力を他のコマンドに連結することができます（読み方はパイプ、パイプライン、バーチカルバーなど）

例）ファイルの先頭 1000 行を抽出した後、その末尾 100 行を表示させます。
$ head -1000 read.fastq | tail -100
結果的に 901 行目から 1000 行目が抽出されて表示されます。

複数のパイプを連結することも可能です。
例）ファイルの先頭 1000 行を抽出した後、その末尾 100 行を、less を用いて表示させます。
$ head -1000 read.fastq | tail -100 | less

---

10. ファイル内の検索 （grep）

ファイルの中から文字列 AAAAAAAAA を含んだ行を表示させます。
$ grep AAAAAAAAA read.fastq

パイプと組み合わせて使用することもできます
$ head -1000 read.fastq | grep AAAAAAAAA

---

11. プログラムの実行

サンプルに含まれる”hello”という名称のプログラムを実行します。

実行権限の付与
$ chmod a+x hello

実行方法は３通りあります。
その１（相対パスで指定）
$ ./hello
カレントディレクトリにあるプログラムを実行するには明示的に./をつける必要があります。

その２（絶対パスで指定）
/Users/ngs/bacteria_analysis/hello

その３　パス（PATH）を通す
PATH とは、プログラムの探索パスの意味です。
echo $PATH とうつと現在の設定内容が見られます。PATH に含まれた場所にあるプログラムはコマンド名をうつだけでどこからでも実行できるようになります。

$ export PATH=/Users/ngs/bacteria_analysis:$PATH
と実行すると、PATH に “/Users/ngs/bacteria_analysis”を追加をすることができます。
このように環境変数 PATH を設定し、コマンドを実行できるようにすることを
「パスを通す」といいます。

参考）
新しいターミナルにログインをすると export コマンドで設定した環境変数の内容は失われてしまいますので、ログインごとに再設定する必要があります。
頻繁に使うツールについては、export コマンドを .bash_profile ファイル（または .bashrc ファイル）の中に記述することで自動で設定するようにできます。
これらのファイルはログイン時に自動で読み込まれ、設定が反映されます。

解説原稿へのリンク

https://docs.google.com/document/d/1NTAw5wFbJKCp3m1hiMslUiZMWhcxU0GaPek2_CLJkVQ/edit#heading=h.szg15sx2ouxk

---

その他

curl, wget インターネット上からファイルを取得する

URL を指定してインターネット上のファイルを取得します。
例）DDBJ の FTP サーバーからシークエンスデータを取得します。
$ curl -O
ftp://ftp.ddbj.nig.ac.jp//ddbj_database/dra/fastq/SRA117/SRA117449/SRX456287/SRR1151187_1.fastq.bz2

$ wget
ftp://ftp.ddbj.nig.ac.jp//ddbj_database/dra/fastq/SRA117/SRA117449/SRX456287/SRR1151187_1.fastq.bz2

ファイルの展開および圧縮

zip 形式 → 展開 unzip、圧縮 zip
gz 形式 → 展開 gunzip、圧縮 gzip
bz2 形式 → 展開 bzip2、圧縮 bzip2
tar or tar.gz 形式 → 展開および圧縮 tar コマンド

例 tar.gz ファイルの展開
$ tar xvfz archive_file.tar.gz
x は展開（extract）, v は展開の過程を画面に表示（verbose）を表します。
z は対象が gz 形式で圧縮されている場合につけますが、最近はつけなくても展開できます。
f は展開する対象がファイルであることを示します。