經濟部資訊專業人員鑑定—開放式系統類

Linux 基礎運作—BASH shell

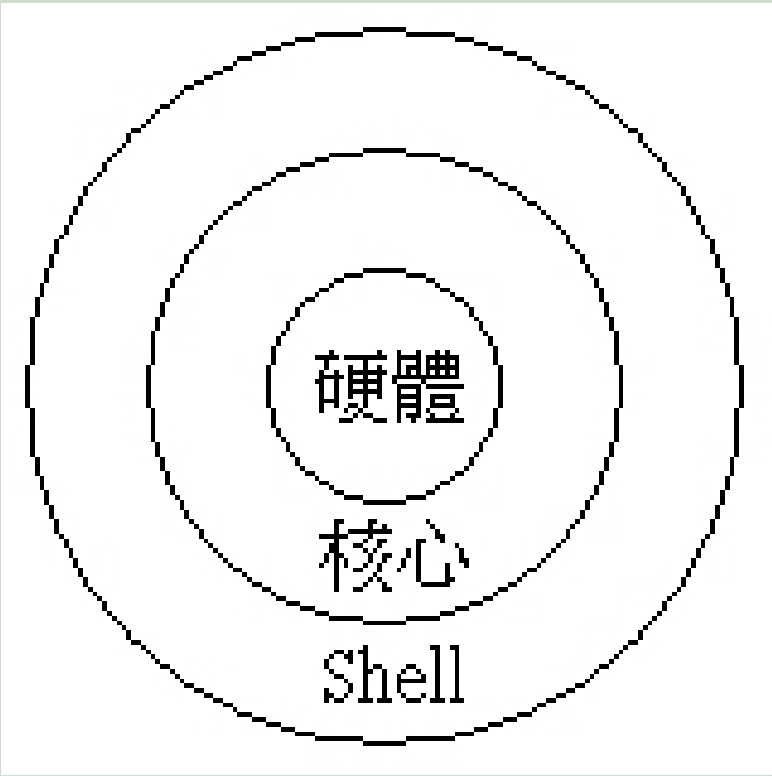
崑山科技大學資訊傳播系
蔡德明
(鳥哥, VBird)

分享指引

- Bash Shell
- 查閱檔案內容指令
- vi 與 vim 程式編輯器
- 資料流重導向
- 管線命令
- 檔案/指令搜尋
- 正規表示法
- 工作管理(job control)
- 精選範例

Bash shell

Shell 的角色



- 使用者可以透過 shell 直接控制核心，以達成各項任務

Linux shell

- Linux 合法的 shell
  - /etc/shells
    - /bin/sh
    - /bin/bash
    - /sbin/nologin
    - /bin/ksh
    - /usr/bin/ksh
    - /bin/tcsh
    - /bin/csh
    - /bin/zsh
- 預設的 shell
  - /bin/bash
- 使用者修改預設shell
  - chsh –s ‘shellname’

bash 的功能

- 命令編修能力（類似 DOS 的 doskey 功能）
- 命令與檔案補全功能：
- 命令別名(alias)設定功能
- 工作控制(jobs)、前景背景控制：
- Shell scripts 的強大功能
- 萬用字元！

bash 的慣用按鍵

- [Tab] 按鍵
  - [Tab] 接在一串指令的第一個字的後面，則爲命令補全；
  - [Tab] 接在一串指令的第二個字以後時，則爲『檔案補齊』！
- [Ctrl]-c 組合鍵
  - 可以中斷目前正在執行中的程式
- [Ctrl]-d 組合鍵
  - 結束某些程式所需的輸入資訊
- [Shift]-[Pageup]/[Shift]-[Pagedown]
  - 在終端機模式下，向前/向後翻頁

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『=』來連結，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

影響bash操作環境的變數

- 幾個較重要的，會影響環境的變數
  - HOME　家目錄，即 ~ 所代表的目錄
  - MAIL　輸入 mail 即可收信(信箱)
  - HISTSIZE　history 的紀錄比數
  - LANG　語系資料 (可用 locale –a)
  - PATH　指令執行查詢目錄順序
  - PS1　命令提示字元(man bash)
  - $　此 shell 的 PID 號碼
  - ?　上個指令執行回傳值 (0為正確)

變數的有效範圍

- 環境變數
  - 當啓動一個 shell ，作業系統分配一記憶區塊給 shell 使用，此區域之變數可以讓子程序存取；
  - 利用 export 功能，可以讓變數的內容寫到上述的記憶區塊當中(環境變數)；
  - 當載入另一個 shell 時 (亦即啓動子程序，而離開原本的父程序了)，子 shell 可以將父 shell 的環境變數所在的記憶區塊導入自己的環境變數區塊當中。

bash 的內建指令

- bash 本身有提供一些內建指令
  - cd, pwd
  - echo
  - jobs, fg, bg
  - history
  - type
  - …
  - man cd (可以看到很多喔！)

歷史命令

- 呼叫過去下達過的指令
  - history [-n]
- history 的紀錄檔
  - ~/.bash_history
- 指令的應用
  - !!
  - !vi
  - !50
  - !-5

命令別名

- 建立簡單好用的新指令
  - alias ll='ls –al'
  - alias h=history
- 取消的方式
  - unalias h

指令執行的順序

- 若於多個地方擁有相同指令，如 ls, echo
  - 絕對路徑/相對路徑直接執行某程式
  - 命令別名所載 (alias)
  - bash 內建指令
  - 由 PATH 所找到的指令
  - 可用 type –a 指令 來檢查！

環境參數設定檔

- login-shell：登入時會讀取的設定檔
  - /etc/profile
  - ~/.bash_profile, ~/.bash_login, ~/.profile
- non-login-shell：非登入時所取得 bash 的環境
  - 例如 X 畫面下的終端機
  - 在 bash 中執行 bash
  - 執行 script 時
  - ~/.bashrc
- 不登出立刻讓設定檔生效的方法
  - . ~/.bashrc
  - source ~/.bashrc

萬用字元

- 常見的bash環境萬用字元
  - * 0到無窮多個任意字元
  - ? 一個任意字元
  - [a-c] 一個在中刮號中的字元存在
  - [^a-c] 一個不在中刮號中的字元存在
- 一些範例
  - 具有3個字母的檔案： /etc/???
  - 具有數字的檔名： /etc/*[0-9]*
  - 大寫字元的檔案： /etc/*[[:upper:]]*

身份切換

- bash 環境的操作
  - 盡量不要使用 root 身份，以免不小心影響系統
  - 一般使用者想要切換身份，可用 su –
    - 轉身份成爲root： su – (然後輸入root密碼)
  - 離開 su - 的環境，使用exit來回到原本的身份
  - 切換成爲其他使用者時
    - su – username
    - 需要輸入該使用者的密碼才行
    - root 變身成爲他人，不需要輸入密碼

查閱檔案內容指令

常用來查詢檔案內容的指令

- cat　　由第一行開始顯示檔案內容
- tac　　從最後一行開始顯示
- nl　　顯示的時候，順道輸出行號！
- more　　一頁一頁的顯示檔案內容
- less　　與 more 類似，且可以往前翻頁！
- head　　只看頭幾行
- tail　　只看尾巴幾行
- od　　以二進位的方式讀取檔案內容

cat 與 nl

```
[root@linux ~]# cat [-AEnTv]
參數：
-A   ：相當於 -vET 的整合參數，可列出一些特殊字符~~
-E   ：將結尾的斷行字元 $ 顯示出來；
-n   ：列印出行號；
-T   ：將 [tab] 按鍵以 ^I 顯示出來；
-v   ：列出一些看不出來的特殊字符
```

```
[root@linux ~]# nl [-bnw] 檔案
參數：
-b   ：指定行號指定的方式，主要有兩種：
       -b a  ：表示不論是否為空行，也同樣列出行號；
       -b t  ：如果有空行，空的那一行不要列出行號；
-n   ：列出行號表示的方法，主要有三種：
       -n ln  ：行號在螢幕的最左方顯示；
       -n rn  ：行號在自己欄位的最右方顯示，且不加 0 ；
       -n rz  ：行號在自己欄位的最右方顯示，且加 0 ；
-w   ：行號欄位的佔用的位元數。
```

head 與 tail

```
[root@linux ~]# head [-n number] 檔案
參數：
-n  ：後面接數字，代表顯示幾行的意思
範例：
[root@linux ~]# head /etc/man.config
# 預設的情況中，顯示前面十行！若要顯示前 20 行，就得要這樣：

[root@linux ~]# head -n 20 /etc/man.config
```

```
[root@linux ~]# tail [-n number] 檔案
參數：
-n  ：後面接數字，代表顯示幾行的意思
範例：
[root@linux ~]# tail /etc/man.config
# 預設的情況中，顯示最後的十行！若要顯示最後的 20 行，就得要這樣：

[root@linux ~]# tail -n 20 /etc/man.config
```

tail +5 /etc/man.config → 第五行以後的資料通通印出來
tail –f /var/log/messages→持續追蹤該檔案的內容

查詢檔案屬性

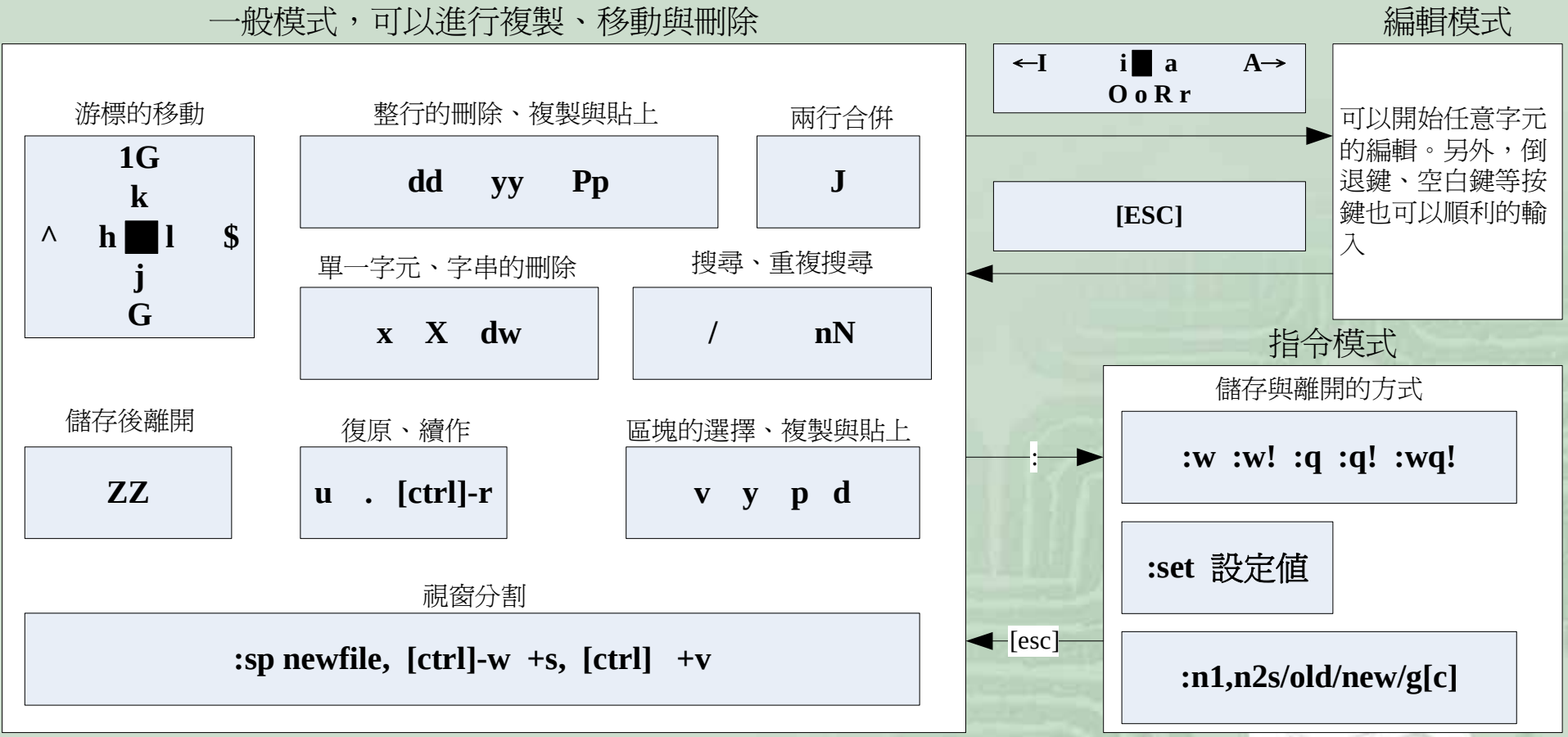
```
[root@linux ~]# ls [-aAdfFhilRS] 目錄名稱
[root@linux ~]# ls [--color={none,auto,always}] 目錄名稱
[root@linux ~]# ls [--full-time] 目錄名稱
參數：
-a   ：全部的檔案，連同隱藏檔( 開頭為 . 的檔案) 一起列出來~
-A   ：全部的檔案，連同隱藏檔，但不包括 . 與 .. 這兩個目錄，一起列出來~
-d   ：僅列出目錄本身，而不是列出目錄內的檔案資料
-f   ：直接列出結果，而不進行排序 (ls 預設會以檔名排序！)
-F   ：根據檔案、目錄等資訊，給予附加資料結構，例如：
       *：代表可執行檔； /：代表目錄； =：代表 socket 檔案； |：代表 FIFO 檔案；
-h   ：將檔案容量以人類較易讀的方式(例如 GB, KB 等等)列出來；
-i   ：列出 inode 位置，而非列出檔案屬性；
-l   ：長資料串列出，包含檔案的屬性等等資料；
-n   ：列出 UID 與 GID 而非使用者與群組的名稱 (UID與GID會在帳號管理提到！)
-r   ：將排序結果反向輸出，例如：原本檔名由小到大，反向則為由大到小；
-R   ：連同子目錄內容一起列出來；
-S   ：以檔案容量大小排序！
-t   ：依時間排序
```

vi 與 vim 程式編輯器

vi 是什麼

- vi 與 vim
  - vi 是一個文書編輯器，在各主要 Unix like 系統中均存在
  - vi 會被其他軟體所呼叫，例如 crontab
  - vim 是加強版的 vi ，可以具有顏色顯示、語法校驗等功能
  - vim 應該可被稱爲程式編輯器

vi 的慣用按鈕


一般模式，可以進行複製、移動與刪除
編輯模式
游標的移動
1G
k
^ h l $
j
G
整行的刪除、複製與貼上
dd yy Pp
兩行合併
J
←I i a A→
O o R r
可以開始任意字元的編輯。另外，倒退鍵、空白鍵等按鍵也可以順利的輸入
[ESC]
單一字元、字串的刪除
x X dw
搜尋、重複搜尋
/ nN
指令模式
儲存與離開的方式
儲存後離開
ZZ
復原、續作
u . [ctrl]-r
區塊的選擇、複製與貼上
v y p d
:
:w :w! :q :q! :wq!
:set 設定值
視窗分割
:sp newfile, [ctrl]-w +s, [ctrl] +v
[esc]
:n1,n2s/old/new/g[c]

vim的環境設定

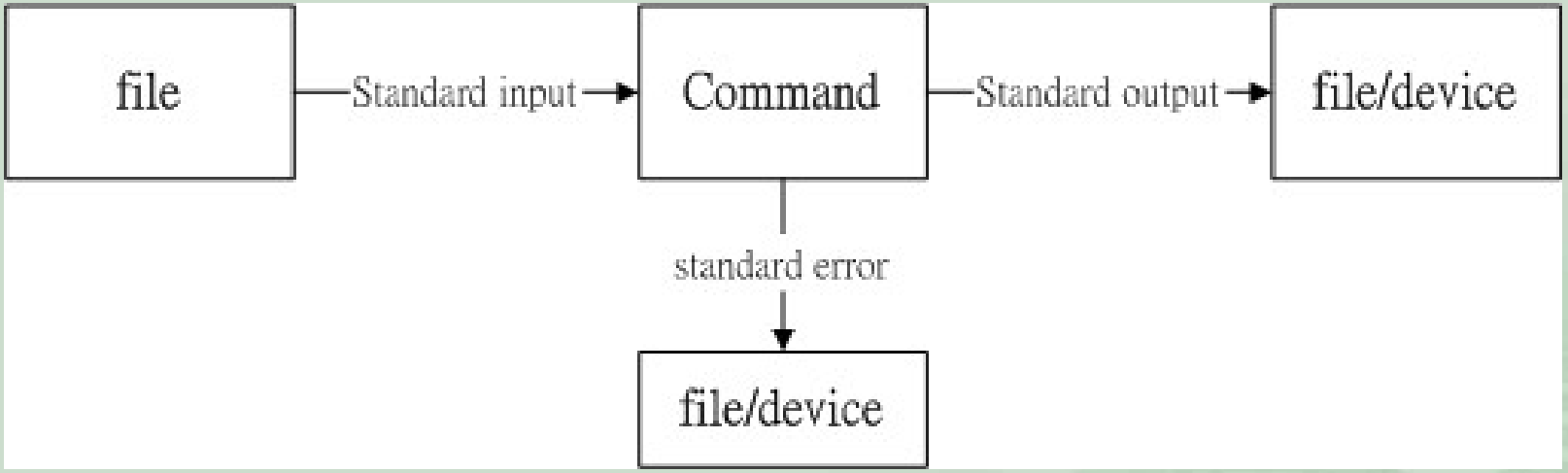
- vim 尚有非常多的設定資訊，包括有：
  - :set nu (行號)
  - :set autoindent(縮排)
  - :set textwidth=80(行寬)
  - :set hlsearch(高亮度反白)
  - :syntax {on|off}(語法的正確性與否檢驗)
- 各項目可寫入設定檔，亦即： ~/.vimrc

資料流重導向

指令的訊息

- 每個指令的執行結果可能都會有輸出的資料
  - 正確的資料：Standard Output (STDOUT)
  - 錯誤的資料：Standard Error Output(STDERR)
- 指令在運作時，可能需要讀入資料
  - 輸入的資料：Standard Input
  - 可能由檔案，或者是鍵盤輸入而來。

訊息的顯示方式



- STDOUT 與 STDERR 預設都輸出至螢幕上
  - >, >>　可將STDOUT轉傳到其他檔案/裝置
  - 2>, 2>>　可將STDERR轉傳到其他檔案/裝置
  - <　可代表讀入的資料。

一個範例

```
[dmtsai@linux ~]$ find /home -name testing
find: /home/test1: Permission denied    <== Starndard error
find: /home/root: Permission denied     <== Starndard error
find: /home/masda: Permission denied    <== Starndard error
/home/dmtsai/testing                    <== Starndard output
```

```
[dmtsai@linux ~]$ find /home -name testing > list_right 2> list_error
```

- 透過 > 與 2> 將原本由螢幕輸出的資料分別轉送到 list_right 與 list_error 當中。
- 螢幕不會有任何訊息的產生

特殊寫法

```
[dmtsai@linux ~]$ find /home -name testing > list_right 2> /dev/null
```

```
[dmtsai@linux ~]$ find /home -name testing > list 2> list  <==錯誤寫法
[dmtsai@linux ~]$ find /home -name testing > list 2>&1     <==正確寫法
```

- 可用垃圾桶 (/dev/null) 去除不要的資訊
- 可用 2>&1 將所有訊息導向同一個檔案

結束輸入關鍵字

```
[root@linux ~]# cat > catfile <<eof
> This is a test testing
> OK now stop
> eof  <==輸入這個玩意兒，嘿！立刻就結束了！
```

- 透過 <<keyword 來結束鍵盤的輸入

資料流重導向的使用時機

- 當螢幕輸出的資訊很重要，而且我們需要將他存下來的時候；
- 背景執行中的程式，不希望他干擾螢幕正常的輸出結果時；
- 一些系統的例行命令（例如寫在 /etc/crontab 中的檔案）的執行結果，希望他可以存下來時；
- 一些執行命令，我們已經知道他可能的錯誤訊息，所以想以『 2> /dev/null 』將他丟掉時；
- 錯誤訊息與正確訊息需要分別輸出時。

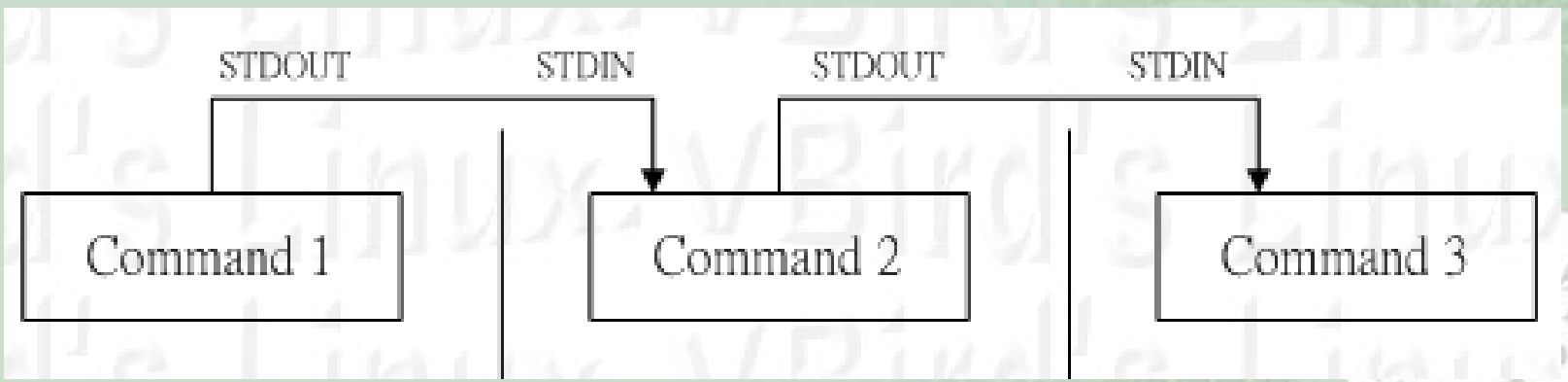
連續指令的下達

- 逐次執行指令
  - cmd1 ; cmd2 ; cmd3
- 前一個指令回傳值爲0後面才執行
  - cmd1 && cmd2
- 前一個指令回傳值非爲0後面就執行
  - cmd1 || cmd2
- 綜合處理
  - cmd1 && cmd2 || cmd3

管線命令

管線命令

- 管線命令的意義
  - 可以處理來自前一個指令的STDOUT
  - 不處理STDERR的資訊
  - cat, more, less都是管線命令
  - ls, cp…並非管線命令



cut

```
[root@linux ~]# cut -d'分隔字元' -f fields
[root@linux ~]# cut -c 字元區間
參數：
-d  ：後面接分隔字元。與 -f 一起使用；
-f  ：依據 -d 的分隔字元將一段訊息分割成為數段，用 -f 取出第幾段的意思；
-c  ：以字元 (characters) 的單位取出固定字元區間；
範例：

範例一：將 PATH 變數取出，我要找出第五個路徑。
[root@linux ~]# echo $PATH
/bin:/usr/bin:/sbin:/usr/sbin:/usr/local/bin:/usr/X11R6/bin:/usr/games:
[root@linux ~]# echo $PATH | cut -d ':' -f 5
# 嘿嘿！如此一來，就會出現 /usr/local/bin 這個目錄名稱！
# 因為我們是以 : 作為分隔符號，第五個就是 /usr/local/bin 啊！
# 那麼如果想要列出第 3 與第 5 呢？，就是這樣：
[root@linux ~]# echo $PATH | cut -d ':' -f 3,5
```

擷取字元 grep

```
[root@linux ~]# grep [-acinv] '搜尋字串' filename
參數：
-a ：將 binary 檔案以 text 檔案的方式搜尋資料
-c ：計算找到 '搜尋字串' 的次數
-i ：忽略大小寫的不同，所以大小寫視為相同
-n ：順便輸出行號
-v ：反向選擇，亦即顯示出沒有 '搜尋字串' 內容的那一行！
範例：

範例一：將 last 當中，有出現 root 的那一行就取出來；
[root@linux ~]# last | grep 'root'

範例二：與範例一相反，只要沒有 root 的就取出！
[root@linux ~]# last | grep -v 'root'

範例三：在 last 的輸出訊息中，只要有 root 就取出，並且僅取第一欄
[root@linux ~]# last | grep 'root' |cut -d ' ' -f1
# 在取出 root 之後，利用上個指令 cut 的處理，就能夠僅取得第一欄囉！
```

排序 sort

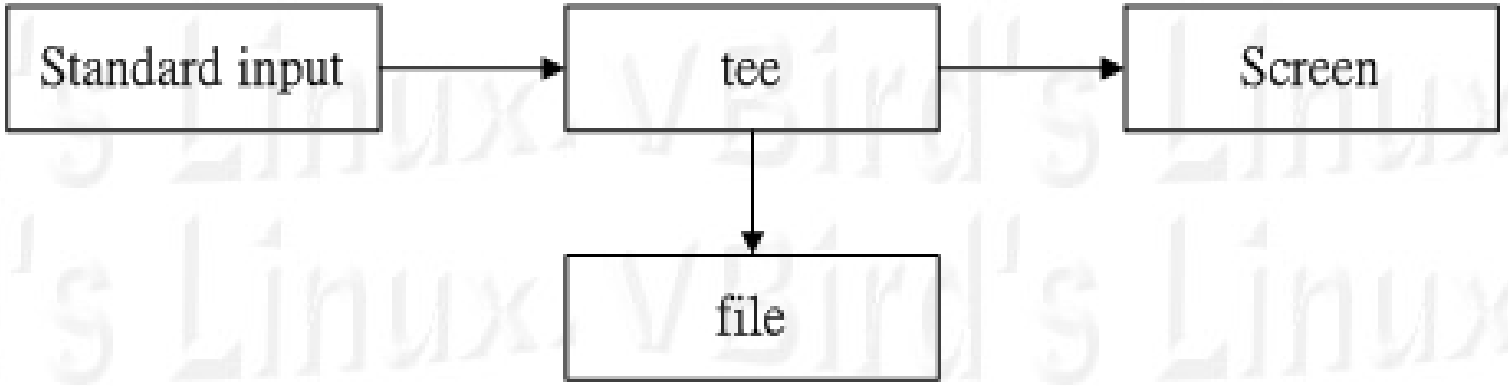
```
[root@linux ~]# sort [-fbMnrtuk] [file or stdin]
參數：
-f  ：忽略大小寫的差異，例如 A 與 a 視為編碼相同；
-b  ：忽略最前面的空白字元部分；
-M  ：以月份的名字來排序，例如 JAN, DEC 等等的排序方法；
-n  ：使用『純數字』進行排序(預設是以文字型態來排序的)；
-r  ：反向排序；
-u  ：就是 uniq ，相同的資料中，僅出現一行代表；
-t  ：分隔符號，預設是 tab 鍵；
-k  ：以那個區間 (field) 來進行排序的意思，
範例：

範例一：個人帳號都記錄在 /etc/passwd 下，請將帳號進行排序。
[root@linux ~]# cat /etc/passwd | sort
adm:x:3:4:adm:/var/adm:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologin
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
# 我省略很多的輸出~由上面的資料看起來， sort 是預設『以第一個』資料來排序，
# 而且預設是以『文字』型態來排序的喔！所以由 a 開始排到最後囉！
```

單一輸出uniq與字元計算wc

```
[root@linux ~]# uniq [-ic]
參數：
-i   ：忽略大小寫字元的不同；
-c   ：進行計數
範例：

範例一：使用 last 將帳號列出，僅取出帳號欄，進行排序後僅取出一位；
[root@linux ~]# last | cut -d ' ' -f1 | sort | uniq

範例二：承上題，如果我還想要知道每個人的登入總次數呢？
[root@linux ~]# last | cut -d ' ' -f1 | sort | uniq -c
```

```
[root@linux ~]# wc [-lwm]
參數：
-l   ：僅列出行；
-w   ：僅列出多少字(英文單字)；
-m   ：多少字元；
範例：

範例一：那個 /etc/man.config 裡面到底有多少相關字、行、字元數？
[root@linux ~]# cat /etc/man.config | wc
    138     709    4506
# 輸出的三個數字中，分別代表：『行、字數、字元數』
```

雙重導向 tee



圖五、tee 的工作流程

```
[root@linux ~]# tee [-a] file
參數：
-a   ：以累加 (append) 的方式，將資料加入 file 當中！
範例：
[root@linux ~]# last | tee last.list | cut -d " " -f1
# 這個範例可以讓我們將 last 的輸出存一份到 last.list 檔案中；
[root@linux ~]# ls -l /home | tee ~/homefile | more
# 這個範例則是將 ls 的資料存一份到 ~/homefile ，同時螢幕也有輸出訊息！
[root@linux ~]# ls -l / | tee -a ~/homefile | more
```

參數代換 xargs

```
[root@linux ~]# xargs [-0epn] command
參數：
-0   ：如果輸入的 stdin 含有特殊字元，例如 `, \, 空白鍵等等字元時，這個 -0 參數
       可以將他還原成一般字元。這個參數可以用於特殊狀態喔！
-e   ：這個是 EOF (end of file) 的意思。後面可以接一個字串，當 xargs 分析到
       這個字串時，就會停止繼續工作！
-p   ：在執行每個指令的 argument 時，都會詢問使用者的意思；
-n   ：後面接次數，每次 command 指令執行時，要使用幾個參數的意思。看範例三。
當 xargs 後面沒有接任何的指令時，預設是以 echo 來進行輸出喔！
範例：

範例一：將 /etc/passwd 內的第一欄取出，僅取三行，使用 finger 這個指令將每個
        帳號內容秀出來
[root@linux ~]# cut -d':' -f1 < /etc/passwd |head -n 3| xargs finger
```

- 讓無法支援管線命令的指令可以讀取STDOUT成爲其參數(argument)

檔案/指令搜尋

指令的搜尋

- 判斷指令從何而來，包括內建指令的顯示：
  - type command
    - ex> type –a echo
- 從 PATH 當中搜尋實際指令檔案
  - which command

由資料庫中搜尋檔案

- 檔名資料庫的建置
  - updatedb
  - 資料庫在：/var/lib/slocate
- 檔名關鍵字的搜尋
  - locate keyword
  - locate –r {正規表示法}
- man page 的搜尋
  - makewhatis →建立資料庫
  - whatis keyword →搜尋keyword是否有man page

直接找硬碟：find

- 指令語法：
  - find [目錄] [類型] [動作]
  - 範例：
    - 將過去系統上面 24 小時內有更動過內容 (mtime) 的檔案列出
      - [root@linux ~]# find / -mtime 0
    - 尋找 /etc 底下的檔案，如果檔案日期比 /etc/passwd 新就列出
      - [root@linux ~]# find /etc -newer /etc/passwd
    - 搜尋 /home 底下屬於 dmtsai 的檔案
      - [root@linux ~]# find /home -user dmtsai
    - 找出檔名爲 passwd 這個檔案
      - [root@linux ~]# find / -name passwd
    - 找出系統中，大於 1MB 的檔案
      - [root@linux ~]# find / -size +1000k

正規表示法

正規表示法

- 正規表示法的功能：
  - 透過一種符號表示的方法，來擷取所需要的資訊
  - 通常爲『一行一行』處理
  - 常用的指令如：
    - grep
    - sed
    - awk

正規表示法的字符

RE 字符	意義與範例
^word	待搜尋的字串(word)在行首！ 範例：`grep -n '^#' regular_express.txt` 搜尋行首爲 # 開始的那一行！
word$	待搜尋的字串(word)在行尾！ 範例：`grep -n '!$' regular_express.txt` 將行尾爲！的那一行列印出來！
.	代表『任意一個』字符，一定是一個任意字符！ 範例：`grep -n 'e.e' regular_express.txt` 搜尋的字串可以是 (eve) (eae) (eee) (e e)， 但不能僅有 (ee) ！亦即 e 與 e 中間『一定』僅有一個字元，而空白字元也是字元！
\	跳脫字符，將特殊符號的特殊意義去除！ 範例：`grep -n \' regular_express.txt` 搜尋含有單引號 ' 的那一行！
*	重複零個或多個的前一個 RE 字符 範例：`grep -n 'ess*' regular_express.txt` 找出含有 (es) (ess) (esss) 等等的字串，注意，因爲 * 可以是 0 個，所以 es 也是符合帶搜尋字串。另外，因爲 * 爲重複『前一個 RE 字符』的符號， 因此，在 * 之前必須要緊接著一個 RE 字符喔！例如任意字元則爲 『.*』 ！

工作管理(job control)

Job control

- 單一終端機工作介面
  - command & →在背景中『執行』
  - jobs →查看背景中的工作情況
  - fg %n →取出第n個工作到前景
  - bg %n →讓第n個工作在背景執行
  - kill %n →刪除第 n 個工作
  - [ctrl]-z →將前景的工作丟到背景暫停

精選範例

- 請問在bash中，||和&&各代表什麼意思，例如 ls foo || cd /tmp; ls&& cd /tmp？ 複選 AB
  - (A)前面一個命令若執行成功則後面的命令就不執行；前面一個命令如果執行成功則後面的命令就執行
  - (B)前面一個命令若執行失敗則後面的命令就須執行；前面一個命令如果執行失敗則後面的命令就不執行
  - (C)前面一個命令若執行成功則後面的命令就須執行；前面一個命令如果執行成功則後面的命令就不執行
  - (D)前面一個命令若執行失敗則後面的命令就不執行；前面一個命令如果執行成功則後面的命令就不執行

- Linux的 shell 操作環境中，支援下列哪些功能？複選ABD
  - (A) 提供 alias 別名設定功能
  - (B) 按<tab>鍵可以指令補齊
  - (C) 可用 ？ 號查詢指令用法，如 ls /?
  - (D) 提供 history 歷史命令功能，方便使用者操作

- 命令串『cat –n < test1 > test2 』是何意思？單選 D
  - (A) 將 test1 合併到 test2
  - (B) 將 test1 重導到 test2
  - (C) 將 test2 合併到 test1
  - (D) 將 test1 加入列號重導到 test2 去

- 關於指令『 cat /etc/passwd | grep 'vincent' 』何者正確？B
  - (A) 列出 /etc/passwd 所有內容
  - (B) 只顯示 /etc/passwd 中有 vincent 字串的整列
  - (C) 列出 /etc/passwd 中 vincent 的字串有幾個
  - (D) 指 vincent 字串清掉，並顯示其部分

- 以下關於 su 指令的敘述，何者正確？複選 AB
  - (A) 用於切換使用者身份
  - (B) 如果沒有指定欲切換之使用者，預設切換成爲 root
  - (C) 一旦切換成爲其他使用者，若欲切換回原本的使用者，必須再次執行 su 指令進行切換
  - (D) 無論任何使用者，若欲切換至其他使用者，必須輸入欲切換者之密碼，方可順利切換

- 以下關於『program1 | tee /tmp/logfile』的敘述，何者正確？A
  - (A) 所有程式執行結果都會被寫入 /tmp/logfile
  - (B) 所有程式執行結果將無法顯示於螢幕上
  - (C) 如果程式發生錯誤，所有錯誤訊息也會一併寫入 /tmp/logfile
  - (D) 本指令之作用，等同於 program | tee > /tmp/logfile

- 以下哪些指令可以列出/etc目錄下的所有檔案名稱？AB
  - (A) ls –lR /etc
  - (B) find /etc -print
  - (C) dump –R /etc
  - (D) search –l /etc

- 當使用者 david 執行『 cd ‘~david’』會產生何種動作？C
  - (A) 切換目錄到 david 的 $HOME 目錄
  - (B) 切換目錄至名稱爲 ‘~david’ 的目錄
  - (C) 切換目錄至名稱爲 ~david 的目錄
  - (D) 切換目錄至名稱爲 david 的目錄

- 下列哪依個命令行可以找出當前目錄內(不含子目錄)的符號連結檔？ A
  - (A) ls –l | grep ‘^l’ | awk ‘{print $NF}’
  - (B) ls –type l
  - (C) find –type l
  - (D) ls -ll

- 假設當前工作目錄有一個檔案，檔名爲1，且其內容爲2，請問下列那個命令會得到 2 這樣的輸出結果？ B
  - (A) /bin/false || echo $? | xargs ls
  - (B) /bin/false || echo $? | xargs cat
  - (C) /bin/false || xargs echo
  - (D) /bin/false | xargs cat

- 變數有兩種，一種是全域變數(Global)一種是自訂變數(Local)，請問下列何者正確？ ACD
  - (A) Global變數可被繼承於子程序中，而Local不行
  - (B) 想要顯示目前的 local 變數，可用 env
  - (C) 如果要讓 local 變數成爲 global ，可用 export 指令
  - (D) 命令 set 可同時顯示 local 與 global 變數

- 關於『find / -name “java*” > msg1 2>msg2』的描述，下列何者正確？C
  - (A) 該指令會搜尋根目錄底下檔名以 java 爲開頭的檔案，並將結果顯示於螢幕上
  - (B) 該指令的執行結果如了會顯示在螢幕上外，也會儲存在兩個檔案中
  - (C) 該指令正確執行結果會儲存在檔案 msg1 中，錯誤執行結果會儲存在檔案 msg2 中
  - (D) 在檔案 msg2 的內容可能會有 /home/peter/ibm_java字串

- 在 bash 中組合按鍵 [ctrl]-z 代表什麼功能？A
  - (A) 暫停目前的命令
  - (B) 終止目前的命令
  - (C) 暫停螢幕的輸出
  - (D) 恢復螢幕的輸出