
Lập trình Hệ thống

Unix Programming

Part 1: Shell Programming

Nguyễn Quốc Tuấn

Network and Communication System Department
Faculty of Electronics and Communications
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Shell Programming

❑ CÁC TOÁN TỬ STRING

- Các biến có thể được kiểm tra và sửa đổi bằng cách sử dụng các công cụ sửa đổi đặc biệt. Công cụ sửa đổi kiểm tra xem một biến đã được đặt hay chưa, sau đó gán giá trị cho biến dựa trên kết quả của phép thử.

Toán tử	Chức năng
<code>\${var:- word}</code>	Nếu biến tồn tại và xác định thì trả về giá trị của nó, nếu không thì trả về word
<code>\${var:= word}</code>	Nếu biến tồn tại và xác định thì trả về giá trị của nó, nếu không thì gán biến thành word, sau đó trả về giá trị của nó

<code>\${var:+ word}</code>	Nếu biến tồn tại và xác định thì trả về word, còn không thì trả về null
<code>\${var:?message}</code>	Nếu biến tồn tại và xác định thì trả về giá trị của nó, còn không thì hiển thị "bash: \$var:\$message" và thoát ra khỏi lệnh/tập lệnh hiện thời.
<code>\${var:offset[:length]}</code>	Trả về một chuỗi con của var bắt đầu tại offset của độ dài length. Nếu length bị bỏ qua, toàn bộ chuỗi từ offset sẽ được trả về.

Shell Programming

❑ CÁC TOÁN TỬ STRING

- Ví dụ

- Xét một biến shell tên là status được khởi tạo giá trị “defined”. Sử dụng các toán tử string cho kết quả status như sau:

```
$ echo ${status:-undefined}
```

```
defined
```

```
$ echo ${status:=undefined}
```

```
defined
```

```
$ echo ${status:+undefined}
```

```
undefined
```

```
$ echo ${status:?Dohhh\! undefined}
```

```
defined
```

Shell Programming

❑ CÁC TOÁN TỬ STRING

- Lệnh **unset** để xoá biến status,

```
$ unset status
```

```
$ echo ${status:-undefined}
```

```
undefined
```

```
$ echo ${status:=undefined}
```

```
undefined
```

```
$ echo ${status:+undefined}
```

```
undefined unset status
```

```
$ echo ${status:?Dohhh\! undefined} bash:status Dohhh!
```

```
Undefined
```

Shell Programming

❑ CÁC TOÁN TỬ PATTERN-MATCHING

- Có thể sử dụng để loại bỏ hoặc loại bỏ mẫu mong muốn khỏi biến hoặc chuỗi đã cho

Toán tử	Chức năng
<code>\${var#pattern}</code>	Xoá bỏ phần khớp (match) ngắn nhất của pattern trước var và trả về phần còn lại
<code>\${var##pattern}</code>	Xoá bỏ phần khớp (match) dài nhất của pattern trước var và trả về phần còn lại
<code>\${var%pattern}</code>	Xoá bỏ phần khớp ngắn nhất của pattern ở cuối var và trả về phần còn lại

<code>\${var%%pattern}</code>	Xoá bỏ phần khớp dài nhất của pattern ở cuối var và trả về phần còn lại
<code>\${var/pattern/string}</code>	Thay phần khớp dài nhất của pattern trong var bằng string. Chỉ thay phần khớp đầu tiên. Toán tử này chỉ có trong bash 2.0 hay lớn hơn.
<code>\${var//pattern/string}</code>	Thay phần khớp dài nhất của pattern trong var bằng string. Thay tất cả các phần khớp. Toán tử này có trong bash 2.0 hoặc lớn hơn.

Shell Programming

❑ CÁC TOÁN TỬ PATTERN-MATCHING

▪ Ví dụ

```
#!/bin/bsh
```

```
echo "Enter a string: "
```

```
read str
```

```
echo "The original string is $str"
```

```
echo "The string after deleting first 2 characters is ${str#??}"
```

```
echo "The string after deleting the last 2 characters is ${str%??}"
```

Output:

Enter a string: education

The original string is education

The string after deleting first 2 characters is ucation

The string after deleting the last 2 characters is educati

Shell Programming

❑ LỆNH SHELL

▪ Lệnh tr: Applying Translation

- Được sử dụng để dịch một tập các chuỗi với một tập khác, nghĩa là, mỗi ký tự trong tập ký tự đầu tiên được thay thế bằng một ký tự tương ứng trong tập hợp cụ thể thứ hai.. Các chuỗi được chỉnh sửa cụ thể bằng cách sử dụng dấu ngoặc kép.

- Cú pháp tr [options] [set1 [set2]]

Mô tả tùy chọn

-d Xóa ký tự đặc biệt trong set1

-s Bóp ký tự lặp lại trong set1

-c Áp dụng bản dịch phần bổ sung của set1,

Ví dụ:

\$ tr "aei" "AEIOU"

It is very easy to use

It is vEry EAsy to usE ^d

Ví dụ: Để đổi các kí tự lower-case sang upper case trong file school

\$ tr '[a-z]' '[A-Z]' < school

- Xóa các kí tự B from the file school.txt

\$ tr -d B < school.txt

Shell Programming

❑ LỆNH SHELL

▪ Lệnh `tty`: Terminal Command

- Tiện ích `tty` được sử dụng để hiển thị tên của thiết bị đầu cuối mà chúng đang được sử dụng. Unix coi mỗi thiết bị đầu cuối là một tệp, có nghĩa là tên của thiết bị đầu cuối của chúng thực sự là tên của một tệp

Khuôn dạng: `tty [s]`

- Ví dụ : `$ tty`

`/dev/tty2`

Kết quả đầu ra cho thấy tên của thiết bị đầu cuối là `/dev/tty2` hoặc đơn giản hơn là `tty2`. Trong Unix, tên của một thiết bị đầu cuối thường có tiền tố `tty`.

Shell Programming

❑ LỆNH SHELL

▪ Lệnh trap:

- **Trap** được sử dụng để thực hiện lệnh khi lệnh của chúng ta nhận được tín hiệu.

Cú pháp: *trap cmd signals*

- Signals là danh sách các báo hiệu để ngắt. Nếu cmd bị thiếu thì sẽ không có gì xảy ra khi nhận được hiệu.

- **Ví dụ** : (a) `trap "ls" 1 2 3 15`

(b) `trap "ls" SIGHUP SIGINT SIGQUIT SIGTERM`

Các báo hiệu có thể dung số hay tên

Không thể dung báo hiệu SIGKILL và SIGSTOP để bắt

Shell Programming

❑ LỆNH SHELL

Lệnh typeset

- Lệnh typeset của ksh được dùng để định nghĩa các biến

Khuôn dạng: **typeset** [+attributes] [name[=value]]

Ý nghĩa thuộc tính

- Đặt thuộc tính sau khi đặt giá trị
- + Bỏ đặt thuộc tính sau khi đặt giá trị
- A arr Mảng
- E n Số exp(n)
- F n Số phẩy động
- I n Số nguyên cơ số n
- l Đổi ký tự hoa thành chữ thường
- r Biến là chỉ đọc
- u Đổi các ký tự thường thành chữ hoa

Ví dụ:

```
$ typeset -i16 hexvalue
$ hexvalue=2091
$ echo $hexvalue
16#3487
```

Shell Programming

❑ LỆNH SHELL

Lệnh typeset

- Nhằm để nhanh chóng chuyển đổi hệ đếm bất kì (từ hệ đếm 2 đến hệ đếm 36) có thể sử dụng khuôn dạng sau:

`$echo $((base#number))`

Ví dụ :

```
$ echo $((16#ae09f))  
16652452
```

```
$ ./convbase.bsh  
Enter a value 100  
Original value is 100  
Value in hexa form is 16#64  
Value in octal form is 8#144  
Value in binary form is 2#1100100
```

```
#!/bin/ksh  
print -n "Enter a value "  
read n  
integer -i10 value=$n  
print "Original value is $value"  
typeset -i16 value  
print "Value in hexa form is $value"  
typeset -i8 value  
print "Value in octal form is $value"  
typeset -i2 value  
print "Value in binary form is $value"
```

Shell Programming

❑ LỆNH SHELL

▪ Lệnh điều kiện *if*

- Lệnh *if* /*elif*/*else* cho ra quyết định nhiều chiều.
 - + Nếu *if* không thành công, *elif* sẽ được kiểm tra.
 - + Nếu *if* thành công, các lệnh sau *then* thực thi.
 - + Nếu *elif* không thành công, *elif* nữa được chọn.
 - + Nếu không lệnh nào thành công, các lệnh khác sẽ được thực hiện.

Khuôn dạng

```
if command
    then command(s)
el if command
    then
        commands(s)
el if command
    then
        command(s)
else
        command(s)
fi
```

Shell Programming

❑ LỆNH SHELL

▪ Ví dụ 1 (if)

```
$ cat vui  
#!/bin/sh  
# Scriptname: qtuan  
echo -n "How old are you? "  
read age  
if [ Sage -lt 0 -o Sage -gt 120 ]  
then  
    echo "Welcome to our planet! "  
    exit 1  
fi  
if [ Sage -ge 0 -a Sage -lt 13 ]  
then  
    echo "A child is a garden of verses"
```

Shell Programming

❑ LỆNH SHELL

▪ Ví dụ (tiếp)

```
elif [ Sage -ge 13 -a Sage -lt 20 ]
then
    echo "You without a cause"
elif [ Sage -ge 20 -a Sage -lt 30 ]
then
    echo "You got the world by the tail!!"
elif [ Sage -ge 30 -a Sage -lt 40]
then
    echo "Thirty something.."
else
    echo "Sorry I asked"
fi
```

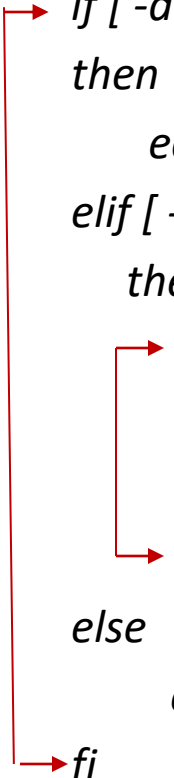
```
$ ./vui
How old are you? 200
Welcome to our planet!
$ ./vui
How old are you? 13
You without a cause
$ ./vui
How old are you? 55
Sorry I asked
```

Shell Programming

❑ LỆNH SHELL

▪ Ví dụ 2

```
#!/bin/sh
file=./testing
if [ -d $file ]
then
    echo "Sfile is a directory"
elif [ -f $file ]
then
    if [ -r $file -a -w $file -a -x $file ]
    then
        # nested if command
        echo "You have read, write, and execute permission on $file."
    fi
else
    echo "Sfile is neither a file nor a directory."
fi
```



```
# Script fn
name = qtuan
if grep "Sname" databasefile > /dev/null 2>&1
then
    :
else
    echo "$1 not found in databasefile"
    exit 1
fi
```

Shell Programming

❑ LỆNH SHELL

▪ Lệnh **case**

- Lệnh rẽ nhánh nhiều đường được sử dụng thay thế cho lệnh if/elif.
- Giá trị của biến case so khớp với `value1`, `value2`, v.v., cho đến khi tìm thấy khớp.
- Khi một giá trị khớp với biến trường hợp, các lệnh theo sau giá trị được thực thi cho đến khi đạt đến dấu chấm phẩy kép. Sau đó, việc thực thi bắt đầu sau từ `esac` (viết hoa chữ thường).
- Nếu biến trường hợp không khớp, chương trình sẽ thực hiện các lệnh sau dấu `*)`, giá trị mặc định, cho đến khi `;;` hoặc `esac` đạt được.

Khuôn dạng:

```
case variable in  
value1)  
    command(s)  
    ;;  
value2)  
    command(s)  
    ;;  
*) command(s)  
    ;;  
esac
```


Shell Programming

❑ LỆNH SHELL

- Lệnh **case**

```
#!/bin/sh
```

```
# Scriptname: colors
```

```
echo -n "Which color do you like?"
```

```
read color
```

```
case "$color" in
```

```
  [Bb]l??)
```

```
    echo I feel Scolor
```

```
    echo The sky is Scolor ; ;
```

```
  [Gg]ree*)
```

```
    echo Scolor is for trees echo Scolor is for seasick ; ;
```

```
  red | orange)    # The vertical bar means "OR"
```

```
    echo Scolor is very warm! ; ;
```

```
  8 *)
```

```
    echo No such color as Scolor ; ;
```

```
esac
```

```
echo "Out of case"
```

Shell Programming

❑ LỆNH SHELL

- Lệnh **case**

(The .profile file)

```
echo "Select a terminal type: "
```

```
cat « ENDIT
```

```
1) vt 120
```

```
2) wyse50
```

```
3) sun
```

```
ENDIT
```

```
read choice
```

```
case "Schoice" in
```

```
1) TERM=vt120
```

```
export TERM ; ;
```

```
2) TERM=wyse50
```

```
export TERM ; ;
```

```
3) TERM=sun
```

```
export TERM ; ;
```

```
esac
```

```
echo "TERM is $TERM."
```

Shell Programming

❑ LỆNH SHELL

▪ Lệnh select: Creating Menus

- Vòng select là vòng lặp đặc biệt được dùng để tạo các menus. Một menu là danh mục của các tùy chọn được hiện trên monitor.

Cú pháp: **select** variable in menu_opt1 menu_opt2...menu_optn

do

case \$variable in

menu_opt1) command1;;

menu_opt2) command2;;

menu_optn) commandn;;

esac

done

Shell Programming

❑ LỆNH SHELL

▪ Lệnh select: Creating Menus

Ví dụ: *demomenu2*

```
#!/bin/bash
```

```
select k in month year quit
do
```

```
case $k in
```

```
month) cal;;
```

```
year) yr=`date +%Y`
```

```
cal $yr;;
```

```
quit) echo Bye Bye
```

```
exit;;
```

```
*)echo Please try again
```

```
esac
```

```
done
```

```
$/demomenu2
```

```
1) month
```

```
2) year
```

```
3) quit
```

```
#? 1
```

```
March 2012
```

S	M	Tu	W	Th	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

```
#?2
```

```
2012
```

Jan							Feb							Mar						
S	M	Tu	W	Th	F	S	S	M	Tu	W	Th	F	S	S	M	Tu	W	Th	F	S
1	2	3	4	5	6	7					1	2	3	4				1	2	3
8	9	10	11	12	13	14	5	6	7	8	9	10	11	4	5	6	7	8	9	10
15	16	17	18	19	20	21	12	13	14	15	16	17	18	11	12	13	14	15	16	17
22	23	24	25	26	27	28	19	20	21	22	23	24	25	18	19	20	21	22	23	24
29	30	31					26	27	28	29				25	26	27	28	29	30	31

```
-----
```

Shell Programming

❑ LỆNH SHELL

▪ Lệnh **for**

- Thực hiện các lệnh một số lần hữu hạn trên một danh sách các mục.
- Theo sau lệnh **for** là một biến do người dùng xác định, từ khóa **in** và danh sách các từ.
- + Lần đầu tiên trong vòng lặp, từ đầu tiên từ danh sách từ được gán cho biến, sau đó được chuyển ra khỏi danh sách. Khi từ được gán cho biến, phần nội dung của vòng lặp được nhập và các lệnh giữa từ khóa **do** và **done** được thực thi.
- + Lần tiếp theo trong vòng lặp, từ thứ hai được gán cho biến, v.v.

Khuôn dạng:

```
for variable in wordlist  
do  
    command(s)  
done
```

Shell Programming

❑ LỆNH SHELL

▪ Lệnh **for**

(The Script)

```
#!/bin/sh
```

```
# Scriptname: forloop
```

```
for pal in Cuong Chien Binh Thông
```

```
do
```

```
echo "Hi $pal"
```

```
done
```

```
echo "Out of loop"
```

(The Command Line)

```
$ cat mylist
```

```
Cuong
```

```
Chien
```

```
Binh
```

```
Thông
```

(The Script)

```
#!/bin/sh
```

```
# Scriptname: mailer
```

```
for person in `cat mylist`
```

```
do
```

```
mail $person < letter
```

```
echo $person was sent a letter.
```

```
done
```

```
echo "The letter has been sent"
```

Shell Programming

❑ LỆNH SHELL

- Ví dụ **for**

(The Script)

```
#!/bin/sh
```

```
# Scriptname: backup
```

```
# Purpose: Create backup files
```

```
dir=/home/qtuan/tam
```

```
for file in memo[1-5]
```

```
do
```

```
if [ -f $file ]
```

```
then
```

```
cp $file $dir/$file.bak
```

```
echo "$file is backed up in $dir"
```

```
fi
```

```
done
```

(The Output)

```
memo1 is backed up in /home/qtuan/tam  
memo2 is backed up in /home/jqtuan/tam  
memo3 is backed up in /home/qtuan/tam  
memo4 is backed up in /home/qtuan/tam  
memo5 is backed up in /home/qtuan/tam
```

Shell Programming

❑ LỆNH SHELL

▪ Lệnh **while**

- Lệnh while đánh giá lệnh ngay sau nó và nếu nó trạng thái 0 là thoát ,
- Các lệnh trong phần thân của while (lệnh giữa do và done) được thực thi. Khi đạt đến từ khóa done, quyền điều khiển được đưa trở lại đầu vòng lặp và lệnh while sẽ kiểm tra lại trạng thái thoát của lệnh. Cho đến khi trạng thái thoát của lệnh được đánh giá bởi while trở thành khác không, vòng lặp tiếp tục.

Khuôn dạng:

```
while command  
  do  
      command(s)  
  done
```


Shell Programming

❑ LỆNH SHELL

- Ví dụ **while**

(The Script)

```
#!/bin/sh /
```

```
# Scriptname: num
```

```
num=0 # Initialize num
```

```
while [ $num -lt 10 ] # num with test command
```

```
do
```

```
    echo -n $num
```

```
    num= `expr $num + 1` # Increment num done
```

```
done
```

```
    echo "\n After loop exits, continue running here"
```

(The Output)

```
0123456789
```

```
After loop exits, continue running here
```

Shell Programming

❑ LỆNH SHELL

- Ví dụ **while** *(The Script)*

```
#!/bin/sh
# Scriptname: sayit
echo Type q to quit.
go=start
while [ -n "$go" ]
do
    echo -n I love you.
    read word
    if [ "Sword" = q -o "Sword" = Q ]
    then
        echo "I'll always love you!"
        go=
    fi
done
```

(The Output)

\$ sayit

Type q to quit.

I love you. <- When Enter key,

I love you.

I love you.

I love you.

I love you. <- q

*I'll always love **you!***

\$

Shell Programming

❑ LỆNH SHELL

■ Ví dụ **while**

(The Script)

```
#!/bin/sh
```

```
# Scriptname: quiz
```

```
echo "Who was the chief defense lawyer in the OJ case?"
```

```
read answer
```

```
while [ "Sanswer" != "Nam" ]
```

```
do
```

```
    echo "Wrong try again!"
```

```
    read answer
```

```
done
```

```
echo You got it!
```

(The Output)

```
$ quiz
```

```
Who was the chief defense lawyer in the OJ case? Binh
```

```
Wrong try again!
```

```
Who was the chief defense lawyer in the OJ case? Chau
```

```
Wrong try again!
```

```
Who was the chief defense lawyer in the OJ case? Nam
```

```
You got it!
```

Shell Programming

❑ LỆNH SHELL

▪ Lệnh **until**

- Được sử dụng giống như lệnh **while**, nhưng chỉ thực hiện các câu lệnh lặp nếu lệnh sau **until** là fail; nghĩa là,
- Nếu lệnh sau **until** trả về trạng thái là nonzero. Sau khi các lệnh thực hiện đạt đến từ khóa **done**, quyền điều khiển được đưa trở lại đầu vòng lặp và lệnh **until** kiểm tra lại trạng thái của lệnh tiếp theo lần nữa.
- Cho đến khi trạng thái của lệnh sau **until** được đánh giá trở thành 0, sau khi các lệnh thực hiện đạt đến từ khóa **done** vòng lặp sẽ thoát ra

Khuôn dạng

```
until command  
do  
    command(s)  
done
```

Shell Programming

❑ LỆNH SHELL

- Ví dụ until

(The Script)

#!/bin/sh

until who | grep qtuan

do

sleep 5

done

talk qtuan@vnu.edu.vn

Shell Programming

❑ LỆNH SHELL

▪ Ví dụ until

(The Output)

\$ hour

Good morning!

Good morning!

.....

Lunch time.

.....

Siesta time.

.....

Good night.

.....

(The Script)

#!/bin/sh

Scriptname: hour

hour=1

until [\$hour -gt 24]

do

case "Shour" in

[0-9] | 1[0-1]) echo "Good morning!"

;;

12) echo "Lunch time."

;;

[3-7]) echo "Siesta time."

;;

**) echo "Good night."*

;;

esac

hour= `expr Shour + 1`

done

Shell Programming

❑ LỆNH SHELL

▪ Lệnh Break

- Được sử dụng để buộc thoát ngay lập tức từ một vòng lặp, (không phải từ một chương trình).
- Sau khi lệnh break được thực hiện, điều khiển bắt đầu sau từ khóa done.
- Lệnh break gây ra một lỗi ra khỏi vòng lặp trong cùng, vì vậy nếu có nhiều vòng lặp lồng nhau, lệnh break sẽ lấy một số làm đối số, cho phép thoát ra khỏi một vòng lặp cụ thể bên ngoài.

Khuôn dạng

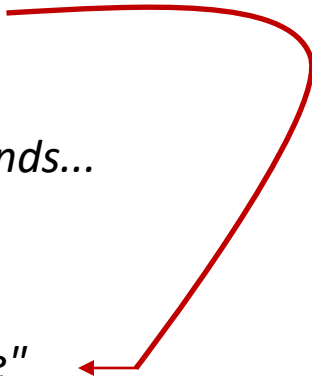
break [n]

Shell Programming

❑ LỆNH SHELL

▪ Ví dụ break

```
#!/bin/sh
while true;
do
    echo Are you ready to move on\?
    read answer
    if [ "Sanswer" = Y -o "Sanswer" = y ]
    then
        break
    else
        commands...
    fi
done
print "Here we are"
```



Shell Programming

❑ LỆNH SHELL

▪ **Lệnh continue: SKIPPING STATEMENTS IN LOOPS**

- Trả về quyền điều khiển ở đầu vòng lặp,
- Nếu một điều kiện nào đó trở thành true, tất cả các lệnh bên dưới tiếp tục sẽ bị bỏ qua.
- Nếu lồng trong một số vòng lặp, lệnh continue trả về điều khiển cho vòng lặp trong cùng.
- Nếu có một số của nó, thì điều khiển có thể được bắt đầu ở đầu bất kỳ vòng lặp nào.

Khuôn dạng

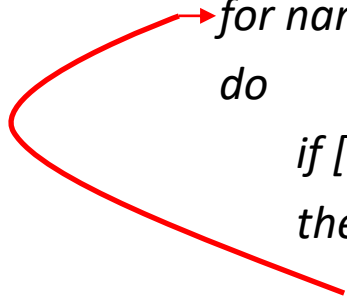
continue [n]

Shell Programming

❑ LỆNH SHELL

▪ Ví dụ continue

```
(The Script)
#!/bin/sh
# Scriptname: mailem
# Purpose: To send a list
for name in `cat mailjist`
do
    if [ "Sname" = "Thuy" ];
    then
        continue
    ---
    else
        mail Sname < memo
    fi
done
```



(The mailing List)

\$ cat mail.list

Thanh

Thuy

Lan

Hung

Shell Prog

❑ LỆNH SHELL

▪ Lệnh continue

```
$ months
Processing the month of Jan. Okay?
Processing the month of Feb. Okay? Y
Process week 1 of Feb? y
Now processing week 1 of Feb.
Done processing...
Processing the month of Feb. Okay? y
Process week 2 of Feb? y
Now processing week 2 of Feb.
Done processing...
Processing the month of Feb. Okay? n
Processing the month of Mar. Okay? n
Processing the month of Apr. Okay? n
Processing the month of May. Okay? n
```

```
#!/bin/sh
```

```
for month in Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
do
    for week in 1 2 3 4
    do
        echo -n "Processing the month of Smonth. Okay? "
        read ans
        if [ "Sans" = n -o -z "Sans" ]
        then
            ----
            continue 2
        else
            echo -n "Process week Sweek of Smonth? "
            read ans
            if [ "Sans" = n -o -z "Sans" ] /
            then
                continue
            else
                echo "Now processing week Sweek of Smonth."
                sleep 1
            echo "Done processing..." // fi fi done done
```

Shell Programming

❑ LỆNH SHELL

Khuôn dạng

```
function.name () { commands ; commands; }
```

▪ Hàm

- Các hàm sử dụng để mô-đun hóa chương trình và làm cho nó hiệu quả hơn
- Nguyên tắc sử dụng hàm
 1. Shell tìm kiếm các hàm đã có sẵn dưới dạng hàm, chương trình ..
 2. Hàm phải được khai báo trước khi dùng
 3. Các biến trong hàm sẽ biến mất khi thoát khỏi hàm
 4. Lệnh return trả trạng thái thoát cuối cùng không được vượt quá giá trị 255.
 5. Hàm chỉ tồn tại trong shell nơi được định nghĩa; không thể xuất sang shell con.
 6. Để liệt kê hàm và xác định hãy dùng lệnh **set**
 7. Nếu hàm trong file khác có thể tải về file hiện hành

Shell Programming

❑ LỆNH SHELL

▪ Ví dụ Hàm

```
dir () {  
echo "Directories: " ;  
ls -l | awk '/^d/ {print $NF}' ;  
}
```

- Tên của hàm là *dir*. Các dấu ngoặc đơn trống là cú pháp cần thiết để đặt tên cho hàm nhưng không có mục đích nào khác.
- Các lệnh trong dấu ngoặc nhọn ({ }) sẽ được thực hiện khi nhập *dir*.
- Mục đích của hàm là chỉ liệt kê các thư mục con bên dưới thư mục làm việc hiện tại.
- Các khoảng trắng bao quanh dấu ngoặc nhọn là bắt buộc.

Shell Programming

❑ LỆNH SHELL

▪ Ví dụ Hàm

(Using the return command - The Script)

```
#!/bin/sh
```

```
# Scriptname: do_increment
```

```
increment () {
```

```
    sum= `expr $1 + 1`
```

```
    return $sum      # Return the value of sum to the script.
```

```
}
```

```
echo -n "The sum is "
```

```
increment 5      # Call function increment; pass 5 as a parameter.
```

```
    # 5 becomes $1 for the increment function.
```

```
echo $?          # The return value is stored in $?
```

```
echo $sum        # The variable "sum" is known to the function,
```

```
    # and is also known to the main script.
```

(The Output)
The sum is 6
6

Thanks