技術研修 第11回
UNIXコマンド編

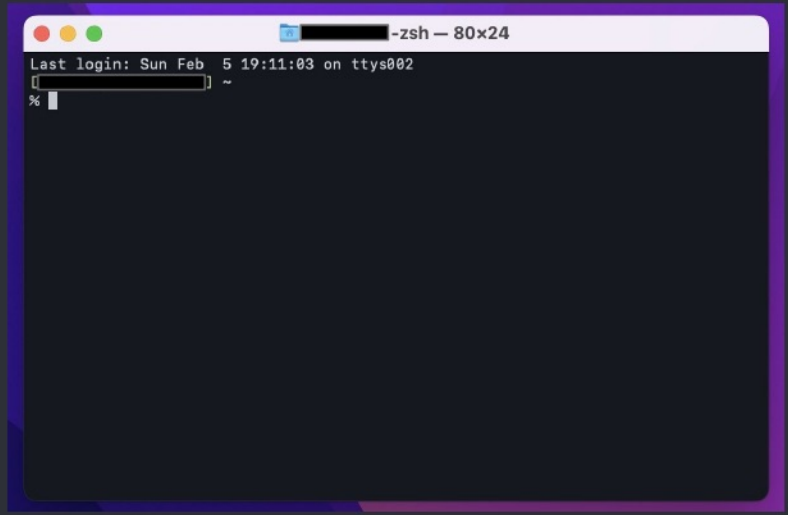
目次

- UNIXコマンドとは
  - いろいろコマンド紹介
  - 演習① 調べながらコマンドを使ってみる
- プロセスとは
  - プロセスを見たい→ psコマンド
  - プロセスを終了させたい→ killコマンド
- リダイレクト、パイプ
  - 演習② ワンライナーで書いてみる
- シェルスクリプト

UNIXコマンドとは

UNIXコマンドとは

- macOSやLinuxなどで利用できる
- CLIを通してPCに送る命令
- ターミナルから打ち込んでいるアレです！

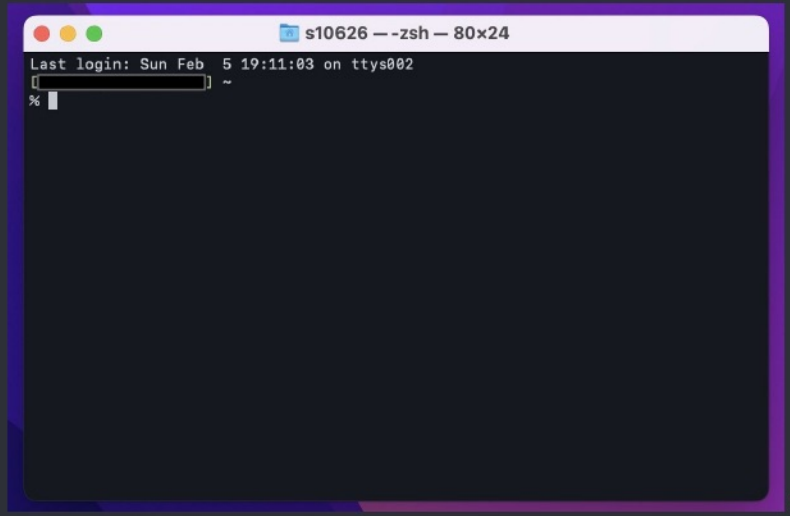


UNIXコマンドとは

- macOSやLinuxなどで利用できる
- CLIを通してPCに送る命令
- ターミナルから打ち込んでいるアレです！

ちなみに、みなさん普段どれくらい

UNIXコマンド使ってますか？



いろいろコマンド紹介

基本的なもの

ls, cd, mv, cp, rm, mkdir, ...

ちょっとした作業で使うもの

echo, cat, head, tail, less, touch, ...

いろいろあると思いますが、

今回は「うまく使えると便利なもの」に注目して紹介します！

いろいろコマンド紹介 ～よく使う系～

grep

対象のファイルから指定の文字列を含む行を見つける

```
grep [options] [pattern] [file ... ]
```

ログから特定の出力を見つけたい時など、結構使います！

後で話しますが、正規表現も使えるので便利

いろいろコマンド紹介 ～よく使う系～

wc

ファイルの行数、文字数などを調べる

```
wc [options: -clmw] [file ... ]
```

-c: バイト数

-l: 行数　←これを使うことが多い。csvファイルに使えばレコード数がわかる

-m: 文字数（マルチバイト文字を1としてカウント）

-w: 単語数

いろいろコマンド紹介 ～よく使う系～

sed

文字列の置換など、テキスト編集処理ができる

```
sed [options] [command] [file ... ]
```

-i: ファイルに上書きする

例：test.csv内の「hoge」を「fuga」に置換する

```
sed -i -e “s/hoge/fuga/g” text.csv
```

いろいろコマンド紹介 ～よく使う系～

tr

文字列の置換や削除ができる

置換

```
tr [文字1] [文字2]
```

削除

```
tr -d [文字]
```

いろいろコマンド紹介 ～データファイルいじる系～

q

csv, tsvファイルをテーブルとして扱い、select文などクエリを実行できる

yq

yamlファイルからのデータ取得や編集ができる

jq

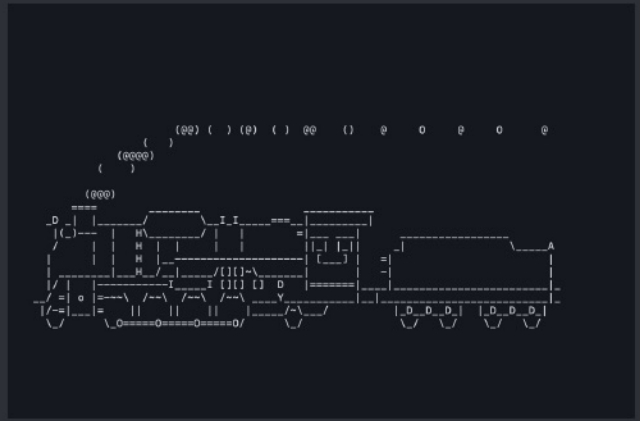
jsonファイルからのデータ取得や編集ができる

おまけ

slコマンド

機関車がターミナルを走り抜ける

こいつが表示されたらいったん休みましょう



プロセスとは

プロセスとは

- プログラムやコマンドを実行することで生まれる処理
- PC上では様々なプロセスが並行して走っており、
  これをOSが管理している

- 例えば、lsコマンドを実行するとlsコマンドのプロセスが生まれる。
  処理が終わるとプロセスも終了する

プロセスを見たい→ psコマンド

```
[                    ] ~
% ps ax
  PID   TT  STAT      TIME COMMAND
    1   ??  Ss    75:57.13 /sbin/launchd
   77   ??  Ss    29:16.61 /usr/libexec/logd
   78   ??  Ss     3:57.78 /usr/libexec/UserEventAgent (System)
```

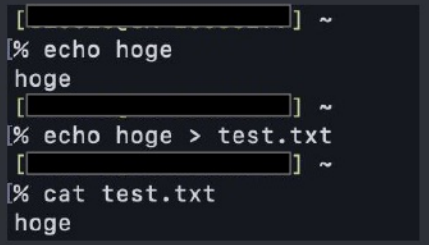
- プロセスを終了させたい→ killコマンド

```
[                          ] ~
% sleep 3600 &
[1] 43992
[                          ] ~
% ps
  PID TTY           TIME CMD
30243 ttys000    0:47.74 -zsh
62918 ttys001    0:00.26 -zsh
43786 ttys002    0:00.14 -zsh
43992 ttys002    0:00.00 sleep 3600
[                          ] ~
% kill 43992
[1]  + terminated  sleep 3600
[                          ] ~
% ps
  PID TTY           TIME CMD
30243 ttys000    0:47.74 -zsh
62918 ttys001    0:00.26 -zsh
43786 ttys002    0:00.14 -zsh
```

リダイレクト、パイプ

リダイレクト

- コマンドの入出力先をファイルに変える
- “<” を使って入力ファイル名を指定
- “>” を使って出力ファイル名を指定
- ちなみに
  - “>” … ファイル上書き
  - “>>” … ファイルに追記



パイプ

- コマンドの出力を別コマンドの入力としてつなぐ
- “|” を使ってコマンドとコマンドをつなぐ

```
% ls master/*.csv | wc -l
     670
```

いろいろコマンド紹介 ～パイプと組み合わせる系～

xargs

渡されたリストの要素1つ1つに対して指定のコマンドを実行

例

```
ls | xargs rm -rf
```

私がよくやるやつ

```
git tag | xargs git tag -d
```

ローカルに溜まってるgitタグを全て消す

シェルスクリプト

シェルスクリプト

UNIXコマンドだけで完結させたいけど、

処理が複雑だし変数も多い…

→シェルスクリプトを書くのがおすすめです！

- if文やfor文、関数を使って処理を構造化できる
- 書き方はちょっとクセあり

```
#!/bin/bash

for i in `seq 1 5`; do
    if [ $i -ge 3 ]; then
        echo $i
    fi
done
```

シェル変数

- イコールで値を代入
- 「$」を変数名の前につけて値を参照

```
[                              ] ~/test
% HOGE=test_string
[                              ] ~/test
% echo $HOGE
test_string
[                              ] ~/test
% HOGE=`ls`
[                              ] ~/test
% echo $HOGE
test01.txt
test02.txt
test03.txt
```

- `ls` のように「`」で囲うとコマンドの結果が取れるので
  コマンドの結果を保存したい時などよく使う

if文

- […] が条件式になっている
- [ ] はtestコマンドに書き換え可能

```
HOGE=3

if [ $HOGE -eq 3 ]; then
    echo "3です"
elif [ $HOGE -ge 4 ]; then
    echo "4以上です"
else
    echo "その他です"
fi
```

testコマンドのオプションを調べれば
いろいろな条件式が出てくると思います

```
if test $HOGE -eq 3; then
    echo "3です"
elif test $HOGE -ge 4; then
    echo "4以上です"
else
    echo "その他です"
fi
```

for文

```
for i in "hoge" "fuga" "piyo"; do
    echo "${i}です"
done
```

- inの後に指定したリストから1つずつ取り出してループする
- 下の例では、lsコマンドの結果を1つずつ出力している

```
for file in `ls`; do
    echo "${file}"
done
```

関数

- 関数名 () {} で定義
- $@で全ての引数を、$nでn番目の引数を参照する
- コマンド実行時のように、引数を横に並べて関数を呼ぶ

```
test_func () {
    echo "this is test.  first arg is $1"
    for arg in $@; do
        echo "${arg}"
    done
}

test_func hoge fuga piyo
# $1 -> hoge
# $2 -> fuga
# $3 -> piyo
# $@ -> hoge fuga piyo
```

最後に

こんな時はUNIXコマンド、シェルスクリプトの出番かも

- GUIが使えない環境でちょっとした作業を一括で実行したい
- 決まった作業を自動化したい
- CRしたい

個人的に思うUNIXコマンドやシェルスクリプトのメリットは、環境構築ナシで
大抵のことはできるところかなと思っています（ pythonとかの方が自由度は上がる
けど、ライブラリ入れたりバージョン気にしたりが面倒臭い）