經濟部資訊專業人員鑑定—開放式系統類

Linux進階系統管理
Shell scripts

崑山科技大學資訊傳播系
蔡德明
(鳥哥, VBird)

分享指引

- BASH複習
- shell script
- 精選範例

BASH複習

萬用字元的支援

- bash 的萬用字元功能：
  - * 0~無窮多個任意字元
  - ? 一定有一個任意字元
  - [0-9] 有一個在中刮號內的字元存在
  - [^abc] 有一個不在中刮號內的字元存在
  - 可 man grep 搜尋 \[: 找出特殊字浮功能
    - [:upper:], [:lower:]...
    - 可略過語系的問題

Bash的變數

- 變數的設定方式：
  - 變數名稱=“變數內容”
- 變數設定規則
  - 變數與變數內容以等號『=』連結，且等號兩邊不能直接接空白字元
  - 變數名稱只能是英文字母與數字，且數字不能是開頭字元；
  - 可以使用雙引號『“』或單引號『‘』來將變數內容結合起來
    - 雙引號內的特殊字元可以保有變數特性，
    - 單引號內的特殊字元則僅爲一般字元；
  - 跳脫字元『\』來可特殊符號 變成一般字元；
  - 指令內的指令可用『`command`』或『$(command)』
  - 可以 export 來使變數變成環境變數， 如『export PATH』；
  - 取消變數的方法爲：『unset 變數名稱』。

變數的呼叫/使用

- 變數的呼叫：
  - echo $var
  - echo ${var}
- 變數的使用：
  - mkdir '~dmtsai' →建立名爲 ~dmtsai 的目錄
  - echo "$PATH" →叫出PATH變數的內容
  - kversion=$(uname –r) →設定kversion爲核心版本
  - echo "\$PATH" →顯示 $PATH 在螢幕上
  - set →顯示目前所有的變數

亂數產生器

- BASH當中的特殊變數
  - RANDOM
    - 值介於0 ~ 32767
  - 隨機產生 0-100的亂數
    - echo $(($RANDOM*100/32727))
- 亂數產生器
  - /dev/random
  - /dev/urandom

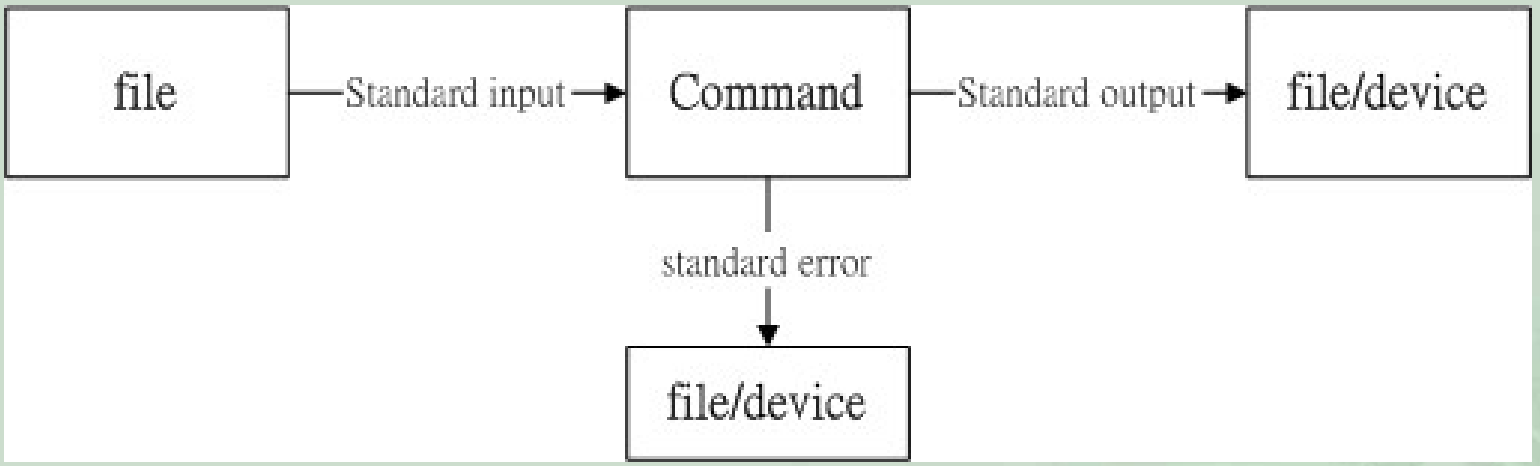
命令別名與歷史命令

- 命令別名
  - 透過 alias newcmd='oldcmd options'
  - 取消使用 unalias newcmd
  - 可載入 ~/.bashrc 讓每次都生效
- 歷史命令
  - history
  - 透過 !n, !!, !cmd 等方式重複執行某指令
  - HISTSIZE可控制筆數
  - ~/.bash_history為記錄的檔案

指令執行的順序

- 指令執行的順序：
  - 絕對/相對路徑直接執行指令
  - alias 所建立的別名
  - bash 內建的指令
  - PATH 所找到的指令
- 控制指令執行的細項：
  - cmd1 ; cmd2　　兩者間沒有關係，依序執行
  - cmd1 && cmd2　　cmd1成功，cmd2才會執行
  - cmd1 || cmd2　　cmd1失敗，cmd2才會執行

資料流重導向

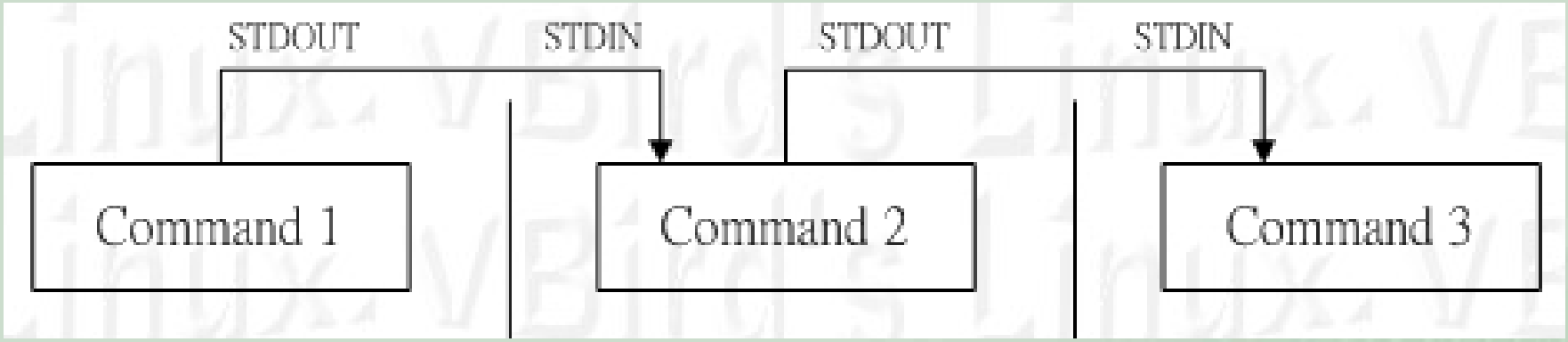


STDOUT： >(覆蓋) >>(累加)

STDERR： 2>(覆蓋) 2>>(累加)

全部的輸出： &> command > something 2>&1

管線命令



- 前一個指令的 STDOUT 作爲後一個指令的 STDIN 之意。(STDERR沒有作用)

正規表示法

- 常用的正規表示法(基礎正規表示法)：利用一些字元符號(字符)來作爲關鍵字的一種表示方式
  - ^ ：一行的行首
  - $ ：一行的行尾
  - [a-b] ：一定有一個，視爲集合
  - [^a-b] ：反向集合
  - * ：重複前一個 0 ~ 無窮多次
  - . ：一定有一個任意字元
  - .* ：0~無窮多個任意字元。

利用test判斷

- 關於檔名的『類型』偵測(存在與否)，如 test -e filename
  - -e　　該『檔名』是否存在？
  - -f　　該『檔名』是否爲檔案(file)？
  - -d　　該『檔名』是否爲目錄(directory)？
- 關於檔案的權限偵測，如 test -r filename
  - -r　　偵測該檔名是否具有『可讀』的屬性？
  - -w　　偵測該檔名是否具有『可寫』的屬性？
  - -x　　偵測該檔名是否具有『可執行』的屬性？
- 兩個檔案之間的比較，如： test file1 -nt file2
  - -nt　　(newer than)判斷 file1 是否比 file2 新
  - -ot　　(older than)判斷 file1 是否比 file2 舊

利用test判斷(續)

- 關於兩個整數之間的判定，例如 test n1 -eq n2
  - -eq　　兩數值相等 (equal)
  - -ne　　兩數值不等 (not equal)
  - -gt　　n1 大於 n2 (greater than)
  - -lt　　n1 小於 n2 (less than)
  - -ge　　n1 大於等於 n2 (greater than or equal)
  - -le　　n1 小於等於 n2 (less than or equal)
- 判定字串的資料
  - test -z string　　字串是否爲空, 若 string 爲空,則 true
  - test -n string　　字串是否非爲空,若 string 爲空,則 false。

Shell scripts

Shell script

- 什麼是 shell script
  - 將許多指令寫入一個批次檔，藉此快速執行
  - 透過 shell 的語法，可以設定程式 (if, then…)
  - 就是純文字檔囉
  - 需具有 r 或者是 r+x 才能夠被執行
- 爲何需要學習 shell script
  - 很多的指令可以整合在一起，然後讓系統自動執行
  - 可以撰寫讓系統自動管理及自動進行錯誤偵測(troubleshooting)的任務
  - 可以用之以建立簡單的應用程式(如鳥哥自己的 logfile.sh 程式)

撰寫習慣與執行

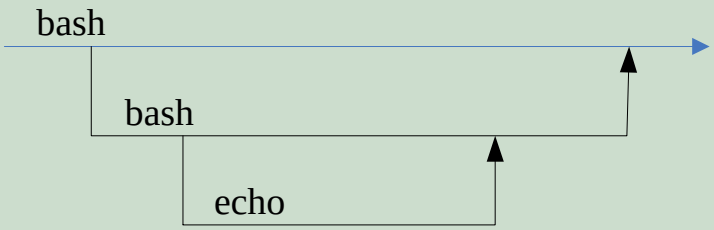
- 撰寫習慣之建立：
  - 第一行請宣告 shell ，例如 #!/bin/bash
  - 第二行說明程式功能、授權、作者、此程式的版本等等
  - 建議常使用變數來處理複雜的資料
  - 常用『 #註解內容』來說明比較難懂得程式碼部分。
- 如何執行？
  - /root/script1.sh　　需具有 r 與 x
  - cd /root; ./script1.sh　　需具有 r 與 x
  - sh /root/script1.sh　　具有 r 即可
  - 將 script1.sh 放入 $PATH 中　　需具有 r 與 x

第一支script

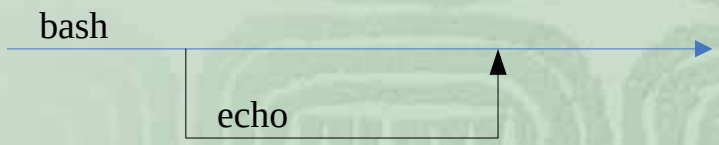
- 第一支shell script
  - #!/bin/bash
  - # This script displays some information about your system
  - # write by VBird 2007/11/17
  - echo “Hello world!”
  - echo “The day is $(date)”
  - echo “Your working directory is: $(pwd)”
  - exit 0

執行方式與結果

- 新增bash程序來處理
  - ./script.sh
  - sh script.sh
  - 與原本的bash環境無關



- 原bash環境下處理
  - source script.sh
  - . script.sh
  - 在原本的bash中執行



鍵盤直接輸入變數 read

- 作中學一：利用 read 這個指令，讓使用者連續輸入first name與last name，最終組合起來顯示出來

```
#!/bin/bash
read -p "Please input your first name: " firstname
read -p "Please input your last name:  " lastname
echo -e "\nYour full name is: $firstname $lastname"
```

外加指令$(cmd)與`cmd`的功能

- 作中學二：透過 date 來產生與日期有關的檔案。

```
#!/bin/bash
echo -e "I will use 'touch' command to create 3 files."
read -p "Please input the filename what you want: " fileuser
date1=`date --date='2 days ago' +%Y%m%d`
date2=`date --date='1 days ago' +%Y%m%d`
date3=`date +%Y%m%d`
file1="$filename""$date1"
file2="$filename""$date2"
file3="$filename""$date3"
touch {$file1,$file2,$file3}
```

數學運算功能

- 作中學三：讓使用者自己輸入數字，好進行乘法！

```
#!/bin/bash
echo -e "You SHOULD input 2 number, I will cross them! \n"
read -p "first number:  " firstnu
read -p "second number: " secnu
total=$(($firstnu*$secnu))
echo -e "\nThe number $firstnu x $secnu is ==> $total"
```

條件判斷 if…then…fi

- 作中學四：利用if…then的條件判斷式來處理重要任務。例如判斷使用者輸入的是 yes 或 no：

```
#!/bin/bash
read -p "Please input (Y/N): " yn
if [ "$yn" == "Y" -o "$yn" == "y" ]; then
        echo "OK, continue"
        exit 0
elif [ "$yn" == "N" -o "$yn" == "n" ]; then
        echo "Oh, interrupt!"
        exit 0
else
        echo "I don't know what is your choice" && exit 0
fi
```

script的內建變數

- Shell script的內建變數(數字表示)：

/path/to/scriptname opt1 opt2 opt3 opt4 ..
$0 $1 $2 $3 $4 ..

  - $* ：代表所有的後續參數，亦即『$1 $2 $3 $4…』
  - $@ ：代表所有參數總和，亦即『“$1 $2 $3 $4..”』
  - $# ：代表總共有幾個參數，例如 3 個或 4 個等等。
  - 可被 shift 這個指令挪動 opt 的號碼
    - ex> shift 2 → $1=opt3 (原本 $1=opt1)

script後接參數的運用

- 作中學五：利用 script 後接的參數來處理：

  - #!/bin/bash
  - echo "The script name is ==> **$0**"
  - echo “There are total $# arguments used in this script”
  - [-n "**$1**"] && echo "The 1st paramter is ==> $1" || exit 0
  - [-n "**$2**"] && echo "The 2nd paramter is ==> $2" || exit 0
  - [-n "**$3**"] && echo "The 3th paramter is ==> $3" || exit 0

    - 執行：ex> ./script.sh one two three

利用特定目標 case … esac

- 作中學六：利用『case ….esac』來進行特定參數的定義：

```
#!/bin/bash
case $1 in
  "hello")
            echo "Hello, how are you ?"
            ;;
  "")
            echo "You MUST input parameters, ex> $0 someword"
            ;;
  *)
            echo "Usage $0 {hello}"
            ;;
esac
```

固定數量迴圈for do done

- 作中學七：使用迴圈 part I：利用 for do done處理固定次數的迴圈動作

```
#!/bin/bash
for animal in dog cat elephant
do
        echo "There are ""$animal""s.... "
done
```

不固定數量迴圈while, until

- 作中學八：使用迴圈 part II：利用條件判斷來處理不特定的迴圈次數

```
#!/bin/bash
while [ "$yn" != "yes" ] && [ "$yn" != "YES" ]
do
        read -p "Please input yes/YES to stop this program: " yn
done
```

精選範例

- script中的 $RANDOM有何作用？ C
  - (A) 讀取亂數
  - (B) 傳遞亂數
  - (C) 產生亂數
  - (D) 無任何作用

- 請問如何使你的Linux可有路由的功能？C
  - (A) echo “1” > /proc/net/sys/ipv4/ip_forward
  - (B) echo “0” > /proc/net/sys/ipv4/ip_forward
  - (C) echo “1” > /proc/sys/net/ipv4/ip_forward
  - (D) echo “0” > /proc/sys/net/ipv4/ip_forward

- 執行 shell 時，用下列那個方法可以顯示每個執行步驟的回傳值？ AB
  - (A) sh –v test.sh
  - (B) sh –x test.sh
  - (C) sh –c test.sh
  - (D) sh –f test.hs

- 下列應用中，變數設定其傳回值何者正確？『now=`date`』 A
  - (A) 四 6 月 3 17:35:07 CTS 2004
  - (B) date
  - (C) $now
  - (D) 1

- 要寫作一支 script 程式時，程式的第一行應該要如何撰寫？B
  - (A) #/bin/bash
  - (B) #!/bin/bash
  - (C) #$/bin/bash
  - (D) #%/bin/bash

- 若有程式 s.sh 的內容如右所示
  當執行 ./s.sh a b c d e f g，輸出結果爲？C
  - (A) a b c d e f g; f g
  - (B) a b c d e f g; e f g
  - (C) b c d e f g; f g
  - (D) b c d e f g; e f g

```
#!/bin/bash
shift
echo $@
shift 4
echo $@
```

- 若有程式 ev.sh 如右所示，
  執行 ./ev.sh a b c 的結果爲？B
  - (A) c is $3
  - (B) $3 is c
  - (C) \$ is \$
  - (D) \$$# is \$$#

```
#!/bin/bash
echo –n \$$#
echo –n “ is “
eval echo \$$#
```

- 如下關於變數的定義，哪些是不正確的？ ACD
  - (A) a1=b c
  - (B) a1=“b c”
  - (C) a1 = “b c”
  - (D) 1a=“b c”

- 假設使用者寫了一支script名爲 foobar.sh ，其內容爲『#!/bin/bash; cd /tmp』並放在 ~/bin/ 內，然後使用者在家目錄執行該script，當script結束時，使用者的工作目錄會在？A
  - (A) ~/
  - (B) ~/bin
  - (C) /tmp
  - (D) /

- 假設我在目前目錄撰寫 test.sh ，但是我並沒有建立 PATH，則該如何下達指令執行？ BCD
  - (A) test.sh
  - (B) ./test.sh
  - (C) sh test.sh
  - (D) exec test.sh

- 有關 shell script的 case 用法，何者有誤？AB
  - (A) 以『end case』作爲指令敘述的結尾
  - (B) 以『break』作爲條件區塊的結束
  - (C) 『*』加上右刮號『*)』代表所有條件都不符合時，則執行其後的敘述
  - (D) 可使用參數『 $1』 接在執行檔案的後面執行之

- 一個shell script叫做 foo，foo能被執行的先決條件有哪些？BC
  - (A) 爲了安全考量foo可以有執行的權限開放，但不需要有讀的權限
  - (B) 如果要使命令 ./foo 能夠被所指定的shell所執行的話，foo中必須要有『#!』的符號指示
  - (C) foo必須要有可讀與執行的權限開放給欲執行foo的使用者
  - (D) shell script一定要用bash來執行

- 請問下列何者爲bash的判別是用法？ BD
  - (A) if … elseif … fi
  - (B) if … then … elif … then … else … fi
  - (C) if … then … elseif … fi
  - (D) if … then … else … fi

- 請問下列何者爲 bash 使用迴圈的命令 ABD
  - (A) while
  - (B) until
  - (C) foreach
  - (D) for