



第 4 篇

作業系統與軟體設計

無
敵
隊
友

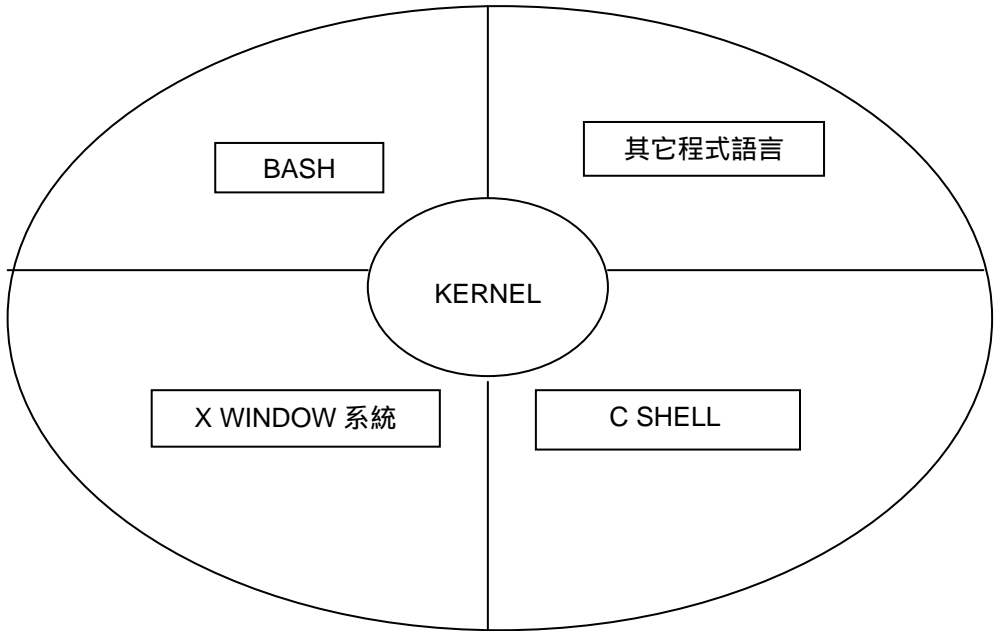
Linux

第 16 章
SHELL 程式設計

第 16 章 SHELL 程式設計

SHELL 是一種介於使用者與 UNIX 系統之間的介面程式，它讓使用者可以輸入指令來執行工作，就像我們在 DOS 下達各種的指令，來控制 KERNEL(作業系統)。

SHELL 可以看成是一種使用者環境，SHELL 我們也可以把它看成是指令直譯器。我們常見的 SHELL 有 BOURNE SHELL(BSH)、BSD CSHELL(CSH)、C SHELL(TCSH)、GNU BOURNE AGAIN SHELL。

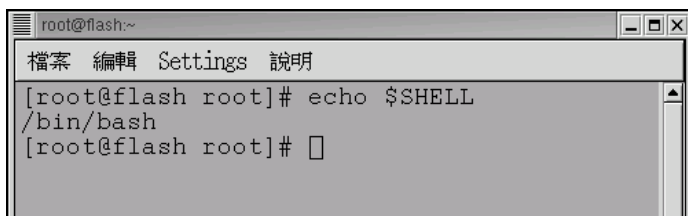


C SHELL 是由 BERKELEY UNIX 的 BILL JOY 所建立。

我們可以使用 `ECHO $SHELL` 來檢視目前所使用的 SHELL 為何。

我們預設的是 `bash`，也就是 Bourne shell。





```
root@flash:~  
檔案 編輯 Settings 說明  
[root@flash root]# echo $SHELL  
/bin/bash  
[root@flash root]#
```

我們使用者預設的 shell 是在/etc/passwd 來設定。

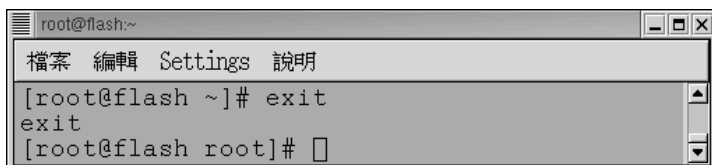
我們也可以暫時更換我們的 Shell，我們也可以永久更換我們的 shell。

我們直接打上 shell 就可以暫時改變我們 shell。例如我們打上 tcsh 就會暫時改成使用 tcsh。



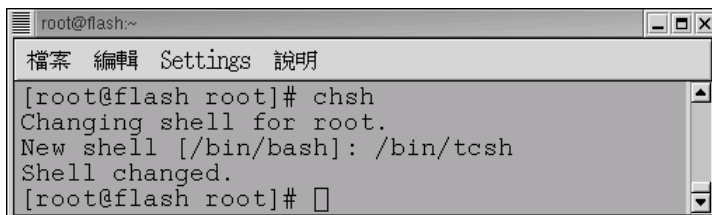
```
root@flash:~  
檔案 編輯 Settings 說明  
[root@flash root]# tcsh  
[root@flash ~]# ls  
anaconda-ks.cfg mbox
```

我們使用 exit 就可以離開這個 shell 了。



```
root@flash:~  
檔案 編輯 Settings 說明  
[root@flash ~]# exit  
exit  
[root@flash root]#
```

我們也可以永久更換我們的 shell 使用 chsh 指令(chsh)。我們然後再輸入 /bin/tcsh，這個是 tcsh 的路徑，這樣就改變 shell 了。



```
root@flash:~  
檔案 編輯 Settings 說明  
[root@flash root]# chsh  
Changing shell for root.  
New shell [/bin/bash]: /bin/tcsh  
Shell changed.  
[root@flash root]#
```



shell 本身有一組用來儲存系統資訊的變數，稱之為環境變數。環境變數依據 shell 種類不同，會有不同的變數及設定方法。我們使用 set 指令來觀看 shell 的環境變數。

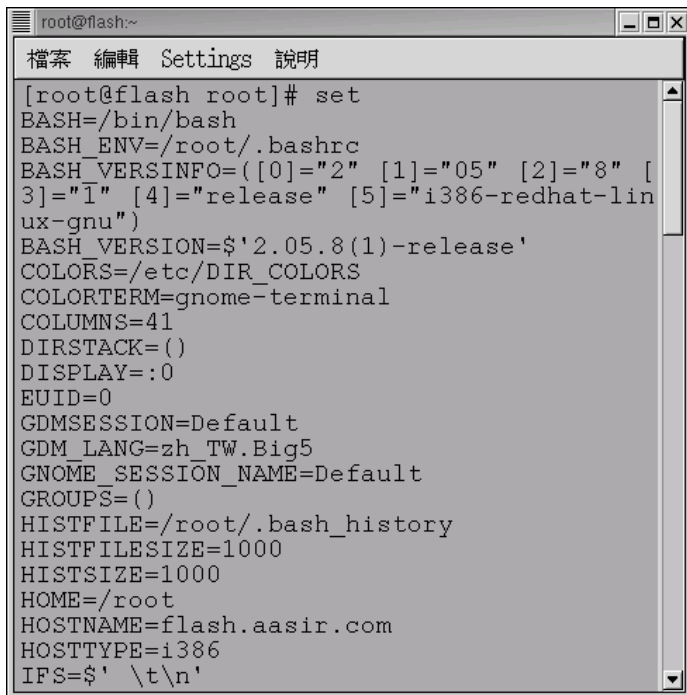
在這裏 BASH=/bin/bash 指的是 bash 所在位置。

BASH_ENV=/root/.bashrc 這是環境設定檔。

HOME=/root 使用者家目錄。

HOSTNAME=flash.aasir.com 主機名稱。

HOSTTYPE=i386

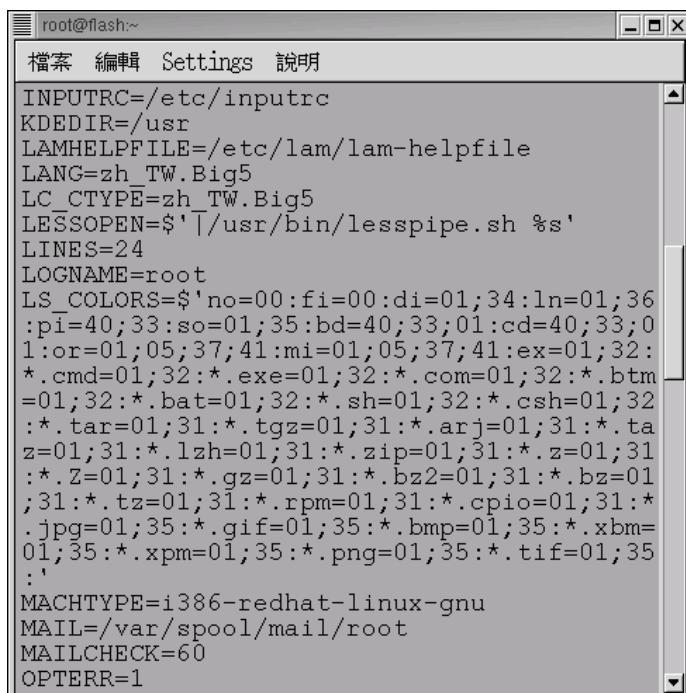


```
root@flash:~  
檔案 編輯 Settings 說明  
[root@flash root]# set  
BASH=/bin/bash  
BASH_ENV=/root/.bashrc  
BASH_VERSINFO=( [0]="2" [1]="05" [2]="8" [3]="1" [4]="release" [5]="i386-redhat-linux-gnu" )  
BASH_VERSION=$'2.05.8(1)-release'  
COLORS=/etc/DIR_COLORS  
COLORTERM=gnome-terminal  
COLUMNS=41  
DIRSTACK=()  
DISPLAY=:0  
EUID=0  
GDMSESSION=Default  
GDM_LANG=zh_TW.Big5  
GNOME_SESSION_NAME=Default  
GROUPS=()  
HISTFILE=/root/.bash_history  
HISTFILESIZE=1000  
HISTSIZ=1000  
HOME=/root  
HOSTNAME=flash.aasir.com  
HOSTTYPE=i386  
IFS=$' \t\n'
```



LANG=zh_TW.Big5 這是台灣區大五碼。

MAIL=/var/spool/mail/root 這是存放 mail 的地方。



```
root@flash:~  
檔案 編輯 Settings 說明  
INPUTRC=/etc/inputrc  
KDEDIR=/usr  
LAMHELPPFILE=/etc/lam/lam-helpfile  
LANG=zh_TW.Big5  
LC_CTYPE=zh_TW.Big5  
LESSOPEN=$'|/usr/bin/lesspipe.sh %s'  
LINES=24  
LOGNAME=root  
LS_COLORS=$'no=00:fi=00:di=01;34:ln=01;36  
:pi=40;33:so=01;35:bd=40;33;01:cd=40;33;0  
1:or=01;05;37;41:mi=01;05;37;41:ex=01;32:  
*.cmd=01;32:*.exe=01;32:*.com=01;32:*.btm  
=01;32:*.bat=01;32:*.sh=01;32:*.csh=01;32  
:*.tar=01;31:*.tgz=01;31:*.arj=01;31:*.ta  
z=01;31:*.lzh=01;31:*.zip=01;31:*.z=01;31  
:*.Z=01;31:*.gz=01;31:*.bz2=01;31:*.bz=01  
;31:*.tz=01;31:*.rpm=01;31:*.cpio=01;31:*.  
.jpg=01;35:*.gif=01;35:*.bmp=01;35:*.xbm=  
01;35:*.xpm=01;35:*.png=01;35:*.tif=01;35  
'  
MACHTYPE=i386-redhat-linux-gnu  
MAIL=/var/spool/mail/root  
MAILCHECK=60  
OPTERR=1
```

OSTYPE=LINUX-GNU 這指作業系統是 LINUX。

PATH=/USR/KERBEROS/SBIN:....., 這是指執行指令時的搜尋路徑。

PPID=2277 行程(PROCESS)的 ID 為 2277。

PROMPT_COMMAND=\$'ECHO -NE.....表示為 SHELL 的提示符號。

PWD=/root 指目前的目錄為/ROOT。

SHELL=/bin/bash 指 SHELL 目前的所在目錄。



```

root@flash:~
檔案 編輯 Settings 說明
OPTIND=1
OSTYPE=linux-gnu
PATH=/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/sbin:/bin:/usr/bin:/usr/bin/X11:/usr/local/bin:/usr/bin:/usr/X11R6/bin:/root/bin:/root/bin
PIPESTATUS=( [0]="0" )
PPID=2277
PROMPT_COMMAND=$'echo -ne "\\033]0;${USER}@${HOSTNAME}%. *}:${PWD/$HOME/~}\\007"'
PS1=$'[\\u@\\h \\w]\\$ '
PS2=$'> '
PS4=$'+ '
PVM_ROOT=/usr/share/pvm3
PVM_RSH=/usr/bin/rsh
PWD=/root
QTDIR=/usr/lib/qt-2.3.1
SESSION_MANAGER=local/flash.aasir.com:/tmp/.ICE-unix/1725
SHELL=/bin/bash
SHELLOPTS=braceexpand:hashall:histexpand:monitor:history:interactive-comments:emacs
SHLVL=3

```

TERM=XTERM 終端機型態。

UID=0 指的是 USERID

USER=root 指的是使用者名稱。

```

root@flash:~
檔案 編輯 Settings 說明
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
SUPPORTED=zh_TW.Big5:zh_TW:zh:en_US:en
TERM=xterm
UID=0
USER=root
USERNAME=root
WINDOWID=29360262
XAUTHORITY=/root/.Xauthority
XMODIFIERS=@im=xcin
XPVM_ROOT=/usr/share/pvm3/xpvm
_=/etc/bashrc
i=/etc/profile.d/xpvm.sh
langfile=/root/.i18n
sourced=1
mc ()
{
    mkdir -p ~/.mc/tmp 2>/dev/null;
    chmod 700 ~/.mc/tmp;
    MC=~/.mc/tmp/mc-$$;
    /usr/bin/mc -P "$@" >"$MC";
    cd "`cat $MC`";
    /bin/rm "$MC";
    unset MC
}

```

16-1 shell script

shell 是功能強大的交談式指令解譯器，而 shell script 是普通的文字檔案，它能夠在 Shell 下執行，而且具有接受指令列參數，使用者輸入、輸出以及設定變數的能力。

Shell script 指令分成系統指令(例如:ls、cat、rm、ps)與 shell 指令(case、loop、if 等 shell 直接解譯的指令)。

我們第一個 shell 程式 first.sh。

我們使用#!/bin/bash 當作告訴系統後面接著是用來執行檔案的程式，也就是我們預設的 shell 程式/bin/sh。

我們使用#當作註解的意義，而#號之後的那一行都會沒有作用，而當作註解。

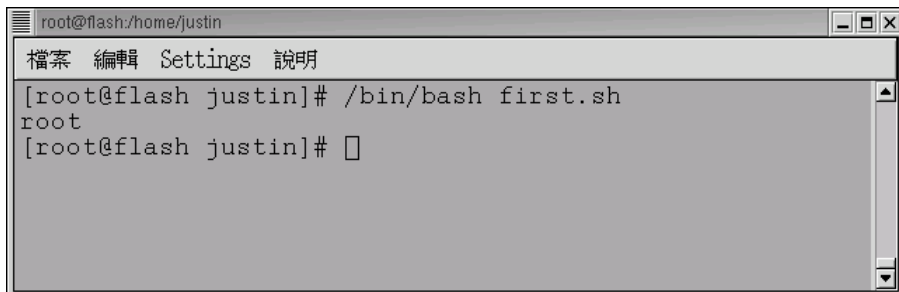
我們使用 echo `whoami` 來顯示使用者的名稱。



```
root@flash:/home/justin
檔案 編輯 Settings 說明
#!/bin/bash
#fisrt.sh
echo `whoami`

~
~
~
"first.sh" 4L, 36C
```

執行 script。我們先指定/bin/bash 這是 shell 的執行路徑，再將我們的 script 當作參數來執行，在這邊執行的情況為 root。



```
root@flash:/home/justin
檔案 編輯 Settings 說明
[root@flash justin]# /bin/bash first.sh
root
[root@flash justin]#
```



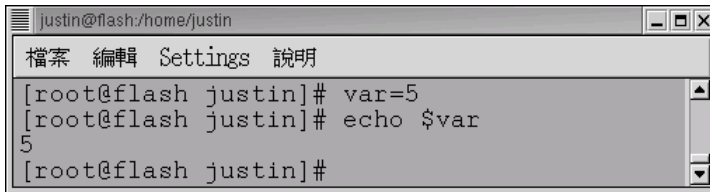
16-2 Shell 語法

shell 包含變數、布林、流程控制、清單、函數、shell 內建的命令、命令執行的結果。

16-2-1 變數

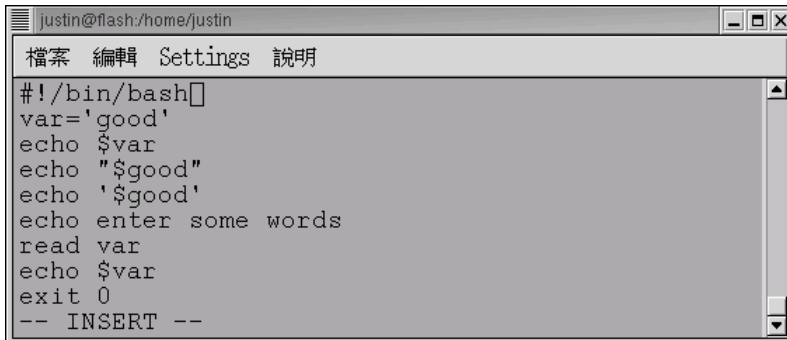
變數名稱的大小寫是不同的，如\$Variable 和\$variable 是不同的。

我們使用 var=5 將數值 5 給 var 變數，再使用 echo \$var 來顯示\$var 變數。



```
justin@flash:/home/justin
檔案 編輯 Settings 說明
[root@flash justin]# var=5
[root@flash justin]# echo $var
5
[root@flash justin]#
```

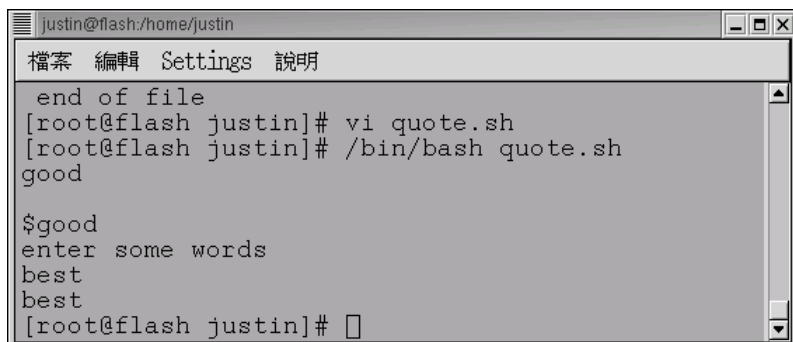
我們使用 vi quote.sh 來編輯 quote.sh 檔，在第一行指定#!/bin/bash，這是指定的程式。我們然後將字串 good 給 var 變數。我們然後使用 read 指令從外部讀取所輸入的值給 var 變數。



```
justin@flash:/home/justin
檔案 編輯 Settings 說明
#!/bin/bash
var='good'
echo $var
echo "$good"
echo '$good'
echo enter some words
read var
echo $var
exit 0
-- INSERT --
```

我們使用/bin/bash quote.sh 來執行 shell 指令。





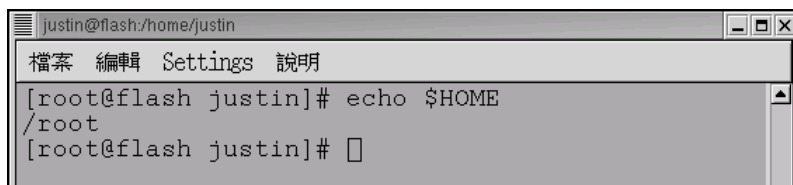
```
justin@flash:/home/justin
檔案 編輯 Settings 說明
end of file
[root@flash justin]# vi quote.sh
[root@flash justin]# /bin/bash quote.sh
good

$good
enter some words
best
best
[root@flash justin]#
```

環境變數

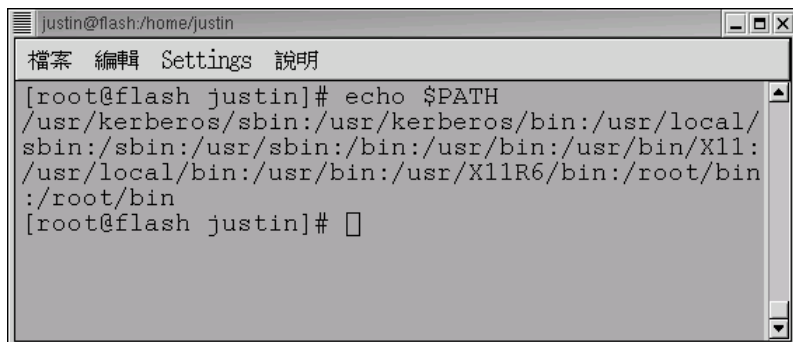
shell script 中的某一些變數為預定的環境變數。

`$HOME` 可以顯示目前使用者的根目錄。



```
justin@flash:/home/justin
檔案 編輯 Settings 說明
[root@flash justin]# echo $HOME
/root
[root@flash justin]#
```

`$PATH` 用來搜尋命令，以冒號分開的目錄清單。



```
justin@flash:/home/justin
檔案 編輯 Settings 說明
[root@flash justin]# echo $PATH
/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/
sbin:/sbin:/usr/sbin:/bin:/usr/bin:/X11:
/usr/local/bin:/usr/bin:/usr/X11R6/bin:/root/bin
[root@flash justin]#
```

`$0` 可以顯示我們 shell script 的名稱。



```

justin@flash:/home/justin
檔案 編輯 Settings 說明
[root@flash justin]# echo $0
bash
[root@flash justin]# █

```

\$#執行所經過參數的名稱。

```

justin@flash:/home/justin
檔案 編輯 Settings 說明
[root@flash justin]# echo $#
0
[root@flash justin]# █

```

\$\$這可以顯示 shell script 的行程 ID。

```

justin@flash:/home/justin
檔案 編輯 Settings 說明
[root@flash justin]# echo $$
2770
[root@flash justin]# █

```

如果我們的 script 需配合某一些參數一啟使用。我們的參數變數有下列幾種。

參數變數	涵義
\$1,\$2,\$3,,,	Script 所使用的參數
\$*	可以列出所有的 script 參數

我們的 set 指令有三個參數，分別是 beee、bee、和 be。我們使用 echo \$1 來顯示第一個參數，因此得到 beee。我們使用 echo \$2 來顯示第二個參數，因此得到 bee。我們使用 echo \$3 來顯示第三個參數，因此得到 be。我們使用 echo \$*來顯示所有的參數。



```
root@flash:~  
檔案 編輯 Settings 說明  
[root@flash root]# set beee bee be  
[root@flash root]# echo $1  
beee  
[root@flash root]# echo $2  
bee  
[root@flash root]# echo $3  
be  
[root@flash root]# echo $*  
beee bee be  
[root@flash root]#
```

這是顯示第一個參數 beee。

```
[root@flash root]# set beee bee be  
[root@flash root]# echo $1  
beee
```

這是顯示所有的參數，分別是 beee,bee 和 be。

```
[root@flash root]# echo $*  
beee bee be
```

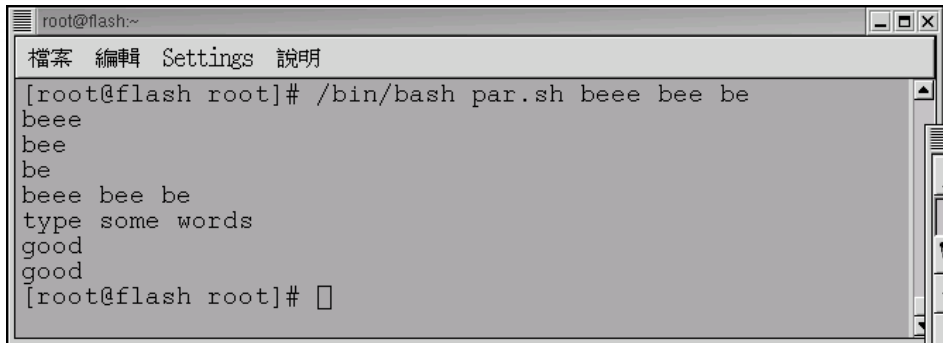
我們建立 par.sh 檔，第一行會顯示 \$1 第一個參數，read par 會將我們輸入的資料存入 par 變數中，而使用 echo \$par 來顯示。

#vi par.sh

```
root@flash:~  
檔案 編輯 Settings 說明  
#!/bin/bash  
echo $1  
echo $2  
echo $3  
echo $*  
echo 'type some words'  
read par  
echo $par  
-- INSERT --
```

我們使用/bin/bash par.sh beee bee be 來執行。





```
root@flash:~  
檔案 編輯 Settings 說明  
[root@flash root]# /bin/bash par.sh beee bee be  
beee  
bee  
be  
beee bee be  
type some words  
good  
good  
[root@flash root]#
```

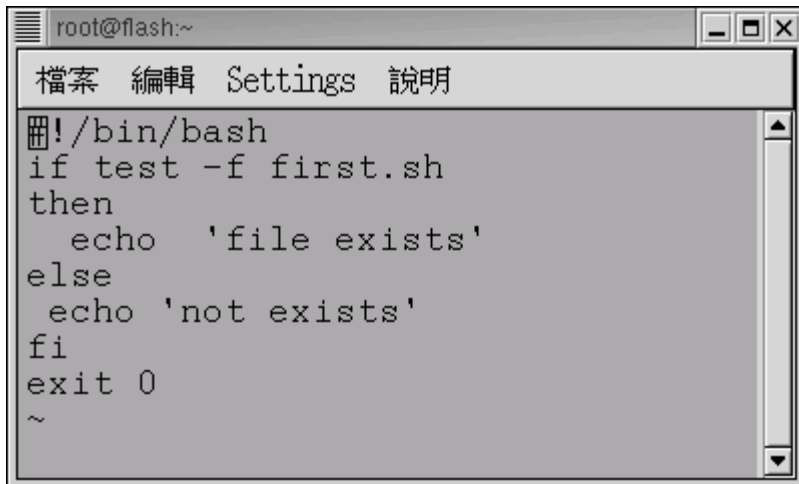
16-2-2 布林值

我們可以使用 `test` 和 `[]` 來測試布林(Boolean check)。

例如我們可以使用 `test -f` 檔案來測試檔案是否存在。

我們可以編輯 `bo.sh` 檔。我們在第一行表示這是要執行 `bash` 檔，而我們使用 `test -f first.sh` 來測試檔案 `first.sh` 是否存在。

```
#vi bo.sh
```



```
root@flash:~  
檔案 編輯 Settings 說明  
#!/bin/bash  
if test -f first.sh  
then  
    echo 'file exists'  
else  
    echo 'not exists'  
fi  
exit 0  
~
```

我們可以編輯 `bo2.sh` 檔，而我們使用 `[-f bo.sh]` 來測試 `bo.sh` 是否存在。

```
#vi bo2.sh
```

```

root@flash:~
檔案 編輯 Settings 說明
#!/bin/bash
if [ -f bo.sh ]
then
    echo 'existence'
else
    echo 'not existence'
fi
~
~
"bo2.sh" 7L, 83C

```

我們執行/bin/bash bo2.sh，得到 existence。

```

root@flash:~
檔案 編輯 Settings 說明
[root@flash root]# ls
anaconda-ks.cfg  bo.sh  par.sh
bo2.sh           mbox
[root@flash root]# /bin/bash bo2.sh
existence
[root@flash root]# █

```

字串比較

字串比較	結果
String1=string2	字串相等則為 true
String !=string2	字串不相等則為 true
-n string	非空字串則為 true
-z string	非空字串則為 true

代數比較

代數比較	結果
運算式 1 -eq 運算式 2	運算式相等則為 true
運算式 1 -ne 運算式 2	運算式不相等則為 true
運算式 1 -gt 運算式 2	運算式 1 大於運算式 2 則為 true



運算式 1 -ge 運算式 2	運算式 1 大於或等於運算式 2 則為 true
運算式 1 -lt 運算式 2	運算式 1 小於運算式 2 則為真
運算式 1 -le 運算式 2	運算式 1 小於或等於運算式 2 則為 true
! 運算式	運算式為 false 則為真

檔案條件式

代數比較	結果
-d file	檔案為一目錄則為 true
-e file	檔案存在則為 true
-f file	屬於一班的檔案則為 true
-g file	檔案中有設定-group 則為 true
-r file	檔案屬性為可讀則為 true
-s file	檔案大小為非 0 則為 true
-u file	檔案中有設定 set -user 則為 true
-w file	檔案屬性為可寫入則為 true
-x file	檔案屬性為可執行則為 true

16-2-3 陣列

陣列就是在記憶體位址上，存放相同型態的資料。

陣列語法：

`${name[i]}` 取用 name 陣列的第 i-1 個元素

`${name[*]}` 取用 name 陣列的所有元素

`${name}` 取用 name 陣列的第一個元素

第二行到第七行為我們將 1、2、3、4、5 分別存入陣列 name 中，再由第七行將陣列內容顯示出來。

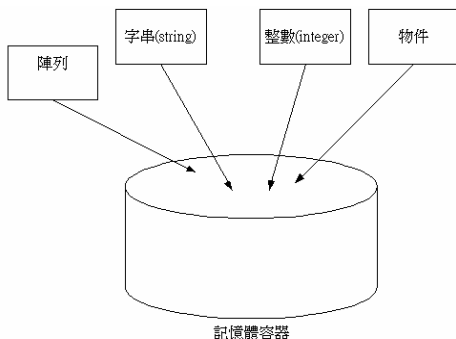


```
1 #!/bin/sh
2 name[1]=1
3 name[2]=2
4 name[3]=3
5 name[4]=4
6 name[5]=5
7 echo ${name[*]}
```

這是當我們執行 array.sh 將陣列資料都顯示出來。

```
[root@aasir chaiyen]# /bin/sh array.sh
1 2 3 4 5
[root@aasir chaiyen]#
```

我們的記憶體可以放陣列、字串、和數值。



16-3 控制結構

控制結構(control structure)控制程式或函數的執行流程，控制結構可將個別的指令組成單一的邏輯單元，有進入點就有出口點。

有三種控制結構可以控制執行的流程，分別是循序結構、選取結構、和迴圈結構。一般程式的行程都是一行接著一行的循序執行，這稱為循序結構。而選取結構為可在不同的程式敘述中作選擇。迴圈結構則為重複執行同樣的幾行程式，直到條件不滿足而跳出迴圈。



循序結構：

程式碼第一行；

程式碼第二行；

程式碼第三行；

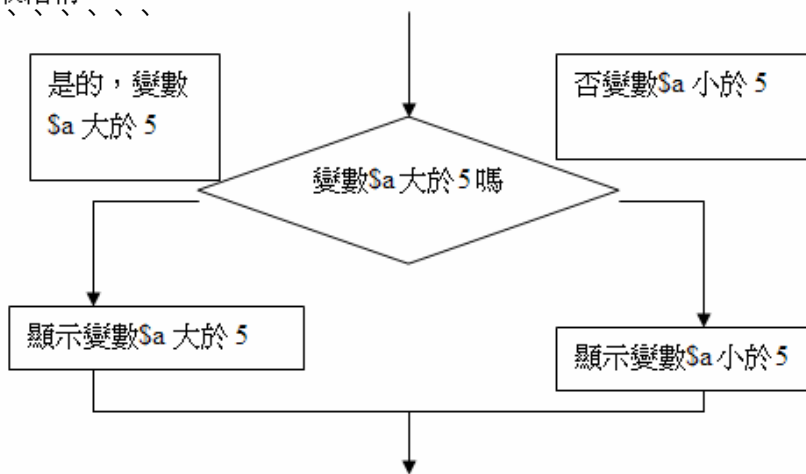
、 、 、 、 、 、

、 、 、 、 、 、

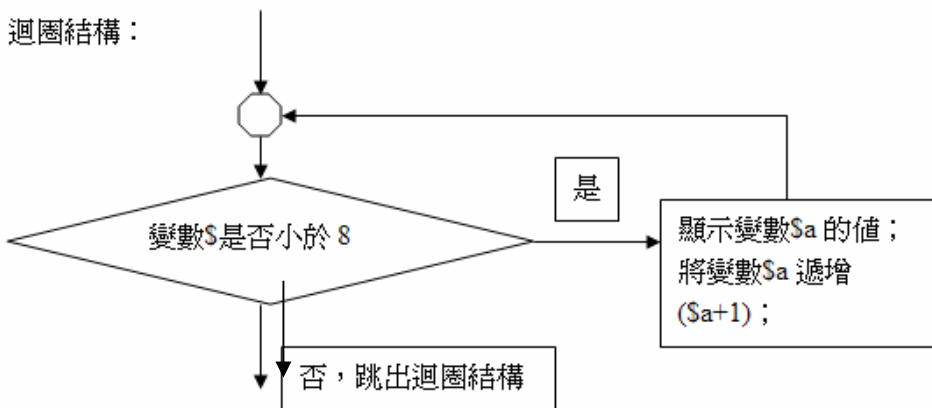
選取結構：

、 、 、 、 、 、

循序結構，就是程式一行一行的由上而下循序執行。

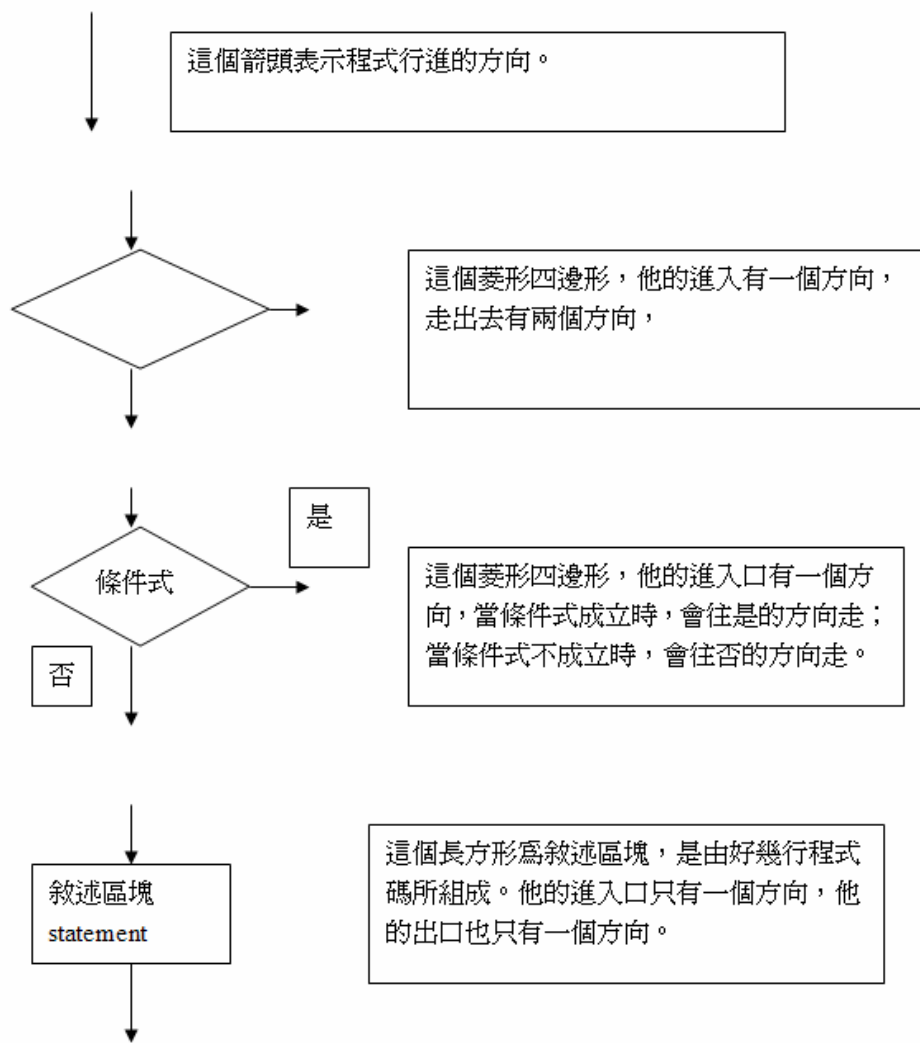


迴圈結構：



圖形解釋：

我們在作大型系統或軟體開發時會使用統一模型(UML)語言，因此使用圖示模型來表示程式的資料與程序，我們在這一章將用到許多的模型圖示來表示。



16-3-1 選取結構 if

if 條件

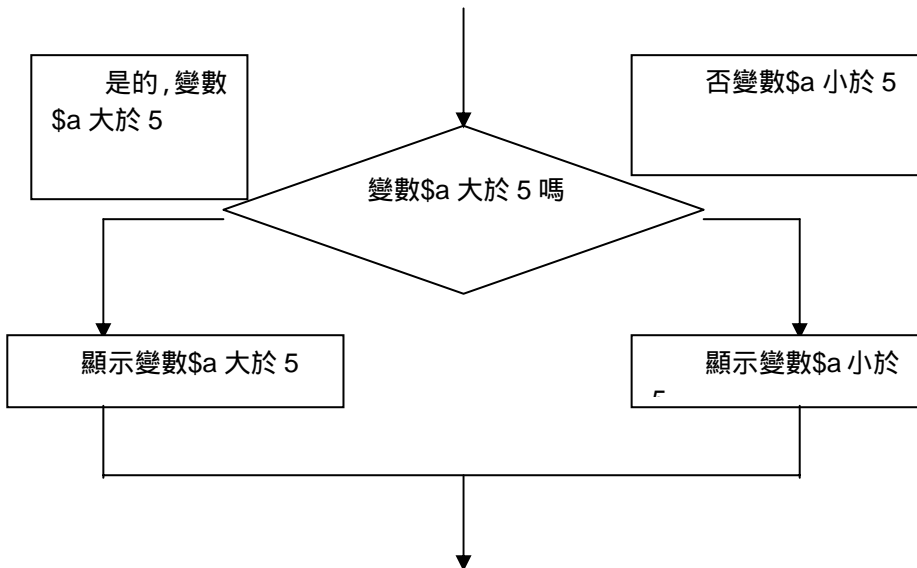
then

敘述

else

敘述

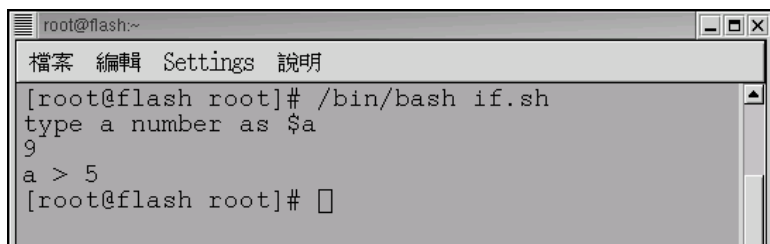
fi



我們可以使用 vi 編輯 if.sh 我們使用[$a > 5$] 來辨別我們輸入的變數\$a(read a) 是否大於 5。

```
root@flash~  
檔案 編輯 Settings 說明  
#!/bin/bash  
echo 'type a number as $a '  
read a  
if [ $a > 5 ]  
then  
  echo 'a > 5'  
else  
  echo 'a <= 5'  
fi  
exit 0
```

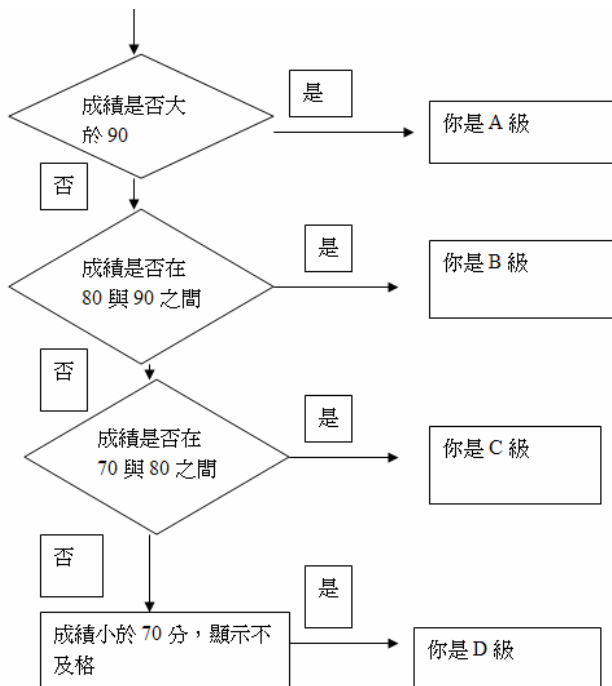
我們使用/bin/bash if.sh 來執行，當我們輸入的數字為 9 時，就會顯示 a > 5。



```
root@flash:~  
檔案 編輯 Settings 說明  
[root@flash root]# /bin/bash if.sh  
type a number as $a  
9  
a > 5  
[root@flash root]#
```

16-3-2 巢狀的 if 敘述

我們已經看過 if 敘述 1 個或 2 個的選擇，在此要用巢狀 if 敘述來撰寫多重選擇決策。例如我們的成績評等，90 到 100 分的為 A 級，80 到 89 的為 B 級，70 到 79 的為 C 級，小於 70 分的為 D 級。如果我有一位學生分數為 82 分，他的分數在 80 到 89 分之間，則他的等級為 B 級。在這個例子中 4 個等級來做選擇，就叫做多重選擇。



巢狀的選擇決策

語法：

```
if 條件 1
```

```
then
```

```
    敘述 1
```

```
    else
```

```
    if 條件 2
```

```
        then
```

```
            敘述 2
```

```
        fi
```

```
        . . .
```

```
    . . . . .
```

```
    . . . . .
```

```
fi
```

當我們要查詢成績時，只要輸入成績，就可以判段成績的好壞。

```
[root@aasir chaiyen]# /bin/bash nif.sh
PLEASE INPUT YOUR GRADE
90
你的成績是A
[root@aasir chaiyen]# /bin/bash nif.sh
PLEASE INPUT YOUR GRADE
80
你的成績是B
[root@aasir chaiyen]# _
```

這是程式碼

第二行是請輸入成績。

第三行是讀取成績變數。

第五行到第二十行是巢狀迴圈的 if 判別。第五行到第七行是顯示當成績大於等於 90 時會顯示 A(記得在中括號要加空格否則為命令), 否則會顯示第八到第二十行(成績小於 90)。第九行到第十一行是顯示當成績大於等於 80, 也就是成績會在 80 到 90



間。第十二到第二十行會顯示成績小於 80。第十二到第十五行會顯示成績小於 80 但大於等於 70。第十六到第十九行則會顯示成績小於 70。

```
1 #/bin/sh
2 echo "PLEASE INPUT YOUR GRADE"
3 read grade
4
5 if [ $grade -ge 90 ]
6 then
7     echo "你的成績是A"
8 else
9     if [ $grade -ge 80 ]
10    then
11        echo "你的成績是B"
12    else
13        if [ $grade -ge 70 ]
14        then
15            echo "你的成績是C"
16        else
17            echo "你的成績是D"
18        fi
19    fi
20 fi
21 exit 0
~
~
;set nu
```

16-3-3 case 選擇決策

case 選取結構允許我們執行設定為複雜多樣的形式，每一敘述皆以雙分號;;作為結尾。而最後以 esac 結尾。

語法: case 變數 in

條件 1)敘述 1;;

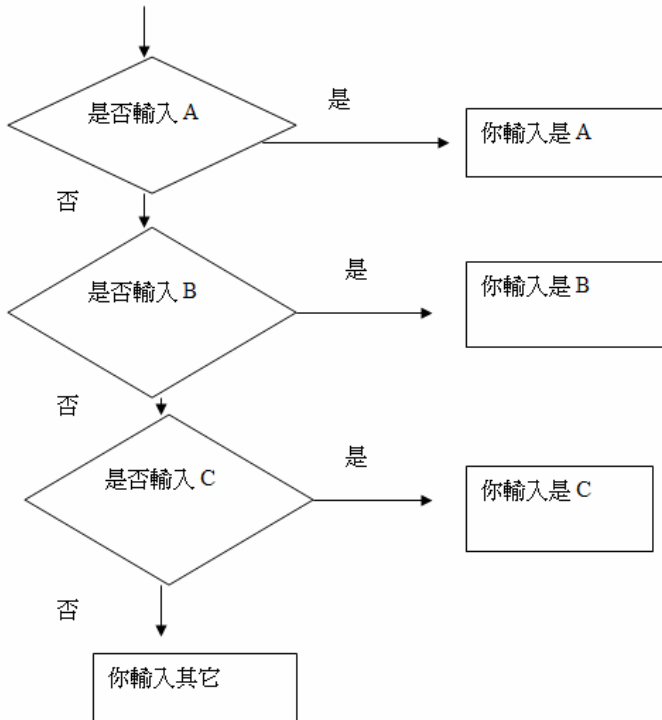
條件 2)敘述 2;;

.....

條件 n)敘述 n;;

esac





第四行到第九行為 case 選取結構，當變數 var 為 A 時就會執行第五行，當變數 var 為 B 時就會執行第六行，當變數 var 為 C 時就會執行第七行。當為其它符號時則執行第八行。星號*是代表其它。

```
1 #!/bin/sh
2 echo "請輸入大寫字母A或B或C"
3 read var
4 case "$var" in
5     A ) echo "你輸入是A";;
6     B ) echo "你輸入是B";;
7     C ) echo "你輸入是C";;
8     *) echo "你輸入其它";;
9 esac
10 exit 0
```

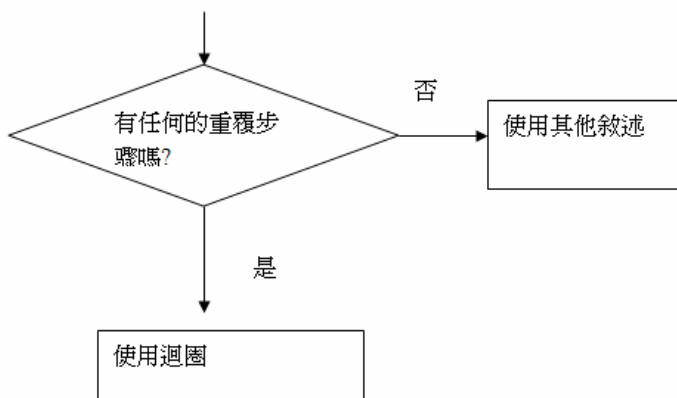


這是我們執行的情況。

```
[root@aasir chaiyen]# /bin/sh case.sh
請輸入大寫字母A或B或C
A
你輸入是A
[root@aasir chaiyen]# /bin/sh case.sh
請輸入大寫字母A或B或C
b
你輸入其它
[root@aasir chaiyen]#
```

16-3-4 迴圈結構

選取結構和循序結構，都只執行程式敘述一次，如果我們要讓同一行程式重複執行好幾遍則要用迴圈敘述。迴圈敘述可以重複執行某一段程式好幾遍，直到條件的不成立才跳出這個迴圈。shell 的迴圈敘述有 WHILE 迴圈和 FOR 迴圈。



1 For 迴圈

語法:FOR 每一個變數 IN 串列

DO

敘述

DONE

這是當我們執行 for.sh 時，會迴圈的執行串列。

```
[root@aasir chaiyen]# /bin/bash for.sh
a
b
c
d
e
f
[root@aasir chaiyen]#
```

第二行到第五行是 for 迴圈，for 迴圈總共會執行六次(也就是串列的結點數目)，第四行就是將串列的值顯示出來。

#vi for.sh

```
1 #!/bin/sh
2 for var in a b c d e f
3 do
4   echo $var
5 done
6 exit 0
```



2 WHILE 迴圈

語法:WHILE(條件式)

DO

敘述

DONE

我們在第二行設定 while 迴圈的初使條件,讓 $x=10$ 第三行到第七行會執行 while 迴圈,每次都會顯示變數 x 的值。第三行中括號裏面會測試 test 變數有沒有大於 0,第四行會顯示變數 x 的值,第六行會將變數 x 減 1。第六行我們使用雙括號來當變數 x 運算的結果,再給變數,就跟 C 語言的 $x--$ 相同。

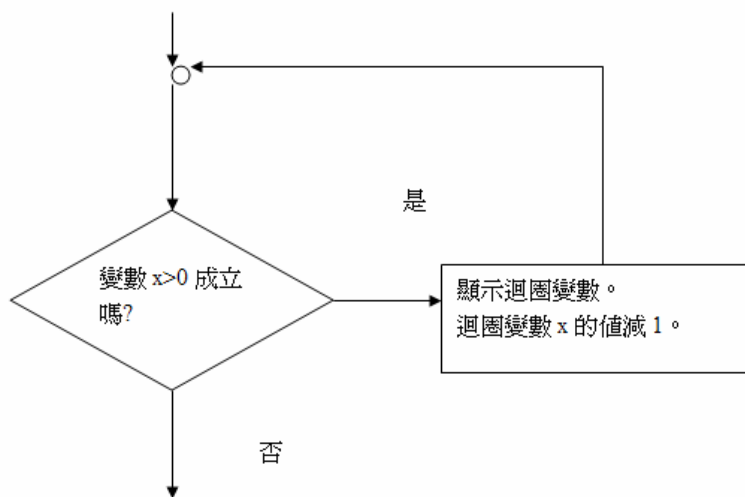
#vi b.sh

```
1 #!/bin/sh
2 x=10
3 while [ "$x" -ge 0 ]
4 do
5     echo $x
6     x=$(( $x-1 ))
7 done
8 exit 0
```

這是執行 while 迴圈的情況,一開始是 10 然後每次減 1 直到 0 為止。



```
[root@aasir chaiyen]# /bin/sh b.sh
10
9
8
7
6
5
4
3
2
1
0
```

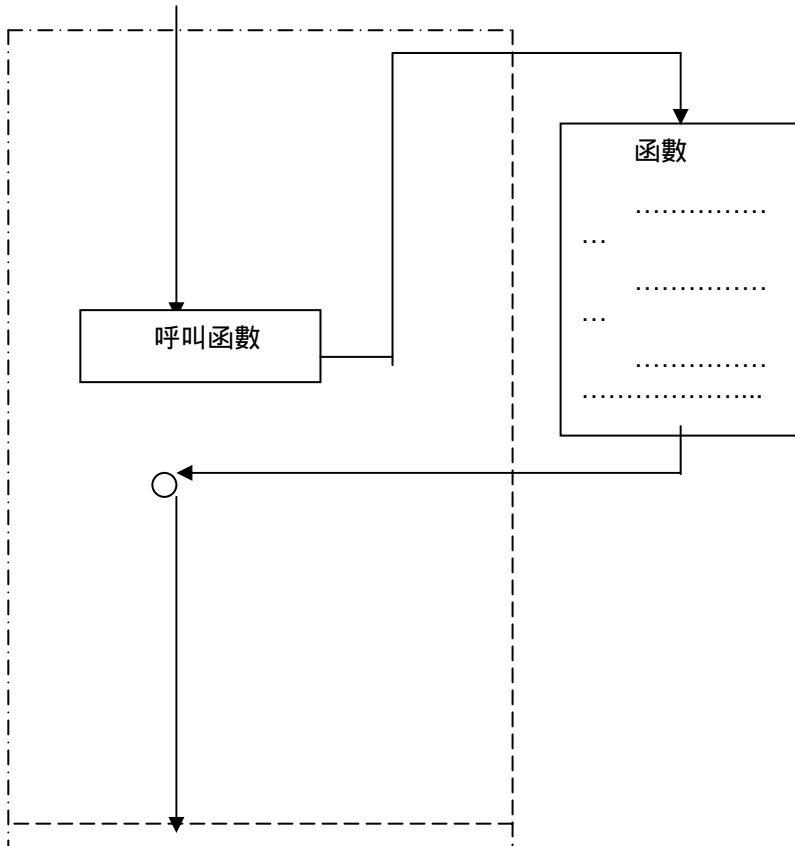


16-4 函數

結構化程式設計，在程式的模組化和由上而下的程式設計。在程式設計時，我們常將較大的程式分成數個較小的功能，每個小功能都能夠很容易的分析，並且撰寫。我們可以將這些小功能寫成副程式。而且我們可以將程式中重複的程式寫成一個副程式，當我們需要時再呼叫這副程式，這樣可以易於維護程式碼。

當主程式在呼叫函數時，執行的程式就會跳到函數的記憶體敘述區塊，然後執行敘述區塊，當執行完後再跳回程式主體繼續往下執行。

虛線所包含的就是主程式



這是我們執行函數的情況。

```
[root@aasir chaiyen]# /bin/sh func.sh
good
函數執行
函數執行
```

第二行到第四行為 fun() 函數。第七行和第八行為呼叫函數 fun，因此會顯示出函數執行。

#vi func.sh

```
1 #!/bin/sh
2 fun(){
3 echo "函數執行"
4 }
5 echo "good"
6
7 fun
8 fun
9 exit 0
```



課後練習

1. Shell 啟動時一定會讀取的檔案是下列哪一個？

- (A). A./etc/profile
- (B). B./etc/bashrc
- (C). C./etc/.bash_logout
- (D). D./etc/inittab

2. 下列哪一種是介於使用者與 UNIX 系統之間的介面程式，它讓使用者可以輸入指令來執行工作，就像我們在 DOS 下達各種的指令，來控制 KERNEL(作業系統)？

- (A). GUI
- (B). APACHE
- (C). Console
- (D). SHELL

3. shell 本身有一組用來儲存系統資訊的變數，稱之為下列哪一個？依據 shell 種類的不同，會有不同的變數及設定方法。我們使用 set 指令來關看 shell 的環境變數。

- (A). 區域變數
- (B). 環境變數
- (C). 全域變數
- (D). 不同變數

4. 下列哪個指令可以顯示該目錄的檔案？複選

- (A). `ls`
- (B). echo 'ls'
- (C). ls
- (D). echo `ls`



5. 陣列就是在記憶體的位址上，存放相同型態的資料。請問`${name[*]}`在 SHELL 中是什麼意義？

- (A). `${name[*]}` 取用 name 陣列的第 I+1 個元素
- (B). `${name[*]}` 取用 name 陣列的第 I-1 個元素
- (C). `${name[*]}` 取用 name 陣列的所有元素
- (D). `${name[*]}`取用 name 陣列的第一個元素

6. case 選取結構允許我們執行設定為複雜多樣的形式，每一敘述皆以雙分號`;;`作為結尾。而最後以下列那一個結尾？

語法: case 變數 in

 條件 1)敘述 1;;

 條件 2)敘述 2;;

 條件 n)敘述 n;;

- (A). end
- (B). out
- (C). esac
- (D). sace

【答案】

1. A 2. D 3. B 4. C,D 5. C 6. C

