シェルプログラミング入門

米野　誠[1]

1　はじめに

シェル (shell) はユーザと UNIX カーネル (kernel) との仲介を行なうコマンドインタープリタ (コマンド解釈プログラム) である．ユーザが入力したコマンドは，このシェルによって，解釈処理を行ない，最終的にカーネルに引き渡し，コマンドが実際に実行される．また，実行結果に何らかの出力がある場合は，カーネルからシェルに渡され，画面やファイルに出力される．このように，シェルが仲介を行なっていることによって，コマンドの起動はもちろんのこと，メタキャラクタの解釈 (`ls *.c` など)，パイプやリダイレクトによる入出力の切替えなどを行なうことができる．

シェルに一連のコマンドをまとめて与えるには，適当なファイルに一連のコマンドを羅列し，そのファイル名をコマンドとして入力するとよい．このファイルをシェルスクリプトといい，UNIX のコマンドはもちろんのこと，条件判断や繰り返しといった基本的な制御を行なわせることもできる．本稿では，代表的なシェルとして標準シェル (Bourne shell) と，これを拡張させた C シェル (C shell) を使ったプログラミング方法について説明する．

2　標準シェル

2.1　スクリプトの書き方

標準シェルは，S.R.Bourne 氏によって開発された UNIX 標準のシェルである．標準シェルでのシェルスクリプトを記述するには，次の 2 点に注意する必要がある．

- シェルスクリプトの先頭 (ファイルの先頭) は `#!/bin/sh` と表記する
- シェルスクリプトを実行するには `chmod +x file_name` で実行権を与える．

なお，シェルスクリプト中の注釈は，行頭に `#` をつけるとよい．

[1]情報科学センター，komeno@isct.kyutech.ac.jp

2.2 変数と定数

プログラミング言語と同様，標準シェルでも変数に値・文字列を代入することができる．シェル変数には，英文字，アンダースコアを指定することが可能である．例えば，nameというシェル変数にhikoという文字列(データ)と代入したい場合は，

```
name=hiko
```

とする．また，定数の置換を行なう場合は，$を使う．nameの内容を表示させるには，

```
echo $name
```

とすればよい．echoとは，入力した文字をそのままエコーバックして画面表示するコマンドである．また，シェル変数の中には特別な値が代入される変数もある．

$$	PID(プロセスID)	PIDが代入される
$0	プログラム名	プログラム名が代入される
$*	引数	コマンド行に入力された全ての引数

2.3 入出力

データの入力は，引数・シェルコマンド(read または，line)を使って行なう．例えば，シェルスクリプト(ex01)を次のようにして実行した場合

```
ex01 mdata1 mdata2
```

シェルスクリプトex01は，引数として2つのデータをシェル変数に代入している．引数をスクリプト中で利用するには，$マークの次に引数の番号を書く．引数が1つの場合は$1となり，上記の例ではmdata1がその内容となる．

また，シェルコマンドのread または，lineとは，一行の文字列を読み込むコマンドである．このコマンドを使って，指定したシェル変数に文字列(データ)を読み込むことができる．

シェル変数の内容を表示するには，標準出力に出力するコマンド(echo)を使うとよい．もちろんリダイレクトを使って結果をファイルに保存することもできる．

2.4 基本制御

シェルスクリプトでも，通常のプログラミング言語と同じように条件判断や繰り返し処理を行なうことができる．標準シェルで使用可能な基本制御文を以下に示す．

- if文 ・・・ 条件判断を行なう
- while文 ・・・ 一定回数処理を繰り返す
- case文 ・・・ 多分岐判断を行なう
- for文 ・・・ 一定回数の処理を繰り返す(繰り返し変数に文字列を使用する)

なお，if,while 文は，終了条件として条件評価コマンド (test) を併用する．その test コマンドをまず説明しておく．

test コマンドの一般の書式は，test expression となっている．ここで expression は，条件式のことである．結果が真であると 0 を返し，偽であると 1 を返す．

演算子	満足した場合には 0 を返す．
$string_1 = string_2$	$string_1$ は $string_2$ と同じである．
$string_1 != string_2$	$string_1$ は $string_2$ と同じでない．
string	string はヌルである．

なお，変数を数値的に比べる場合の評価も存在するが，ここでは説明を省く．

2.4.1 if 文

if 文の書式を以下に示す．まず，条件式として command-test が実行される．これは，前に述べた test コマンドを使う．この条件が真であれば command を実行する．そうでなければスキップされる．

```
if command-test
then
     command1
     command2
     ........
     command(n)
fi
```

また，else が入った書式は次のようになる．

```
if command-test then
     command1
     command2
else
     command3
     .......
     command(n)
fi
```

例 1) if 文の使い方

以下の例では，read 文でシェル変数 moji に文字列を読み込む，それを if 文によって判断し echo 文で結果を出力している．

プログラム

```
#!/bin/sh
read moji
if test $moji = MON then echo 'Today is Monday'
                    else echo 'Today is not Monday'
fi
```

実行結果

```
% mon
MON
Today is Monday
% mon
TUE
Today is not Monday
%
```

2.4.2 while 文

while 文は，一定回数ループを行なうものである．このコマンドの書式は次のようになる．

```
while  command-test
do
       command1
       command2
       ........
       command(n)
done
```

command-test には，test コマンドを使って繰り返し条件を記述する．この時の評価値が 1 であれば繰り返しを抜け，逆に 0 であると do-done 間のコマンドを繰り返すことになる．

例 2) while 文の使い方

プログラム

```
#!/bin/sh
i=1                          # 初期値
while test "$i" != 6         #i が 6 になったら whileloop から脱出する
do
       echo $i
       i=`expr $i + 1`       #i に 1 を足して，i に代入する
done
```

実行結果

```
% wa
1
2
3
・・以下省略・・
```

例 3) while 文と read 文の組み合わせ方

カレントディレクトリーにあるファイルの中身を，ファイル名をキーボードから入力することによって表示させるシェルスクリプト

プログラム

```
#!/bin/sh
while read num              # file 名を読み込む
do
   cat $num                 # 読み込んだ file 名の file の内容を表示
done
```

実行結果

```
% cat1
file1
   ........file1 の内容......
file2
   ........file2 の内容......
file3
   ........file3 の内容......
^C
%
```

このシェルスクリプトは繰り返し条件でプログラムが終了することがないので，コントロール C を入力して強制終了させる．

2.4.3 case 文

case 文は，一つの値 (入力値) と指定値とを比較して同じ場合，指定されたコマンドを実行するものである．書式は，次のようになる．

```
case value in
   1 ) command1
       command2
       ...
       command(n) ;;
   2 ) ...
       command ;;
   n ) command;;
esac
```

この場合，値 value は，値 1,2, ～ n を比較して，マッチングするまで比較される．そしてマッチングした行から，2 つのセミコロンまでを実行して case を終了する．

例 4) case 文の使い方

プログラム

```
#!/bin/sh
echo '1:who 2:finger'          # メニューの表示
read num                       # 番号の読み込み
case "$num"
in
   1 ) who;;                   #1 を入力すると who を実行
   2 ) finger;;                #2 を入力すると finger を実行
   * ) exit                    #1,2 以外を入力すると実行を止める．
esac
```

実行結果

```
% sentaku
1:who 2:finger
1
komeno   ttyp1   Jul 21 13:58   (hiko.isct.kyutec)
% sentaku
1:who 2:finger
2
Login       Name               TTY Idle    When    Where
komeno   Komeno-Mt.man-Makoto  p1      Tue 13:58  hiko.isct.kyutec
%
```

2.4.4 for 文

for 文は，同じコマンドを指定された文字列の数だけ実行するものである．これは，一般の言語と違い，繰り返し数を指定するのでなく，変数に代入する単語数で繰り返し数を決める．書式

は，次のようになる．

```
for var in lan1 lan2 lan3 ...lan(n)
do
    command1
    command2
    ........
    command(n)
done
```

値 lan1 がまず始めに変数 var に代入され，指定したコマンドが実行される．これが lan(n) まで同じ作業が続く．従って，全部で n 回の処理をすることになる．

例 5) finger を使って，教育用 WS の 3 台 (戸畑 yuri,renge,hagi) に誰が login しているか調べるシェルスクリプト

プログラム

```
#!/bin/sh
for host in yuri renge hagi    # 変数を host とする.
do
    finger @$host              # @ は，その後に続くマシン名に
                               # 接続されデータが出力される.
done
```

実行結果

```
% whoall
[yuri.isct.kyutech.ac.jp]
Login       Name               TTY Idle    When    Where
komeno   Komeno-Mt.man-Makoto  p1      Tue 14:28  hiko.isct.kyutec
[renge.isct.kyutech.ac.jp]
Login       Name               TTY Idle    When    Where
tyamano  yamanoue takashi       p1 2:53 Tue 10:31  hiko.isct.kyutec
tyamano  yamanoue takashi       p2 3:36 Tue 10:32  hiko.isct.kyutec
[hagi.isct.kyutech.ac.jp]
No one logged on
%
```

3 Cシェル

Cシェルは，標準シェルを拡張し使いやすくしたものである．エイリアス機能やヒストリ機能など，標準シェルには備えられていない便利な機能がある．

3.1 スクリプトの書き方

スクリプトの書き方については，標準シェルとCシェルはほとんどかわらない．ただし，シェルスクリプトの先頭には `#!/bin/csh` と記述しなければならない．

3.2 変数と定数

標準シェルと同様に変数に値・文字列を代入することができる．シェル変数に英文字，アンダースコアを指定することは標準シェルと同じであるが，代入の仕方が違う．例えば，name に hiko と代入したい場合は，

```
set name=hiko
```

とする．また，定数の置換は，標準シェルと同様に，$ を使う．また，標準シェルと同様に特別なシェル変数が存在する．そして，Cシェル特有のCシェル変数というものがある．

3.3 入出力

データの入出力方法は，標準シェルと同様に行なえる．ただし，Cシェルでは，シェルコマンド(read)を使うことはできない．

3.4 基本制御

Cシェルで使用できる基本制御文は，if,while,switch,foreach 文の4つである．

- if 文 ・・・ 条件判断を行なう
- while 文 ・・・ 一定回数処理を繰り返す
- switch 文 ・・・ 多分岐判断を行なう
- foreach 文 ・・・ 一定回数処理を繰り返す(繰り返し変数に文字列を使用する)

なお，標準シェルでは test コマンドを使って条件判断を行なっていたが，Cシェルでは演算子が存在するので test コマンドは併用しなくてよい．

3.4.1 if 文

この if は，標準シェルと同じように扱えるが endif の部分が異なるので注意されたい．書式は次のようになる．

```
if (expression) command
```

または,

```
if (expression) then
    command1
    .......
else
    command2
    .......
endif
```

である．また，expressionにおいて使用できる演算子は次の通りである．この他にも，一般的な<, >, <=, >=も使用できる．

==	肯定
!=	否定
&&	論理積
||	論理和

例6) if文の使い方

プログラム

```
#!/bin/csh
if ($1 == 1) then      # 引数が1だったらwhoを実行
   who
else                   # 引数が1以外だったらfingerを実行
   finger
endif
```

実行結果

```
% ide 1
komeno   ttyp1   Jul 30 16:23   (hiko.isct.kyutec)
% ide f
Login       Name               TTY Idle    When    Where
komeno   Komeno-Mt man-Makoto  p1       Thu 16:23  hiko.isct.kyutec
%
```

例7)if文の使い方

通常のファイルであれば'ファイルです'と表示し，ファイルでなければ'ファイルでありません'と表示させるシェルスクリプト．

プログラム

```
#!/bin/csh
if (-f $1) then                    #引数が通常ファイルなら真
   echo ファイルです
else
   echo ファイルではありません #ファイルでなければ偽
endif
```

実行結果

```
% file_test genkou.tex
ファイルです
% file_test test
ファイルではありません
%
```

この例では，genkou.tex はファイルであるが，test はディレクトリなのでファイルでないと判断した．

3.4.2 while 文

標準シェルの while 文と同じである．書式は次のようになる．

```
while (expression)
      command1
      .......
      command(n)
end
```

例 8) while 文の使い方

プログラム

```
#!/bin/csh
set i=0
while ($i != 5)          # 5 になるまでループする
    echo $i              # i を表示
    @ i++                # i+1 を実行
end
```

3.4.3 switch 文

switch 文は，標準シェルの case 文と同様，選択文である．書式は次のようになる．

```
switch (string)
case lan1:
     command1
     .......
     breaksw
case lan2:
     command2
     .......
     breaksw
 ............
default:
     command(n)
     .......
     breaksw
endsw
```

string を lan に対応させる．一致していれば，breaksw までのコマンドを実行させる．もし，一致するパターンが無ければ，default に続くコマンドを実行させる．

例 9)switch 文の使い方

プログラム

```
#!/bin/csh
switch ($argv)
case -e:          #引数が -e であれば， emacs を実行
       emacs
       breaksw
case -n:          #引数が -n であれば， ng を実行
       ng
       breaksw
default:          #引数を -e,-n 以外にすると exit する．
       exit
       breaksw
endsw
```

3.4.4 foreach 文

foreach 文は，標準シェルの for 文にあたる．書式は次のようになる．

```
foreach var ( list )
        command1
        command2
        ........
        command(n)
end
```

これは，list の中の単語を var の中に次々に代入して指定されたコマンドを実行する．

例 10) foreach 文の使い方

プログラム

```
#!/bin/csh
foreach file (*)    # 変数 file にカレントディレクトリのファイルを代入する
     echo $file >> /tmp/memo$$
                    #file の中身を /tmp/memo$$ に追加格納する
end
cat /tmp/memo$$     # データが格納されたファイルを表示する
rm -f /tmp/memo$$   # 作業ファイルを削除する
```

実行結果

```
% ls_echo
editor
genkou.aux
genkou.dvi
genkou.log
genkou.tex
ls_echo
mon
sentaku
wa
whoall
%
```

4　具体的な使い方

4.1　コンパイル用のシェルスクリプト

ここでは，言語コンパイルと実行を行なうシェルスクリプトを示しながら具体的な使い方を説明する．

Fortran プログラム等は，まず始めにコンパイルして，実行ファイルを作り，そしてその実行ファイル名をコマンドとして入力することで，はじめてプログラムが実行可能となる．この流れをシェルスクリプトのファイルに上から順に記述するとよい．そうすると，二つの仕事が一度にできることになる．次に例を示す．

例 11) 二つの数を入力して，その和を出力する Fortran プログラム *sample.f* を使い，ファイル *input.data* から二つの数を入力し，その和をファイル *output.data* に出力するシェルスクリプト．

プログラム

```
#!/bin/sh
f77 -o sample sample.f              #sample.f をコンパイルする
sample < input.data > output.data
      # 実行ファイル sample を実行し， input.data からデータ入力し，
      #output.data に出力
```

実行結果

```
% chmod +x fort
% fort
sample.f:
MAIN:
% cat output.data
a+b=     7.00000
%
```

こんな，シェルプログラムも作れる．

```
#!/bin/sh
A=sample
B=input.data
C=output.data
f77 -o $A $A.f
$A < $B > $C
```

このプログラムでは，ファイル指定をしている A,B,C を書き換えることによってソースファイルプログラム，入力データーファイル，出力データファイルを変更することができる．

例 12) 0,5,6 番以外の装置番号を使用したプログラムと，そのコンパイル，ファイルの割り当ておよび実行を行なうシェルスクリプト．

プログラム

```
#!/bin/sh
f77 -o sample sample.f
ln -s input.data fort.10
ln -s output.data fort.11
sample
rm -f fort.10
rm -f fort.11
```

実行結果

```
% fort2
sample.f:
MAIN:
% cat output.data
a+b=     7.00000
%
```

ここで，`ln -s input.dat fort.10` とは，input.data を fort.10 として使うために，リンクをしたわけである．次の行も同じ意味である．

4.2 .cshrc による環境設定の例

ここでは，ログイン時に環境設定等を自動的にするために環境設定コマンド等を書き込む.cshrc というファイルに書き込むコマンドを紹介する．なお，これは，シェルプログラムのファイルである．現在，情報科学センターでは標準環境を設定するために各ユーザの.cshrc の中に，

```
source /usr/local/isc/defaults/cshrc
```

と書き込んでいる．これは，環境設定を source 以下のファイルでまかなっているということである．

この /usr/local/isc/defaults/cshrc の中身を覗くと，次に示すようなスクリプトになっている．ここでは，この中で alias コマンド，set コマンド，setenv コマンド，limit コマンド，umask コマンドについて説明をする．

```
set     path = ( /usr/lang /usr/ucb /usr/bin \
/usr/local/bin /usr/local/X11R5/bin /usr/openwin/bin \
/usr/openwin/bin/xview /usr/local/bin/Wnn4 /usr/local/bin/mh \
/usr/ext/bin /edu/bin ~/bin . )
limit   coredumpsize  0 kbytes
umask   022

if (! $?prompt ) exit

set     filec
set     ignoreeof
set     noclobber
set     history = 100
set     notify
set     prompt = "`whoami`% "

setenv JSERVER `hostname`
setenv LANG japanese
setenv MANPATH '/usr/lang/man:/usr/man:/usr/local/man: \
      /usr/local/X11R5/man:/usr/openwin/man:/usr/ext/man'
setenv EDITOR emacs
setenv PAGER less

alias  uum      'uum -u -U -D$JSERVER'
alias  pwd      'echo $cwd'
```

4.2.1 alias コマンド

これは，別名機能である．エリアスと普通読むが，ユーザによってはアリアス，アライアスと発音する人もいる．この別名の設定は，

alias 新設定コマンド 本来のコマンド

である．例えば，dir を ls と同じにするには次のようにすればよい．

```
alias dir 'ls -l'
```

また，途中で設定を止めたくなることがある時は，unalias コマンドを使う．

```
unalias dir
```

4.2.2 set コマンド

これは，変数の設定などでも使われるものである．この.cshrc で設定される set の次の単語はシェル変数である．

path	コマンドを探す対象となるディレクトリーを設定する．
filec	これを設定するとファイルの先行する文字列を入力して ESC を入力すると一致する範囲最大で置き換える
ignoreeof	コントロール D で終了するのを防ぐ
noclobber	存在するファイルに出力リダイレクトをしないようにする．
notify	これを，設定するとジョブが終了するとシェルが知らせる．バックグラウンドジョブがあるときに有効

4.2.3 setenv コマンド

C シェルの環境変数は，setenv で設定される．ただし，この環境変数を設定する時にはイコール記号は使えない．`setenv VAR word` という書式を持つ．これは，環境変数 VAR に値 word が設定される．ここで使用されている環境変数の意味は次の通りである．

JSERVER	漢字変換をするサーバー
LANG	使用言語
MANPATH	オンラインマニュアルを実行した時にマニュアルを探す対象
EDITOR	エディター指定
PAGER	ページャ指定

4.2.4 limit コマンド

limit は，limit の次に来るソースの量の制限をする．この cshrc の例では，ソースが coredumpsize となっている．これは，コアファイルの大きさを決める．coredumpsize の次は，指定ソースファイルの最大許容量である．この場合は，コアファイルの大きさは 0kbites ということになる．

4.2.5 umask コマンド

umask は，作成マスクを設定する．この指定される数字は 8 進数である．これは，新規ファイル・新規ディレクトリーに対しての許可を与えるもので，022 とは，自分以外のユーザにも読み込み許可を与えるということである．ここの数字を 077 にすると，自分以外のユーザは読み込むことは不可能となる．これは，`'umask 077 '` を実行すればよい．

5 おわりに

今回は，シェルプログラムの基本を説明した．なお，引用符・正規表現等，シェルの特殊な表現の説明は省いたので市販の本を参考にされたい．