情報処理 第7回
プロセス，シェル

2000年5月31日

目 次

1 本日の目標 **1**

2 前回までの復習 (man コマンド) **1**

2.1 man コマンド 1

3 プロセス **1**

3.1 プロセスを調べる．親プロセスと子プロセス． 2

3.2 プロセスの消去 2

3.3 シェル 3

3.4 ワイルドカードを使って複数のファイルをまとめて扱う 4

4 サスペンド (suspend) とフォアグラウンド (foregroud) **5**

5 バックグラウンド (back ground) **5**

6 問題 **6**

1　本日の目標

- バックグラウンド処理 (&) を身につける. プロセス (ps) の抹消 (kill) を学ぶ. シェル (csh) の存在を意識する .

2　前回までの復習 (man コマンド)

2.1　man コマンド

既に man コマンドを活用しているようですが , 練習のため本日のキーワードについて man コマンドで調べてみましょう .

【関連項目】の項 (文末近くにある) を見ると関連する項目がわかります .

```
man ps
man kill
man csh
```

man コマンドの更に詳しい使い方を知りたい人は ,

```
man intro
```

や man コマンドに関するマニュアル

```
man man
```

を読むと良いでしょう .

問題 次のコマンドの意味と使い方を調べなさい.

```
rm

cp

mv

cd

chmod
```

3　プロセス

プロセス: (process) とは , コンピュータがプログラムを管理する単位のことです . ファイルについて学んで来ましたが , コンピュータは入力ファイルをもとにデータを処理し , 出力ファイルに書き込みます . この入力 , 処理 , 出力の作業をプロセスといいます . キーボードからの入力は標準入力 (stdin) というファイルからの入力と考えることが出来ます . 同様に , 画面への出力は標準出力 (stdout) あるいは , 標準エラー出力 (stderr) という名前のファイルへの書き込みと考えます .

ファイルと同様にプロセスには所有者が存在します . プロセス達は独立して動いていて , 各プロセスには番号が割り振られています . この番号をプロセス番号 (process ID) と呼びます .

3.1 プロセスを調べる．親プロセスと子プロセス．

まず，どんなプロセスが動いているかみてみましょう．ファイルの一覧を見るのに ls コマンドを使いましたが，プロセスの一覧を見るには，ps コマンドを使います．

```
ps
```

もっと詳しい情報を見てみます．ps -f と f オプションを付けて入力して下さい．次のような表示が出るはずです．(ログインしているマシンにより多少異なります.)

```
      UID   PID  PPID  C    STIME TTY      TIME CMD
 osaka 23224 23222  0 15:08:39 pts/9    0:00 -tcsh
```

右端の COMD の欄を見ると tcsh という名前のプロセスが動いていることが分かります．tcsh については後程説明しますが，コマンドをコンピュータにわかる言語に翻訳してくれるプロセスです.

左端の UID はプロセスの所有者です．このように，アカウントで表示されることもありますが，アカウントに対応するユーザの番号 (ユーザ ID) で表示されることもあります．

PID はプロセスの番号 (プロセス ID) です．

PPID はプロセスをそのプロセスを生み出した親プロセスの ID です．ファイルの木構造と同様に，プロセスも一つのプロセスが，いくつかの子プロセスを次々に生み出します．

C の欄は man ps を読んで，自習して下さい．

STIME はプロセスの開始時刻を表します．tcsh はログインした時刻になっている筈です．

ps -f では，1 つのウインドウ (端末) から実行したプロセスしか，表示されません．x ウインドウシステムでは，複数のウインドウが表示されています．また，時計やメール到着のアイコン (絵文字) も表示されています．これらのプロセスもユーザーが所有者になっています．

自分の所有しているプロセス全てを表示したいときは，ps -u ログイン名 を入力します．例えば，学籍番号 rp000001 の人の全てのプロセスは，

```
ps -u rp000001
```

と入力します．画面に入り切らない程表示されるときは，パイプで less に繋いで下さい．

3.2 プロセスの消去

何かの事情でプロセスを中止したいとき，プロセスに中止のシグナルを送ります．

練習のため時計のアイコンを消去してみましょう．時計のアイコンを表示するコマンドは xclock です. 文字列の検索をするのに grep というコマンドがありますので，これを利用して xclock のプロセス ID (PID) を調べます．

```
ps -u rp000001 | grep xclock
```

このとき，左端に表示される番号が xclock に対応するプロセス ID です．この番号を控えておいて下さい．プロセスを中止するには，kill コマンドを使います．(プロセスを殺すといったりします．)

```
kill プロセス ID
```

と入力します．(番号を間違えないようにしましょう．) 例えば，今控えた，xclock のプロセス ID が 23013 だとすると，

```
kill 23013
```

とします．

プロセスの中には，安全のため，簡単には終了出来ないものがあります．そのようなプロセスも，オプション -HUP をつけると終了出来る場合があります．

```
kill -HUP プロセス ID
```

それでも終了出来ないプロセスは，オプション -KILL をつけて，

```
kill -KILL プロセス ID
```

とします．

3.3 シェル

シェルは，入力文字列をコンピュータにわかる言葉で翻訳し，コマンドを実行させる (子プロセスを生む) 特別なプロセスです．シェルには，sh, tcsh, csh, などがありますが，RAINBOW のマシンにログインしたときに起動されるシェルは，csh です[1] ．シェルは，コマンドの解釈だけでなくプログラム言語のように変数に値を記憶させておくことも出来ます (シェル変数) . 例えば，変数 A に Aumi という値を代入するには，

```
set A = Aumi
```

とします．2 語以上をまとめて代入するときは，次のように ’ で囲みます．

```
set B = ’Hamasaki Aumi’
```

シェル変数の値を見るには，echo コマンドと $ を使って，

```
echo $A
echo $B
```

とします．

シェルの中からシェルを起動することも出来ます．今動いているシェルと同じ tcsh というシェルを起動してみましょう．

```
tcsh
```

を入力してから，先程のようにプロセスを調べてみて下さい．

```
ps -f
```

今起動した csh と併せて，2 つ tcsh が含まれていることがわかります．先程代入した A の値はどうなっているでしょうか?

[1] tcsh の方が便利なので私は tcsh に設定しております

```
echo $A
```

未定義というメッセージが返されるはずです．このように，シェル変数は一つのシェルの中だけで有効です．新しく発生させたシェルから抜け出るには，exit を使います．

```
exit
```

再び A, B の値を確かめてみましょう．

```
echo $A
echo $B
```

最初に代入した文字列が表示される筈です．A, B に何も代入されていない状態に戻すには，

```
unset A
unset B
```

とします．

```
echo $A
echo $B
```

で未定義の状態を確認して下さい．

3.4 ワイルド カード を使って複数のファイルをまとめて扱う

ここでは，ファイル名の展開に関するメタキャラクタ (ワイルド カード) * について学習します．(その他のワイルド カード はレポート問題です．) まず，/usr/bin の中にあるファイルの一覧を見てみます．次のように入力して下さい．

```
ls /usr/bin
```

この中に，今までに使ったコマンド csh，kill などが入っています．次に，次のように入力してみて下さい。

```
ls /usr/bin/w*
```

また，次のようにも入力してみましょう．

```
ls /usr/bin/ch*
```

出力されるファイル名の特徴がわかるでしょうか？

ls /usr/bin/w* では、w で始まるファイル名が (ディレクトリ名 /usr/bin/ とともに) 全て表示されています。同じように ls /bin/ch* では、ch で始まるファイル名が全て表示されます。

このように文字 * には、(空列を含む) 任意の文字列に展開するという特別な意味があります。このような特別の意味をもつ文字をメタキャラクタと呼びます。シェルは，入力文字列をコマンド と解釈する過程で，メタキャラクタの解釈を行います．今まで出て来た，メタキャラクタには，>，>>，<，>& | などがあります．先程 echo コマンド と一緒に使った $ もメタキャラクタです．これらがどういう風に解釈されるかについては，既に学習しました．

メタキャラクタのうちでファイル名の展開に使われる文字をワイルド カード と呼びます．上で実験したように，ワイルド カード を使うと複数のファイルをまとめて，扱うことが出来ます．

問題

echo は入力文字列をそのまま画面に出力するコマンドである．次のコマンドの実行結果がどうなるか予想せよ（実際に試して，予想が正しかったか確認せよ .)

```
echo ./
echo ./*
```

4 サスペンド (suspend) とフォアグラウンド (foregroud)

まず，

```
man csh
```

を実行してマニュアルが表示させます．その時

```
Ctrl-z
```

と入力すると，

```
停止（利用者要求による）
```

と表示され，コマンドが入力できるようになります．ここで，

```
jobs
```

と入力すると，

```
[1]  + 停止（利用者要求による)man -F csh
```

の様な表示が出て来ます．ここで，

```
fg
```

とすると，またマニュアルが表示されます．マニュアル表示を終えるには，Ctrl-c を入力して下さい．

Ctrl-z でプロセスを中断することをサスペンド（ suspend ）といいます．fg コマンドで中断させていたプロセスを再開することをフォアグラウンド（ fore ground ）に戻すといいます．次のバックグラウンドと比較して感じを掴んで下さい．

5 バックグラウンド (back ground)

```
\mule
```

とすると, 画面に新しく mule のウインドウが開きますが，もとのウインドウは使えなくなります．そこで **mule** タイプしたウインドウにカーソルを入れて Ctrl-z と入力すると, 先程と同様に

```
停止（利用者要求による）
```

と出ます．こうするともとの画面から入力が出来る状態になりますが，今度は新しく mule の画面から入力する事が出来なくなります．この時

```
bg
```

とすると

```
[1]  + mule &
```

と表示されます．先程の fg と違い，bg を実行した端末で続けてコマンド入力が行えます．このとき mule はバックグラウンド（back ground）で実行されているといいます．

kterm を起動する際に，

```
\mule &
```

と入力すると，Ctrl-z で停止してから，bg と入力したのと同じことになります．

6 問題

- csh のファイル名展開で使われるワイルドカードにはどのようなものがあるか，* 以外のものについて調べ，その用法について自習せよ．