Unixシェル

コンピュータリテラシ

2016年5月13日

建部修見

Unixシェル

- コマンドライン・ユーザインタフェイス
  - コマンドライン・インタプリタ
- ワイルドカード、パイプ、ヒアドキュメント、コマンド置換、変数、制御構造
- 会話型コマンド実行、スクリプト言語

Bシェルとcシェル

- Bourne shell (1979 Stephen Bourne)
  - Shell for UNIX Version 7
  - Modeled on ALGOL 68
  - /bin/sh
  - Almquist shell (ash), Bourne-Again shell (bash), Korn shell (ksh), Z shell (zsh)
- C shell (Bill Joy)
  - Modeled on C
  - tcsh

標準出力、入力

- Unixのプロセス（アプリケーション）が標準的に持つ入出力のインタフェイス
  - 標準入力
  - 標準出力
  - 標準エラー出力

パイプライン、リダイレクション

- `$ command1 | command2 ...`
  - command1の標準出力をcommand2の標準入力につなげてcommand1、command2を実行
- `$ command1 > file`
  - command1の標準出力をfileに出力
- `$ command1 < file1 > file2 2> file3`
- `$ cat file1 | command1 > file2 2> file3`
  - file1の内容をcommand1の標準入力とし、標準出力をfile2に、標準エラー出力をfile3に出力

リスト

- ;, &, &&, ||で区切られたコマンド
- & で終わるコマンドはバックグラウンドで実行
  - シェルはコマンド終了を待たない
- ; で区切られたコマンドは逐次的に実行
  - シェルはそれぞれのコマンド終了を待つ
- && はコマンドが正常終了したときに次のコマンドを実行

```
$ command && echo ok
```

- || はコマンドが異常終了したときに次のコマンドを実行

```
$ command || echo ng
```

ジョブ制御

- ^z, Control-z, ジョブの一時停止（suspend）
  - fgコマンド – 再開
  - bgコマンド – バックグラウンドで再開
  - killコマンド – 終了

```
$ kill %
```

- 実行中のジョブはjobsコマンドで分かる
  - 特定のジョブを再開、バックグラウンドで再開、終了したい場合はfg, bg, killのオプションで指定する

```
$ fg %3
$ kill %2
```

- waitコマンドで全ての実行中のジョブの終了を待つ

環境変数（environment variable）

- set（オプションなし）で全て表示

```
$ set
```

- PATH
  - コマンドの検索パス
  - ":"で複数パスを指定。コマンドを実行するとき、記載順にディレクトリを検索し見つかったコマンドを実行する

```
$ PATH=/bin:/usr/bin
```

  - 以後のコマンドにも適用するためにはexportする

```
$ export PATH
```

  - /usr/local/binを先頭に追加

```
$ PATH=/usr/local/bin:$PATH
```

環境変数（２）

- LANG（or LC_ALL）
  - 言語、国、コードセット（符号化方式）を指定
  - C

    ```
    $ LANG=C
    ```
  - ja_JP.eucJP, ja_JP.ujis, ja_JP.UTF8
  - localeコマンドで現在のロケールが分かる
  - locale -aで利用可能なロケールを表示する

環境変数(3)

- PWD
  - Current working directory
- HOME
  - Home directory

エイリアス

- 文字の置き換え
  `$ alias rm=‘rm -i‘`
  – rmコマンドをrm -iに置き換え
  – 削除するときに確認するようになる
  `$ alias ls=‘ls -F‘`
  – lsコマンドをls -Fに置き換え
  `$ alias ll=‘ls -l’`
  – llを実行するとls -lを実行する
- エイリアスを無効にするには¥でエスケープする
  `$ ¥rm foo`

変数

- 代入

  ```
  $ name=value
  ```

  valueがない場合はnull文字列

  ```
  $ i=1
  $ i=“Hello world!”
  ```

- 変数定義をなくす

  ```
  $ unset i
  ```

展開

- 中括弧展開
  `{foo,bar,baz}→foo bar baz`
- 変数展開
  `${name}` or `$name`
  – 変数の内容に置き換える
- コマンド置換
  `$(command)` or `` `command` ``
  – commandを実行しその標準出力に置き換える
- 算術展開
  `$((expression))`

展開(2)

- パス名展開

  * – 任意の文字列(null文字列含む)

  ? – 任意の文字

  [...] – どれかの文字

  [abc], [a-z], [[:alnum:]]

  存在するパスに展開される

```
$ ls a*        # aで始まる
$ ls .??*      # .ではじまり三文字以上
$ ls foo[0-9]  # foo0, foo1, ..., foo9で一致するもの
```

複合コマンド

- (list) – サブシェルでlistを実行
- { list; } – グループコマンド
- ((expression)) – 算術式の評価
- [[expression]] – 条件式の評価
- for name [in word]; do list; done
- case word in pattern [| pattern] …) list ;; … esac
- if list; then list; fi
- while list; do list; done

シェル関数定義

- name () compound-command

シェルスクリプト（１）

```
#! /bin/sh
echo Hi! My name is $(whoami)
```

- #! /bin/sh はスクリプトを実行するプログラムを指定。Rubyやpythonの場合は以下
  #! /usr/bin/ruby
  #! /usr/bin/python
- echoは標準出力に引数を表示する
- $(whoami)はコマンド置換。whoamiの出力に置き換え

シェルスクリプト(2)

```
#! /bin/sh
for i in 0 1 2 3 4 5 6 7
do
        for j in 0 1 2 3 4 5 6 7 8 9 a b c d e f
        do
                printf “¥x$i$j”
        done
done
```

- 0から127までの128バイトを出力

シェルスクリプト（３）

```
#! /bin/sh

humanize() {
    n=$1    # 第一引数をnに代入
    i=0
    # nが1024以上なら繰り返し
    while [ $n -ge 1024 ]
    do
        ((n/=1024)) # n=n/1024
        ((i++))         # i=i+1
    done
```

```
 case $i in
    0) echo $n ;;
    1) echo $n Ki ;;
    2) echo $n Mi ;;
    3) echo $n Gi ;;
    4) echo $n Ti ;;
    *) echo $n Pi ;;
    esac
}

humanize 65536
humanize 1048576
```

以降はコメントで無視される

設定ファイル

- ~/.bash_profile（~/.bash_login, ~/.profile）
  - ログインしたときに実行するファイル
- ~/.bashrc
  - 会話型でbashを起動したときに実行するファイル
- ~/.bash_logout
  - ログアウトしたときに実行するファイル

演習（１）

- 言語を変えてls -lとdateを実行してみよう

  $ LANG=C ls -l

  $ LANG=C date

- aで始まるファイルまたはディレクトリを表示するにはどうすればいいか
- aまたはAで始まるファイルまたはディレクトリを表示するにはどうすればいいか

演習（2）

- ~/.bash_profileに`alias rm='rm -i'`を追加しよう。一度ログアウトして、ログインしたあと、rmを実行したとき消去の確認をおこなってくるか確かめよう

演習（3）

- 以下のシェルスクリプトをhi.shというファイル名で作成しよう

```
#! /bin/sh
echo Hi! My name is $(whoami)
```

- sh hi.shで実行してみよう
- ./hi.shで実行できるか？実行できない場合はどうすれば実行できるようになるか考えよう

演習（4）

- 以下のシェルスクリプトをloop.shというファイル名で保存しよう
  - :は何も効果のないコマンドで、必ず条件を成立させる
- 実行しよう
- ^Zで一時停止させよう
- fgで再開させよう
- ^Zで再び一時停止させよう
- bgでバックグラウンドで実行させよう
- kill %で終了させよう

```
#! /bin/sh

i=0
while :
do
     echo i=$((i++))
     sleep 1
done
```

演習（５）

- loop.shをバックグラウンドで実行しよう

  $ sh loop.sh &

- このプログラムを停止させるにはどうすれば良いか

オプション演習（１）

- シェルスクリプト（３）をhumanize.shというファイル名で保存し、実行してみよう
- このスクリプトは配列を用いるともっと簡単になる

  unit=(“” Ki Mi Gi Ti)

  と配列を定義して以下のように参照する

  ${unit[$i]}

  配列を用いて書き換えてみよう