一行野郎で駆け抜けろ

ytoku (Tokushige Yuuki)

ytoku@mma.club.uec.ac.jp

予定

- コマンドライン操作の基礎
- ワンライナーの七つ道具
- ケーススタディ (問題： やってみよう!)

コマンドライン操作の基礎

- プログラムは標準入力･標準出力 (及び標準エラー出力) を持っている
- リダイレクトやパイプはこれらを繋ぎ合わせる
  - リダイレクト : command < file
    - ファイルの中身をコマンドの標準入力に繋ぐ
    - しかしワンライナーではコマンドライン引数へのファイルの指定や、cat コマンドをよく使う
  - リダイレクト : command > file
    - コマンドの標準出力をファイルに書き込む
  - パイプ : command1 | command2
    - command1 の標準出力と command2 の標準入力を接続

コマンドライン ワンライナー

- 複数のコマンドをパイプでつないでテキストデータを目的のデータに変換していく
- いくつかの簡単なコマンドの組み合わせでいろいろできる

ワンライナーの七つ道具

- grep
- sort
- uniq
- sed
- awk
- xargs
- find

番外

- cut
  - awk で代用可能
- tee
  - 出力を画面とファイルに同時に出力したいときに使う

grep

- 入力の中から、指定したパターンにマッチした行だけを取り出して出力する
- パターンは正規表現
- -o をつけるとマッチした部分だけを取り出す

```
$ grep ssh /etc/services
ssh     22/tcp                  # SSH Remote Login Protocol
ssh     22/udp
```

grep デフォルトの正規表現

- 指定子の一例
  - . は任意の 1 文字
  - [...] は指定した文字のいずれか。[^...] は逆にそれ以外
  - * は直前の文字 (種) が **0 回**以上
  - + は直前の文字 (種) が **1 回**以上
  - | は「または」
  - などなど。
- ソフトウェアによって書き方や機能がバラバラ
- 正規表現についてだけで一冊本が書けてしまう

sed (stream editor)

- 正規表現でマッチした部分を置換する
- \(...\) で括った部分にマッチした部分の文字列を、置換結果に埋め込むこともできる

```
$ grep ytoku /etc/passwd
ytoku:x:1000:1000:Tokushige Yuuki,,,:/home/ytoku:/bin/bash
$ grep ytoku /etc/passwd | sed 's/\([^:]\+\) \(.\+\),,,/\2 \1,,,/'
ytoku:x:1000:1000:Yuuki Tokushige,,,:/home/ytoku:/bin/bash
```

perl を使った方が便利かも

```
$ grep ytoku /etc/passwd | perl -pe 's/([^:]+) ([^:]+)(?=,,,)/$2 $1/'
ytoku:x:1000:1000:Yuuki Tokushige,,,:/home/ytoku:/bin/bash
```

sed の正規表現

- grep で見た表記と異なる
- 指定子の違いの例
  - + ではなく \+
  - * ではなく *
  - | ではなく \|
  - など

sort

- 行単位でソートする
- -k でソートのキーとなるフィールドを指定
- -t でフィールドの区切りを指定
- デフォルトでは文字列を対象としたソートが行われる
  - 数値をソートするには -n オプション

```
$ sort -t: -k3 -n /etc/passwd | head -3
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
```

uniq (unique)

- **連続した**重複した行を取り除く
  - 離れているとダメ
  - 通常は sort と組み合わせて使う
- -c オプションで何個あったかを数える

```
$ echo -ne "a\nb\na\n"
a
b
a
```

```
$ echo -ne "a\nb\na\n" | sort | uniq
a
b
```

```
$ echo -ne "a\nb\na\n" | sort | uniq -c
      2 a
      1 b
```

uniq in sort

- ところで重複を排除するだけなら uniq コマンドは要らない
  - sort の -u を使えばいい

```
$ echo -ne "a\nb\na\n" | sort -u
a
b
```

awk (1)

- 行をフィールドに分けて、条件に一致したレコード (= 行) について処理を行うコマンド
- 構文は 条件 { 処理 }; 条件 { 処理 } ...
  - { 処理 } を省略するとその行を表示する
- フィールドの区切り文字の変更は -F
- 先頭のフィールドから順番に $1, $2, $3...
- 行全体は $0

```
$ awk -F: '$1=="root"; $1=="ytoku"' /etc/passwd
root:x:0:0:root:/root:/bin/bash
ytoku:x:1000:1000:Tokushige Yuuki,,,:/home/ytoku:/bin/bash
$ awk -F: '$1=="ytoku" {print $1, $7}' /etc/passwd
ytoku /bin/bash
```

awk (2)

- 良く使う変数
  - NF: その行にフィールドがいくつあるか
  - NR: 今何行目か
- 特殊な条件
  - BEGIN: 最初に処理される
  - END: 最後に処理される
- 良く使う関数
  - printf とか
- いわゆる連想配列もある x['hoge']

printf 関数について少々

- C 言語を学ぶときに習うと思いますが
- printf(format, value1, value2, …)
  - format の例
    - %d: 数値
    - %s: 文字列
    - %9s: 右詰めで 9 文字確保した文字列
    - %3d: 右詰めで 3 桁確保した数値
    - %03d: 余った左端に 0 を詰めた 3 桁の数値
- などなど

awk で重複除去

- シンプルな指定で awk を使って重複行 (やフィールド) を除去できる

```
$ awk '!d[$0]++'
```

- さて、何が起きているかわかるでしょうか。

xargs

- 入力された値を引数としてコマンドを実行する
- -L1 をつけると 1 行ごとに実行する
  - デフォルトではすべての値を一度に渡す
- -I{} をつけるとコマンドに書いた {} の部分を置き換える

```
$ ls *.zip | xargs -L1 unzip
 →  ファイル名にスペースが含まれると二つの引数に分かれる
$ ls *.zip | xargs -I{} unzip {}
 →  ファイル名にスペースが含まれ手も正しく展開される
```

find

- 他と毛色が違ってこのコマンド自体でファイルに対する操作が結構いろいろできる
- 応用すると可能なことの一例
  - 一定期間使われていないファイルを一斉削除
  - パターンに一致するファイル名を持つファイルについて、ファイル名を表示してから指定したコマンドを実行する
  - などなどなど
- 機能が多すぎるので今回は紹介しません

ケーススタディ

- カレントディレクトリにある 1.jpg,2.jpg,...,10.jpg,... を、001.jpg,002.jpg,...,010.jpg,... にする

```
$ ls | awk '{printf("%s %03d.jpg\n", $1, $1)}' | xargs -L1 mv
```

ケーススタディ

- passwd ファイルからユーザが使っているシェルの一覧を得るには
  - passwd ファイルの構造： コロン区切りで 7 番目がシェル

```
# $FreeBSD: src/etc/master.passwd,v 1.40 2005/06/06 20:19:56 brooks Exp $
#
root:*:0:0:Charlie &:/root:/bin/csh
toor:*:0:0:Bourne-again Superuser:/root:
daemon:*:1:1:Owner of many system processes:/root:/usr/sbin/nologin
operator:*:2:5:System &:/:/usr/sbin/nologin
( 略 )
ytoku:*:9009:9009:TOKUSHIGE Yuuki:/home/ytoku:/usr/local/bin/bash
```

```
$ awk -F: '{print $7}' /etc/passwd | sort -u
```

ケーススタディ

- ユーザ ID が 2000-2099 のユーザ名を辞書順で
  - ユーザ名は 1 番目のフィールド
  - ユーザ ID は 3 番目のフィールド

```
# $FreeBSD: src/etc/master.passwd,v 1.40 2005/06/06 20:19:56 brooks Exp $
#
root:*:0:0:Charlie &:/root:/bin/csh
toor:*:0:0:Bourne-again Superuser:/root:
daemon:*:1:1:Owner of many system processes:/root:/usr/sbin/nologin
operator:*:2:5:System &:/:/usr/sbin/nologin
( 略 )
ytoku:*:9009:9009:TOKUSHIGE Yuuki:/home/ytoku:/usr/local/bin/bash
```

```
$ awk -F: '{print $7}' /etc/passwd | sort -u
```

ケーススタディ

- bash で一番よく使っているコマンド 5 つ
  - 履歴は ~/.bash_history に 1 行 1 コマンドラインで記録されている

```
ssh nest
less /etc/aliases
sudo vim /etc/aliases
sudo newaliases
```

```
$ awk '{print $1}' .bash_history | sort | uniq -c | sort -n | tail -5
```

なお、sudo を通したコマンドについても計上するには

```
$ awk '{if ($1=="sudo") {print $2} else {print $1}}' .bash_history \
  | sort | uniq -c | sort -n | tail -5
```

ケーススタディ

- FreeBSD の master.passwd ファイルからログイン可能ユーザだけを GNU/Linux の passwd ファイルに変換する
  - master.passwd ファイルから passwd ファイルに変換するためには 5 番目から 7 番目までのフィールドを削除すればよい。どちらもコロン区切りで、master.passwd のフィールド数は 10 、passwd のフィールド数は 7
  - ログイン可能ユーザであることを簡易的に判別する一つの方法は第 2 フィールド (パスワード) が * になっていないかを調べることである

ケーススタディ

- perl を使ってごまかしました
- awk でも $1,$2,... と頑張れば書けます

```
$ perl -alF: -ne 'splice @F,4,3; print join ":", @F' master.passwd \
  | awk -F: '$2!="*"'
```