情報処理論1a

6章「スクリプト」

中島康彦

§6. 0 今日の作業ディレクトリを作る

1. % `cd` ⇒ ホームディレクトリへ移動
2. % `mkdir chap06` ⇒ ディレクトリchap06を作成
3. % `cd chap06` ⇒ ディレクトリchap06へ移動
4. % `pwd` ⇒ 作業ディレクトリの確認

今日は，簡単なプログラムに挑戦.
マウス操作は，同じ操作の繰り返しには，実は不向きです.

§6. 1 スクリプトの主な実行環境

キーボード操作は再現できるが，マウス操作は再現できない．
再現する手段が，スクリプト

- sh
  最も古くからあるシェル
- csh
  shに，C言語に似た文法を採り入れたもの
- tcsh
  cshに，mule(emacs)に似たコマンドライン編集機能を採り入れたもの
  メディアセンターのログインシェルでもある
- perl
  Practical Extraction and Report Language
  さらに高度な文字列抽出，文字列処理が可能
  上記シェルとの互換性はない
  サーバ上での文字列処理によく使われる

§6. 2 簡単なシェルスクリプト

1. % vi hello

2. iecho "Hello."[Enter]date[Esc]ZZ
   echo "Hello."
   date■
   ~

3. % sh hello ⇒ shによる実行
   Hello.
   2000年04月24日 11時21分08秒

4. % csh hello ⇒ cshによる実行
   Hello.
   2000年04月24日 11時21分18秒

5. % tcsh hello ⇒ tcshによる実行
   Hello.
   2000年04月24日 11時21分28秒

§6. 3 シェルスクリプトのファイルモード

1. % ./hello
   ./hello: Permission denied.

2. % ls -l hello
   ```
    +---------user---r-  所有者による読み取り
    |+--------user---w-  所有者による書き込み
    ||+-------user---x-  所有者による実行/ディレクトリ検索
    |||
   -rw-r--r-- ......... hello
   ```

3. % chmod a+x hello　　⇒ 実行の許可
   % ls -l hello
   -rwxr-xr-x .... hello

4. % ./hello　　⇒ 特に指定しない場合shによる実行
   Hello.
   2000年04月24日 11時21分48秒

---

§6. 4 コマンドサーチパス

1. % echo $path　　⇒ コマンドサーチパスを表示
   この順にコマンドが検索される
   /home/vine/Bin ..... /usr/sbin

2. % mv hello ~/Bin　　⇒ helloを各自のBinディレクトリに移動

3. % hello
   hello: Command not found.

4. % rehash　　⇒ コマンドの存在を認識させる

5. % hello　　⇒ どこにいても~/Bin/helloを実行
   Hello.
   2000年04月24日 11時21分48秒

6. % ~/Bin/hello　　⇒ パス指定でも実行可能
   Hello.
   2000年04月24日 11時21分48秒

§6. 5 sh以外のシェル指定

- 先頭行が #!/bin/csh の場合，cshが解釈実行
- #!/bin/tcsh の場合，tcshが解釈実行

1．% vi ~/Bin/hello

2．dGi#!/bin/csh[Enter]
```
#!/bin/csh
■
~
```
3．echo "Hello. by" $shell[Enter]date[Esc]ZZ
```
#!/bin/csh
echo "Hello. by" $shell
date■
~
```
4．% hello
```
Hello. by /bin/tcsh（実体はtcsh）
2000年04月24日 11時21分58秒
```

§6. 6 csh用スクリプト

- 1章からのコマンドに加え，以下の記述が可能.
- csh用の記述はtcshでも動作するため，プロンプトに対する通常のコマンド入力にも応用できる.

- 引数渡し

1-1．コマンド名および引数1～9の取り出し
```
% vi test1
#!/bin/csh
echo $0, $1, $2, $3, $4, $5, $6, $7, $8, $9
echo $*
```

1-2．実行例
```
% chmod a+x test1
% ./test1 a b c d e f g h i j k
./test1, a, b, c, d, e, f, g, h, i
a b c d e f g h i j k
```

§6. 6 csh用スクリプト(続き)

▶ 変数への代入

2-1. 引数, 標準入力, ファイルを変数へ代入し加算

```
% vi test2
#!/bin/csh
set i1=$1                          ⇒ 引数1から
set i2=$<                          ⇒ 標準入力から
set i3=`cat counter`               ⇒ ファイルcounterから
@ i4=$i1 + $i2 + $i3
echo $i4 > counter                 ⇒ ファイルcounterへ
echo i1=$i1 i2=$i2 i3=$i3 i4=$i4
```

2-2. 実行例

```
% chmod a+x test2
% echo 10 > counter
% ./test2 1
2
i1=1 i2=2 i3=10 i4=13
```

§6. 6 csh用スクリプト(続き)

▶ 条件分岐

3-1. 標準入力の偶奇を表示

```
% vi test3
#!/bin/csh
echo -n "input number >"
set i=$<                           ⇒ 標準入力から
if ( $i % 2 == 0 ) then            ⇒ 2で割った余り
 echo "even"                       ⇒ 0の場合偶数
else
 echo "odd"                        ⇒ 0以外の場合奇数
endif
```

3-2. 実行例

```
% chmod a+x test3
% ./test3
input number >2
even
```

§6. 6 csh用スクリプト(続き)

▶ ループ

4-1．特定の文字が入力されるまで，入力データをファイルへ書き込む

```
% vi test4
#!/bin/csh
echo -n > data4                  ⇒ ファイルdata4を空にする
echo -n "input data >"           ⇒ プロンプトの表示
set i=$<                         ⇒ 標準入力から
while ( $i != e )                ⇒ 入力がe以外の間，続ける
 echo $i >> data4                ⇒ 入力をファイルdata4へ追加
 echo -n "input data >"          ⇒ プロンプトの表示
 set i=$<                        ⇒ 標準入力から
end
cat data4
```

4-2．実行例

```
% chmod a+x test4
% ./test4
input data >123
input data >efg
input data >e
123
efg
```

§6. 6 csh用スクリプト(続き)

▶ ループ

5-1．複数ファイルの処理

```
% vi test5
#!/bin/csh
foreach i ( test[1-3] )          ⇒ test1～test3が順に変数iに代入される
 sed "s/^/$i\: /" $i             ⇒ 行頭をtestX:に置き換える
 sleep 2                         ⇒ 2秒間停止する
end
```

5-2．実行例

```
% chmod a+x test5
% ./test5
test1: ...
test2: ...
test3: ...
```

注意: sedの引数が" "ではなく' 'の場合，行頭がファイル名ではなく$i:となる.
sedに限らず" "内部のシェル変数は展開され' '内部は展開されない.

```
$i: ...
$i: ...
$i: ...
```

§6. 7 今日の課題

講義中に作成したファイルtest5の実行結果をメールせよ.

宛先: nakashim@econ.kyoto-u.ac.jp
件名: unix1-学生番号

名前(忘れずに)

今日はここまで