シェルプログラミング

- コマンドをパイプでつなげるだけでは済まないような、ある程度まとまった処理を複数のコマンドを制御構文を用いたりしてファイルとしたものを（シェル）スクリプトと呼ぶ。シェルプログラム、バッチなどともいう。
- .bash_profile もシェルスクリプトなので、このファイルを解読したい

163

どのシェルを使うか？

- シェルスクリプトは、どのシェル用のものかということで書き方（ 文法 ） が異なる
- bash特有の文法を学ぶよりは、bashのもとになったsh シェル（Bourne shell）用の文法を取り上げよう（bashはGNUプロジェクトがshと互換性を持たせ、機能を強化したシェル。bourne again shell の意味）

164

（例）以前のクイズで、単語ファイルの真ん中の語を求めたが、

1. ファイルに単語がいくつあるか？
2. 真ん中の単語は何番目に当たるか？
3. その単語は何か？

という3つの処理が必要だった。
これを行うシェルスクリプトを作ってみよう。

165

シェルスクリプトの作成手順

1. シェルスクリプトをテキストエディタを使って作成する
2. 作成したスクリプトを実行可能ファイルにする
3. スクリプトの動作確認を行う
4. 動作確認でエラーが出れば修正する

166

基本事項　シェルスクリプトの形式

- $ sh ファイル名
  でも起動できるが、大抵は次のようにする。
- ファイルの最初に
  #!/bin/sh
  を書き、使用するシェルを明示する。
- 処理は左から右、上から下へ進む

（例） 次の内容のファイルhello.shを作る。

```
#!/bin/sh
#    #で始まる行はコメント
echo  'Hello, World'
```

167

このファイルを　ls –l　で見ると

```
- r w - r - - r - -   1  fr0123     hello.sh
```

のようになっているので、実行権を

```
$ chmod  +x  hello.sh
```

のようにして与える。（これをしないと許可がありませんといわれて、実行できない）

実際に実行するには

```
$  ./hello.sh   または   hello.sh
```

と打つ。（コマンドサーチパスにカレントディレクトリが入っているかどうかです！）

168

シェル変数

- 代入
  variable=value
  変数名variableに値valueを代入する
  variableの変数がない時は作られる
  =の前後に空白を入れてはいけない
- 参照
  $variable
- 変数の削除
  unset variable

169

シェル変数の操作例

```
$ w=/usr/local
$ echo $w
/usr/local
$ ls $w
（略）
$ unset w
$
```

170

ワードリストを値とするシェル変数

スペースで区切られた文字列の並びを変数の値としたいときは
$ w=(/usr /usr/bin /usr/local)
のように値の部分を丸括弧で囲む

値全体を参照するときは
$ echo ${w[@]}
個々の要素を参照するときは
$ echo ${w[0]} ${w[1]} ${w[2]}
のようにする

171

バッククォートによるコマンド置換

- バッククォート（アポストロフィと向きが逆のもの）で囲まれた場合、その中に書かれたコマンドを実行し、その結果をその位置に埋め込む

（例）$ echo Today is `date`
$ w="Today is `date`"

（注意） このようにバッククォートはアポストロフィと意味が全然違うので、きちんと使い分けること！

172

数値と文字列

- shのデータ形式: 基本的に文字列
- 計算が必要な時:
  `expr 数式`
  注意:数式の各要素は空白で区切る
  （例） $ echo `expr 2 + 3`
  $ echo `expr 2 ¥* 3 - 1`
  $ echo `expr ¥( 2 + 4 ¥) / 5`
  $ echo `expr 4 % 2`

173

辞書の真ん中の語を求めるシェルスクリプトを作ってみよう

1. まず変数dictwcに辞書のサイズを求める
   #!/bin/sh
   dictwc=`look . | wc -l`
2. dictwcの半分（真ん中）を求める
   mid=`expr $dictwc / 2 + 1`
3. 真ん中の語が何番目なのかとその単語を表示する
   echo -n "$mid/ $dictwc = "
   echo `look . | head -$mid | tail -1`

174

前スライドの欠陥

辞書が偶数個の単語から構成されるときは、まちがった結果を表示してしまう

↓

偶数個の時は、真ん中の2個の単語を表示させたい

↓

奇数個の時と偶数個の時で処理を分ける

175

辞書が偶数個でも大丈夫なように書くと

```
#!/bin/sh
dictwc=`look . | wc -l`
if [ `expr $dictwc % 2` -ne 0 ]; then
  Nth=`expr $dictwc / 2 + 1`
  echo ` look . | head -$Nth | tail -1`
else
   Nth=`expr $dictwc / 2`
   echo ` look | head -$Nth | tail -1`
   Nth=`expr $NTH + 1 `
   echo ` look . | head -$Nth | tail -1`
fi
```

176

if の書き方1

```
if [ 条件1 ]
then
    コマンド列1
elif [ 条件2 ]
then
    コマンド列2
else
    コマンド列3
fi
```

177

if の書き方2

```
if [ 条件1 ] ; then コマンド列1
elif [ 条件2 ] ; then コマンド列2
else コマンド列3 ; fi
```

（つまり then, else, elif, fi はセミコロンを使って同じ行に書くことができる）

（注）[と条件の間にはスペースを入れる

178

if の書き方3

```
if  test  条件1
then
    コマンド列1
elif  test  条件2
then
    コマンド列2
else
    コマンド列3
fi
```

179

数値比較演算子

表記	意味
a −eq b	Equal to
a −ne b	not equal to
a −gt b	Greater than
a −lt b	Less than
a −ge b	greater than or equal to
a −le b	less than or equal to

180

論理演算子

表記	意味	例
! expr	Exprでなければ真（否定）	! -r file.txt (file.txtが読み込み可能でなければ真)
expr1 -a expr2	expr1かつexpr2が成り立てば真	$a -eq 1 -a $b -eq 10
expr1 -o expr2	expr1またはexpr2が成り立てば真	$a -eq 1 -o $b -eq 10

181

パターン比較演算子

表記	意味
string = = pattern	文字列stringと文字列patternが等しければ真
string != pattern	文字列stringと文字列patternが等しくなければ真

182

ファイル属性演算子

-r filename　ファイルが読めれば真
-w filename　ファイルが書ければ真
-x filename　ファイルが実行可能ならば真
-e filename　ファイルが存在すれば真
-O filename　そのシェルスクリプトを実行した
　ユーザがfilenameの所有者であるとき真
-f filename　ファイルが通常のファイルなら真
-d filename　ファイルが存在しディレクトリなら真

183

ファイル関係演算の例

```
#!/bin/sh
#  カレントディレクトリにファイルa.outが存在し,
#  実行可能なら実行
if [ -f ./a.out -a -x ./a.out ]; then
      ./a.out
else
      echo "a.out not found"
fi
```

184

コマンドのグループ化

現在のシェルの子プロセス（サブシェル）として実行される

- cmd1; cmd2
  コマンドcmd1の後にコマンドcmd2を実行
- (cmd1; cmd2)
  cmd1とcmd2をコマンドグループとして実行
- cmd1 && cmd2
  cmd1が成功したらcmd2を実行
- cmd1 || cmd2
  cmd1が失敗したらcmd2を実行

185

グループ化されたコマンドの例

コンパイルが成功した時だけ，./a.outを実行

```
$ gcc test.c && ./a.out
```

カレントディレクトリを変えずに実行

```
$ ( cd prog1; ls )
```

186

繰り返し処理　for

for　変数　in　単語リスト
do
　　コマンド列
done

（注）for　変数　in　単語リスト　;　do　コマンド列　;　done　も可

187

for　の例

```
#!/bin/sh
for  i  in  *.c
do
    rm  -i  $i
done
```

188

繰り返し処理　while

```
while　条件
do
　コマンド列
done
```

（注）
while　条件　;　do　コマンド列　;　done
も可

189

while の例

```
#!/bin/sh
i=1
a=0
while [ $i -le 3 ]
do
    a=`expr $a + $i`
    echo $i $a
    i=`expr i + 1`
done
```

190

もっと勉強したい方は

ブリン著 「入門UNIXシェルプログラミング」
ソフトバンク

Robbins & Beebe 著　日向あおい訳
「詳解　シェルスクリプト」 オライリー・ジャパン

など

191