Linux 程式設計 03/14/2017 心得筆記

1. 環境變數:

echo $SHELL -> 顯示 shell 的路徑

echo $SHLVL -> 目前的 shell 位於第幾層(盡量別用)

echo $$ -> 現在 shell 的 ps (程序)id

echo $RANDOM -> 隨機產生變數

```
e901@E901-VM:~/ch02$ echo $SHELL
/bin/bash
e901@E901-VM:~/ch02$ echo $SHLVL
1
e901@E901-VM:~/ch02$ echo $$
2572
e901@E901-VM:~/ch02$ echo $RANDOM
12951
```

補充 1 - 1: 現在安裝時系統管理員都將這些變數安裝好做為防呆措施，不必自己手動設定這些。

2-16～2-17 程式 “try_var”

```
#!/bin/sh
salutation="Hello"
echo $salutation
echo "The program $0 is now running"
echo "The second parameter was $2"
echo "The first parameter was $1"
echo "The parameter list was $*"
echo "The user's home directory is $HOME"
echo "Please enter a new greeting"
read salutation
echo $salutation echo "The script is now complete"
exit 0
```

上圖中 salutation="Hello"為設定區域變數，$0 為程式名稱，而$1，$2 皆為變數

最後的$*則可叫出全部的變數，可以下圖做為參考

```
e901@E901-VM:~/ch02$ ./try_var 11 22 33
Hello
The program ./try_var is now running
The second parameter was 22
The first parameter was 11
The parameter list was 11 22 33
The user's home directory is /home/e901
Please enter a new greeting
Siri
Siri
The script is now complete
```

補充 1 - 2: 在執行程式前可以看到上方的範例是有輸入變數的，但就算不輸入變數也可以進行程式，只是呼叫$1，$2 的部分為呼叫空值

Ex1-3-1.

```
e901@E901-VM:~/ch02$ ./try_var
Hello
The program ./try_var is now running
The second parameter was
The first parameter was
The parameter list was
The user's home directory is /home/e901
Please enter a new greeting
123
123
The script is now complete
```

補充 1 - 3: 將 echo "Please enter a new greeting" read salutation 這行改為

read –p “Please enter a new greeting : ” 可讓程式更為簡潔

也可加入 “–t +時間(秒)” 避免成是因為使用者不繼續進行而停滯

Ex1-3-2.

```
e901@E901-VM:~/ch02$ read -p "123:" xxx
123:111
e901@E901-VM:~/ch02$ echo $xxx
111
```

Ex1-3-3.

```
e901@E901-VM:~/ch02$ read -t 3 -p "1234 : " xxx
1234 : e901@E901-VM:~/ch02$
```

※ diff elif2 elif3 ->此程式藉由秀出不同的部分，可用於檢查兩程式的不同 ↓

```
e901@E901-VM:~/ch02$ diff elif2 elif3
3c3
< echo "Is it morning? Please answer yes or no"
---
> echo -n "Is it morning? Please answer yes or no: "
```

2. 條件判斷:

※補充 2-1 :以下舉例皆為 true 的場合 ※此段截錄製課本 2-19 頁 課本應該會更清楚

此單元介紹用於 SHELL 的條件判斷式，並分為下列三種:

字串比對:與一班使用方式相同分為 ” = “(字串是否相等)及” != “(字串是否不相等)

教 c 語言不同的是 ” -n ” (非空字串)及 ” –z ” (空字串)

數值比對:一定得使用英文代碼較特別外，還有提供 !expression 做為正反器的功能

檔案狀況:” -d ”(目標是否為目錄)、" -e "(目標是否存在 ※因為-e 沒有相容性、可攜性故通常使用-f)、” –f “(目標是否為一般檔案)、” –r “ (目標是否可讀)、” -s ”(目標大小是否非零)、” –w ” (目標是否可編輯)及”-x” (目標是否可執行)

3. 控制結構:

(1) If : 測試命令的結果，隨狀況執行不同的敘述

使用方式除了傳統的方式之外也可以用下列寫法 **if** [condition]; then

特別注意上面的敘述之前後是有空格的，那些是不能省略的，以及如果喜歡在 命令之後加上 then，務必在 then 的前面加上分號!!

※補充 3-2 -1： 指令成功執行時將回傳 0，反之亦然!

```
e901@E901-VM:~/ch02$ echo 123
123
e901@E901-VM:~/ch02$ echo $?
0
```

2 – 21 程式" elif * "

```
e901@E901-VM:~/ch02$ cat elif1
#!/bin/sh

echo "Is it morning? Please answer yes or no"
read timeofday

if [ $timeofday = "yes" ]
then
  echo "Good morning"
elif [ $timeofday = "no" ]; then
  echo "Good afternoon"
else
  echo "Sorry, $timeofday not recognized. Enter yes or no"
  exit 1
fi

exit 0
e901@E901-VM:~/ch02$ cat elif2
#!/bin/sh

echo "Is it morning? Please answer yes or no"
read timeofday

if [ "$timeofday" = "yes" ]
then
  echo "Good morning"
elif [ "$timeofday" = "no" ]; then
  echo "Good afternoon"
else
  echo "Sorry, $timeofday not recognized. Enter yes or no"
  exit 1
fi

exit 0
```

```
e901@E901-VM:~/ch02$ ./_if
Is it morning? Please answer yes or no

./_if: 6: [: =: unexpected operator
Good afternoon
```

上述的例子是在呼叫程式後對於程式的提問直接按 enter 輸入，因為 timeofday 這個值被輸入了 NULL 因此他無法辨認，但仔細觀察可發現程式連大小寫不同也被認為是錯誤的，因此課本提供了一個 **elif** 的指令，當第一個 **if** 失敗時就會執行 **elif** 當兩個都失敗時就執行 **else** 回傳錯誤訊息，且可避免 enter 後產生錯誤訊息

(2) for： 根據一些值進行迴圈，這些值也可以任何字串他們可以直接列程式中獲

shell 展開的檔名

※補充 3-4-1 :萬用字元的 for 迴圈，寫為

```
for file in $(ls f* . sh); do
echo $file
done
exit 0
```

此程式即可以用於尋找已 f 頭而黨名為 .sh 的檔案!

※補充 3-2-2 : seq 用法:

seq 1 5 ->會由 1 每次+1 直到 5

seq 1 3 10->會由 1 每次+3 直到 10

```
e901@E901-VM:~/ch02$ seq 1 5
1
2
3
4
5
e901@E901-VM:~/ch02$ for i in $(seq 1 5); do echo $i ; done
1
2
3
4
5
```

(3) while: 此指令最重要的用處就在執行不知道會需要迴圈多少次的狀況下

2-26 程式 "while1"

```
e901@E901-VM:~/ch02$ cat while1
#!/bin/sh

echo "Enter password"
read trythis

while [ "$trythis" != "secret" ]; do
  echo "Sorry, try again"
  read trythis
done
exit 0
```

```
e901@E901-VM:~/ch02$ ./while1
Enter password
1
Sorry, try again
2
Sorry, try again
3
Sorry, try again
secret
```

由上方程式可看到直到輸入正確的密碼前他都會無限的要求重新輸入!!

※補充 3-3-1 : idx=1;　idx =’ expr $idx +3’ ; echo $idx; 4

expr 為算數用命令

(4) until : 與 while 十分類似但判斷條件時機不同，一般，迴圈至少要執行一次，此時但如果迴圈不見得需要執行第一次，此時就要使用 until

```
e901@E901-VM:~/ch02$ cat _until
#!/bin/bash

until who | grep "$1" > /dev/null
do
  sleep 60
done

# Now ring the bell and announce the unexpected user.

echo -e '\a'
echo "***** $1 has just logged in *****"

exit 0
```

```
e901@E901-VM:~/ch02$ ./_until e901

***** e901 has just logged in *****
e901@E901-VM:~/ch02$ ./_until e123
```

※補充 3-4-1 :　”$1” > 成功則不執行

(5) case : 與 C++的 switch case 想法相同但中斷改為”;;”，雖然在本成是內因為只有一個樣本因此不必理會，但忽略 break 並不是一個好習慣，因此在這裡還是將其插入。

```
e901@E901-VM:~/ch02$ cat case3
#!/bin/sh

echo "Is it morning? Please answer yes or no"
read timeofday

case "$timeofday" in
    yes | y | Yes | YES )
           echo "Good Morning"
           echo "Up bright and early this morning?"
           ;;
    [nN]* )
           echo "Good Afternoon"
           ;;
    * )
           echo "Sorry, answer not recognised"
           echo "Please answer yes or no"
           exit 1
           ;;
esac

exit 0
```

※補充 3 -5- 1： 在課本的第 **2-30** 也有給更強力的判斷式

[yY]|[yY][eE][sS])

※補充 3-5-2： 如果程式名相同，則程式會使用 path 最先找到的

4.串列

(1)And :

```
e901@E901-VM:~/ch02$ ./and_list
hello
in else
e901@E901-VM:~/ch02$ cat and_list
#!/bin/sh

touch file_one
rm -f file_two

if [ -f file_one ] && echo "hello" && [ -f file_two ] && echo "there"
then
    echo "in if"
else
    echo "in else"
fi

exit 0
```

&&的判斷式是由左至右執行，而且只有所有條件都符合時才會執行 then 中的指令，，因此到 echo "hello"時程式直接執行畫面中就出現了 hello，接著到第三個判斷式時，找不到該檔案因此直接進入 else 的部分

※補充 4-1-1： 程式中的 touch 為檢查檔案是否存在，如果該檔案不存在就建立一個新檔案的指令 而 rm –f 則是檢查檔案是否存在，如果該檔案存在就將其刪除的指令

(2)Or :

```
e901@E901-VM:~/ch02$ ./or_list
hello
in if
e901@E901-VM:~/ch02$ cat or_list
#!/bin/sh

rm -f file_one

if [ -f file_one ] || echo "hello" || echo "there"
then
    echo "in if"
else
    echo "in else"
fi

exit 0
```

||的判斷式一樣是由左至右執行，而且只要一有條件符合時就會執行 then 中的指令，因此到 echo "hello"時程式直接執行畫面中就出現了 hello， 並在同時直接進入 then 的敘述式中

4. 加密(openssl) :

```
e901@E901-VM:~/ch02$ openssl ?
openssl:Error: '?' is an invalid command.

Standard commands
asn1parse         ca                ciphers           cms
crl               crl2pkcs7         dgst              dh
dhparam           dsa               dsaparam          ec
ecparam           enc               engine            errstr
gendh             gendsa            genpkey           genrsa
nseq              ocsp              passwd            pkcs12
pkcs7             pkcs8             pkey              pkeyparam
pkeyutl           prime             rand              req
rsa               rsautl            s_client          s_server
s_time            sess_id           smime             speed
spkac             srp               ts                verify
version           x509

Message Digest commands (see the `dgst' command for more details)
md4               md5               rmd160            sha
sha1

Cipher commands (see the `enc' command for more details)
aes-128-cbc       aes-128-ecb       aes-192-cbc       aes-192-ecb
aes-256-cbc       aes-256-ecb       base64            bf
bf-cbc            bf-cfb            bf-ecb            bf-ofb
camellia-128-cbc  camellia-128-ecb  camellia-192-cbc  camellia-192-ecb
camellia-256-cbc  camellia-256-ecb  cast              cast-cbc
cast5-cbc         cast5-cfb         cast5-ecb         cast5-ofb
des               des-cbc           des-cfb           des-ecb
des-ede           des-ede-cbc       des-ede-cfb       des-ede-ofb
des-ede3          des-ede3-cbc      des-ede3-cfb      des-ede3-ofb
des-ofb           des3              desx              rc2
rc2-40-cbc        rc2-64-cbc        rc2-cbc           rc2-cfb
rc2-ecb           rc2-ofb           rc4               rc4-40
seed              seed-cbc          seed-cfb          seed-ecb
seed-ofb
```

關於數位指紋，目前較多使用的仍是 md5 及 sha1。

◎md5 使用 ： 假設目前要得到 xxx 的 md5 只需輸入 **openssl md5 xxx**，接著系統就會出現 **MD5(xxx): ..～～～..**，在此也可以使用剛剛的 for+seq 來 md5 如下圖

```
e901@E901-VM:~$ for i in $( seq 1 5 ); do touch $i |openssl md5 $i ; done
MD5(1)= d41d8cd98f00b204e9800998ecf8427e
MD5(2)= d41d8cd98f00b204e9800998ecf8427e
MD5(3)= d41d8cd98f00b204e9800998ecf8427e
MD5(4)= d41d8cd98f00b204e9800998ecf8427e
MD5(5)= d41d8cd98f00b204e9800998ecf8427e
```

※補充 4-1 : 使用 cut -c 10-15 即可將第 10 個字到第 15 個字剪下，是一個方便的功能

也可以使用 des 將其加密，如下圖

```
e901@E901-VM:~$ openssl des -in xxx -out xxx.dex
enter des-cbc encryption password:
Verifying - enter des-cbc encryption password:
e901@E901-VM:~$ cat xxx
1231321321
e901@E901-VM:~$ cat xxx.dex
e901@E901-VM:~$ openssl des -d -in xxx.dex -out xxx.ans
enter des-cbc decryption password:
e901@E901-VM:~$ openssl md5 xxx
MD5(xxx)= 6ca42950d56fb3525c942fdb055e9f30
e901@E901-VM:~$ openssl md5 xxx.ans
MD5(xxx.ans)= 6ca42950d56fb3525c942fdb055e9f30
```