

Bài thực hành buổi 3

LẬP TRÌNH CẤU TRÚC VÀ VÒNG LẶP TRONG SHELL

Nội dung thực hành

- Lệnh if
- Lệnh for
- Lệnh while
- Lệnh case

Trước khi tìm hiểu về các cấu trúc lập trình và vòng lặp trong shell của Linux, chúng ta đề cập một chút về kiểu boolean trong shell.

Trong shell của Linux giá trị True và False được thể hiện tương ứng bằng 2 giá trị không (0 – zero) và khác không (non-zero).

Ví dụ:

$5 > 2 \rightarrow 0$ (True)

$5 < 2 \rightarrow 1$ (False)

1. Câu lệnh if

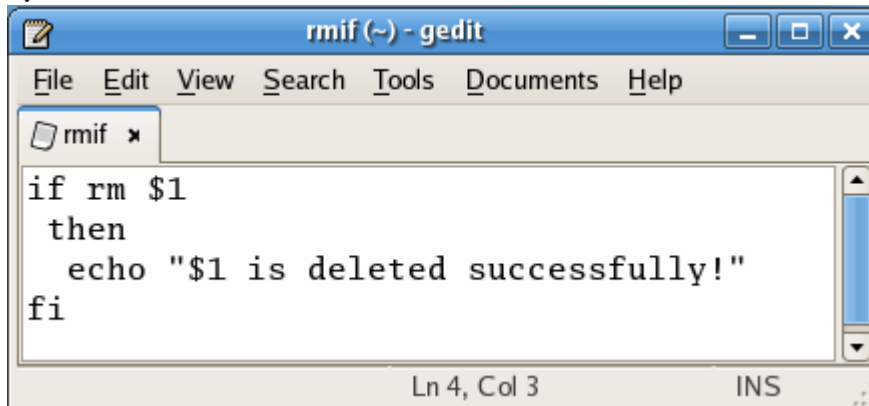
Câu lệnh if được dùng để thiết lập quyết định (lệnh rẽ nhánh) trong shell script.

Cú pháp:

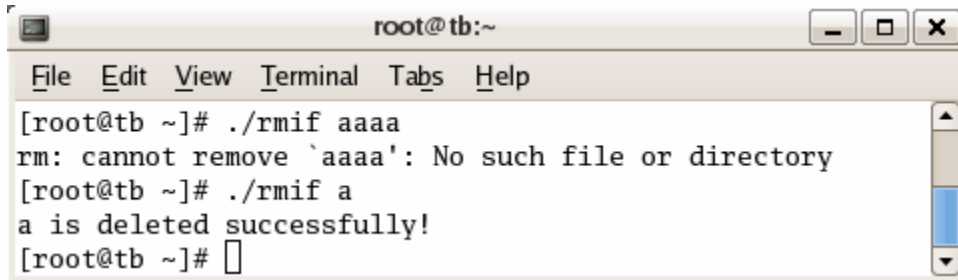
```
if condition
then
    command1
    ...
fi
```

Lệnh *command1* được thực thi khi điều kiện *condition* là True (0), ngược lại lệnh tiếp sau lệnh if (sau từ khóa fi) được thực hiện.

Ví dụ:



Lưu đoạn script trên với tên file là `rmif`, thực hiện `chmod` và thực thi. Kết quả thực thi như sau:



```
root@tb:~  
File Edit View Terminal Tabs Help  
[root@tb ~]# ./rmif aaaa  
rm: cannot remove `aaaa': No such file or directory  
[root@tb ~]# ./rmif a  
a is deleted successfully!  
[root@tb ~]#
```

Lần đầu thực thi `rmif` với tham số `aaaa` sẽ nhận được một thông báo lỗi “No such file or directory”. Lần thứ 2 thực hiện với tham số `a` thì nhận được một thông báo “a is deleted successfully”.

Như vậy đoạn script trên thực hiện xóa file với tên file được truyền vào từ tham số của script. Nếu tên file không tồn tại thì lệnh `rm` không thành công hay nói cách khác lệnh sẽ trả về exit status là khác không có nghĩa là `False`. Ngược lại exit status bằng không (0) có nghĩa là `True`, khi đó lệnh `echo` được in ra màn hình.

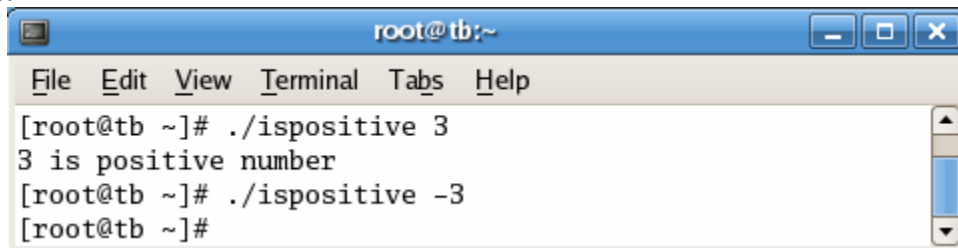
2. Lệnh `test` hoặc `[expr]`

Lệnh `test` hoặc `[expr]` dùng để kiểm tra một biểu thức logic là `True` hay `False`. Nếu biểu thức có giá `True` lệnh `test` hoặc `[expr]` sẽ trả về giá trị 0, ngược lại trả về giá trị khác không.

Khảo sát đoạn script sau:

```
if test $1 -gt 0  
then  
    echo "$1 is positive number"  
fi
```

Lưu script trên với tên file là `ispositive`, thực hiện `chmod` và thực thi. Kết quả như sau:



```
root@tb:~  
File Edit View Terminal Tabs Help  
[root@tb ~]# ./ispositive 3  
3 is positive number  
[root@tb ~]# ./ispositive -3  
[root@tb ~]#
```

Đoạn script trên nhận vào một tham số, sau đó kiểm tra xem tham số có lớn hơn số 0 hay không nhờ vào lệnh `test`. Nếu lớn hơn 0 thì in câu “\$1 is positive number”, ngược lại không in gì cả.

Lệnh test và [expr] hoạt động với các loại dữ liệu như: Integer, file và chuỗi ký tự. Dưới đây là các phép toán được dùng trong lệnh test hoặc [expr]

Các phép toán so sánh

Phép toán	Ý nghĩa	Lệnh test sử dụng với if	[expr] sử dụng với if
-eq	So sánh bằng (==)	if test 5 -eq 6	if [5 -eq 6]
-ne	Không bằng (!=)	if test 5 -ne 6	if [5 -ne 6]
-lt	Nhỏ hơn (<)	if test 5 -lt 6	if [5 -lt 6]
-le	Nhỏ hơn hoặc bằng (<=)	if test 5 -le 6	if [5 -le 6]
-gt	Lớn hơn (>)	if test 5 -gt 6	if [5 -gt 6]
-ge	Lớn hơn hoặc bằng (>=)	if test 5 -ge 6	if [5 -ge 6]

Phép toán so sánh chuỗi

Phép toán	Ý nghĩa	Lệnh test sử dụng với if	[expr] sử dụng với if
=	So sánh bằng	if test "aa" = "bb"	if ["aa" = "bb"]
!=	So sánh không bằng	if test "aa" != "bb"	if ["aa" != "bb"]

Sử dụng lệnh test hoặc [expr] với file và thư mục

test	Ý nghĩa
-s file	Trả về True (0) nếu file không rỗng
-f file	Trả về True (0) nếu là file (không phải là thư mục)
-d dir	Trả về True (0) nếu là thư mục
-w file	Trả về True (0) nếu file có thể viết
-r file	Trả về True (0) nếu file chỉ đọc
-x file	Trả về True (0) nếu file có thể thực thi

Phép toán logic

Phép toán	Ý nghĩa
! expression	Phủ định biểu thức expression
expr1 -a expr2	Phép toán AND 2 biểu thức
expr1 -o expr2	Phép toán OR 2 biểu thức

3. Lệnh if ... else ... fi

Cú pháp:

```
if condition
then
    # condition trả về 0 (true)
    # thực thi tất cả các câu lệnh cho đến từ khóa else
else
    # condition là sai
    # thực thi tất cả các câu lệnh cho đến từ khóa fi
fi
```

Câu lệnh if ... else ... fi sẽ thực hiện các câu lệnh trong phần *then* nếu điều kiện đúng, ngược lại sẽ thực thi các câu lệnh trong phần *else*.

Ví dụ:

```
if test $1 -gt 0
then
    echo "$1 is positive"
else
    echo "$1 is negative"
fi
```

Khi thực thi đoạn script trên sẽ kiểm tra tham số đầu vào, nếu lớn hơn 0 sẽ in câu thông báo “\$1 is positive” ngược lại sẽ in câu “\$1 is negative”.

4. Lệnh if-then-else nhiều cấp

Cú pháp

```
if condition
then
    # condition trả về 0 (true)
    # thực thi tất cả các câu lệnh cho đến từ khóa elif
elif condition1
then
    # condition1 trả về 0 (true)
    # thực thi tất cả các câu lệnh cho đến từ khóa elif
elif condition2
then
    # condition2 trả về 0 (true)
    # thực thi tất cả các câu lệnh cho đến từ khóa else
else
    # Tất cả condition, condition1, condition2 là False
    # thực thi tất cả các câu lệnh cho đến từ khóa fi
fi
```

Câu lệnh if-then-else nhiều cấp cũng được sử dụng giống như lệnh if-then-else

5. Lệnh lặp for:

Lệnh lặp for trong shell của Linux có 2 cú pháp

Cú pháp 1:

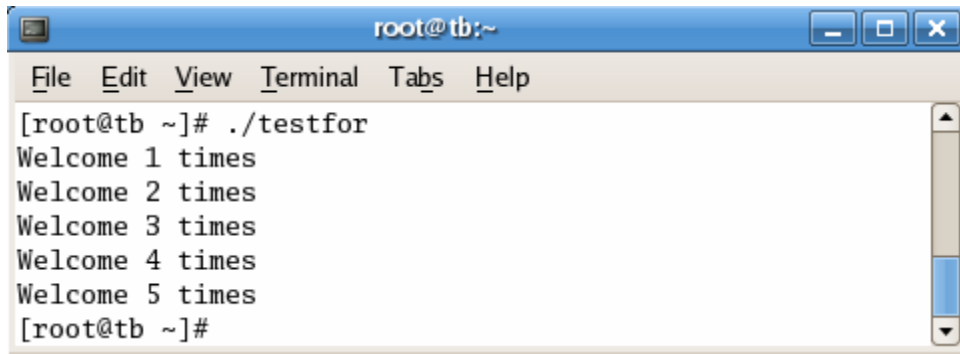
```
for { variable name } in { list }
do
    command
done
```

Biến *name* sẽ được lần lượt gán từng phần tử trong danh sách *list* và thực hiện lệnh *command*. Để hiểu rõ hơn xem ví dụ dưới đây:

Ví dụ:

```
for i in 1 2 3 4 5
do
    echo "Welcome $i times"
done
```

Lưu lại với tên testfor và thực thi ta có kết quả như sau:



```
root@tb:~  
File Edit View Terminal Tabs Help  
[root@tb ~]# ./testfor  
Welcome 1 times  
Welcome 2 times  
Welcome 3 times  
Welcome 4 times  
Welcome 5 times  
[root@tb ~]#
```

Với kết quả trên ta thấy rằng biến *i* sẽ được lần lượt gán các giá trị 1, 2, 3, 4, 5 và mỗi lần như vậy vòng lặp *for* thực hiện lệnh *echo* “Welcome \$i times”.

Cú pháp 2:

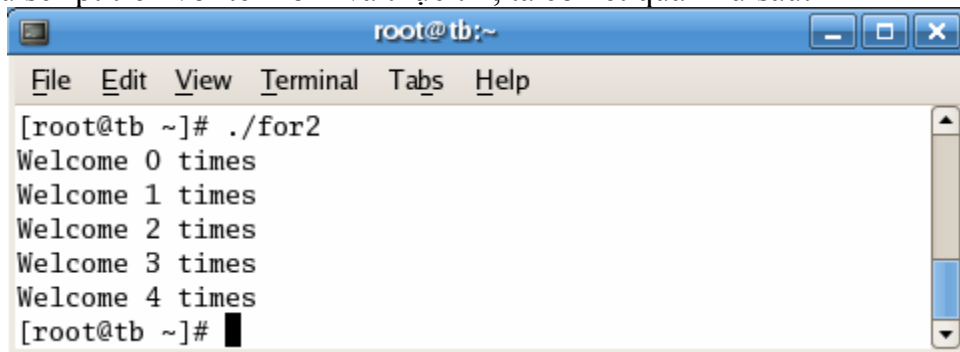
```
for (( expr1; expr2; expr3 ))  
do  
    .....  
    ...  
    # Thực hiện tất cả các câu lệnh giữa do và done  
    # cho đến khi expr2 là TRUE  
done
```

Với cú pháp này trước tiên biểu thức *expr1* sẽ được thực thi (thông thường được dùng để khởi tạo giá trị cho biến vòng lặp). Sau đó các câu lệnh giữa *do* và *done* sẽ được thực hiện lặp đi lặp lại cho đến khi biểu thức *expr2* có giá trị True. Sau mỗi lần lặp biểu thức *expr3* sẽ được thực thi (thông thường là làm thay đổi biến vòng lặp). Để thấy rõ hơn việc sử dụng vòng lặp này, xem ví dụ sau:

Ví dụ:

```
for (( i = 0 ; i <= 5; i++ ))  
do  
    echo "Welcome $i times"  
done
```

Lưu script trên với tên *for2* và thực thi, ta có kết quả như sau:

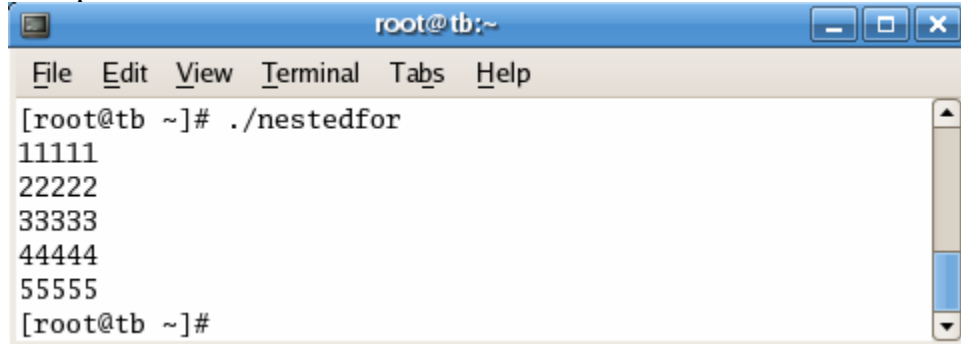


```
root@tb:~  
File Edit View Terminal Tabs Help  
[root@tb ~]# ./for2  
Welcome 0 times  
Welcome 1 times  
Welcome 2 times  
Welcome 3 times  
Welcome 4 times  
[root@tb ~]#
```

Vòng lặp *for* hoàn toàn có thể lồng nhau tạo thành vòng lặp phức tạp hơn. Dưới đây là một ví dụ sử dụng vòng lặp *for* lồng nhau.

```
for (( i = 1; i <= 5; i++ ))      ### Vòng lặp ngoài ###
do
    for (( j = 1 ; j <= 5; j++ )) ### Vòng lặp trong ###
    do
        echo -n "$i "
    done
    echo "" ##### in dòng mới ###
done
```

Lưu script trên với tên nestedfor và thực thi.



```
root@tb:~
File Edit View Terminal Tabs Help
[root@tb ~]# ./nestedfor
11111
22222
33333
44444
55555
[root@tb ~]#
```

6. Vòng lặp while

Cú pháp:

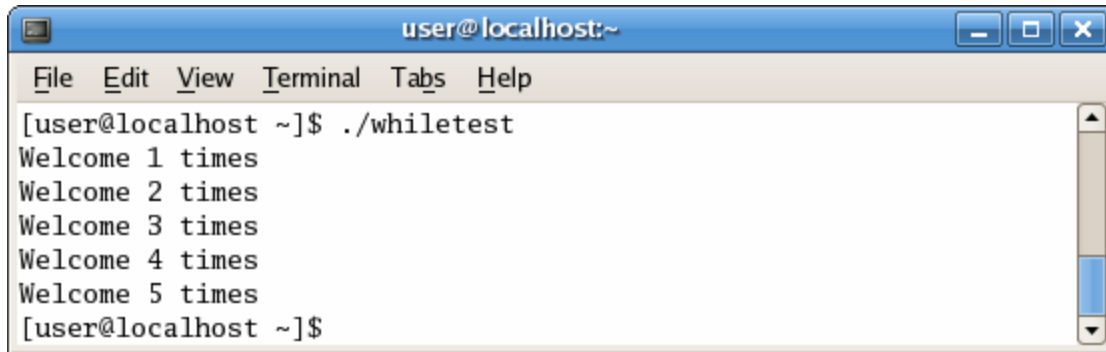
```
while [ condition ]
do
    command1
    command2
    command3
    ..
    ....
done
```

Các câu lệnh giữa do và done sẽ được thực hiện lặp đi lặp lại cho đến khi điều kiện *condition* là False.

Ví dụ:

```
i=1
while [ $i -le 5 ]
do
    echo "Welcome $i times"
    i=`expr $i + 1`
done
```

Đoạn chương trình này sẽ thực hiện in câu “Welcome \$i times” với i là biến integer. Kết quả khi thực thi đoạn script trên là:



```

user@localhost:~
File Edit View Terminal Tabs Help
[user@localhost ~]$ ./whilettest
Welcome 1 times
Welcome 2 times
Welcome 3 times
Welcome 4 times
Welcome 5 times
[user@localhost ~]$

```

4.7 Lệnh case

Cú pháp:

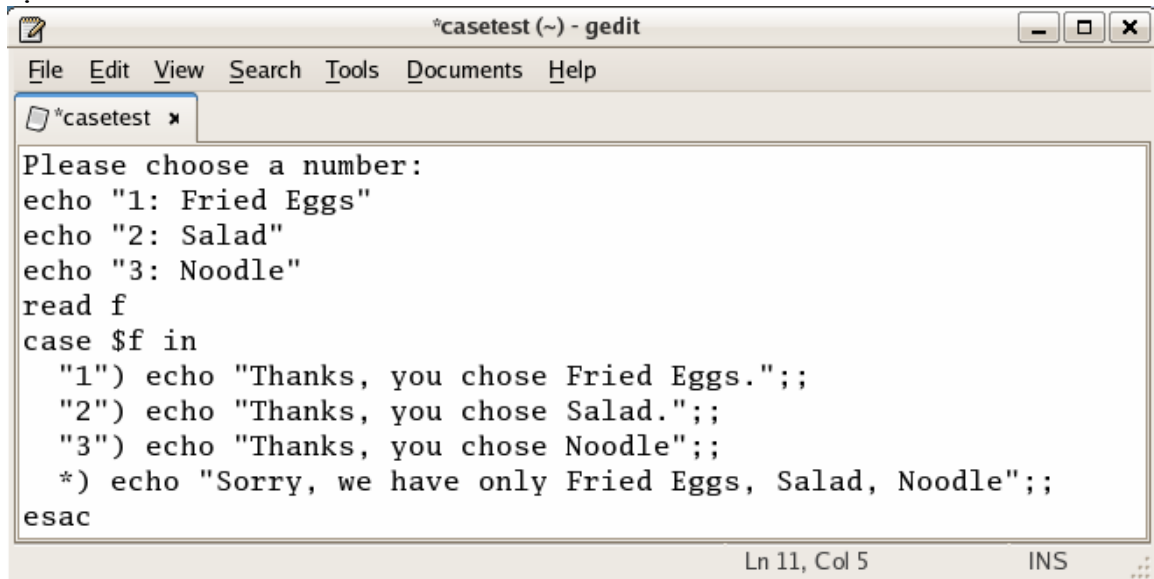
```

case $variable-name in
    pattern1) command
        ...
        command;;
    pattern2) command
        ...
        command;;
    patternN) command
        ...
        command;;
    *)
        command
        ...
        command;;
esac

```

Biến *variable-name* được so sánh với các *pattern1*, *pattern2*, *pattern3*... cho đến khi trùng khớp với một pattern nào đó. Khi đó sẽ thực thi các lệnh ngay sau pattern trùng khớp cho đến khi gặp 2 dấu chấm phẩy “;;”. Nếu không trùng khớp với pattern nào thì các câu lệnh của phần *** sẽ được thực thi.

Ví dụ:

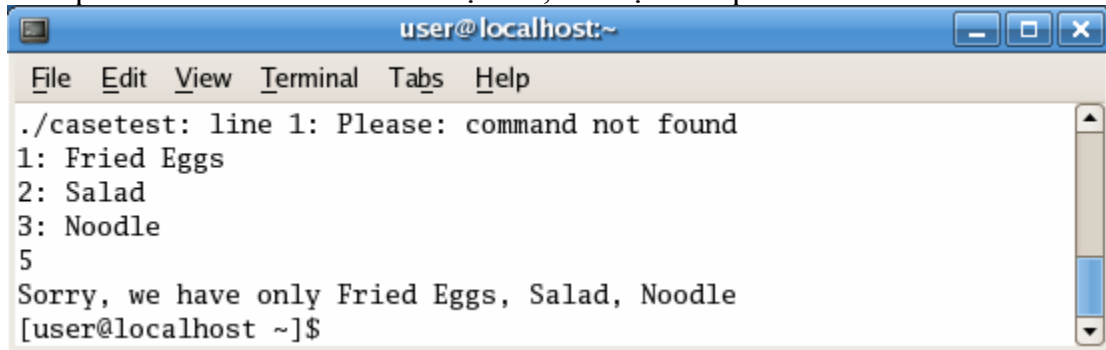


```

*casetest (~) - gedit
File Edit View Search Tools Documents Help
*casetest
Please choose a number:
echo "1: Fried Eggs"
echo "2: Salad"
echo "3: Noodle"
read f
case $f in
    "1") echo "Thanks, you chose Fried Eggs.";;
    "2") echo "Thanks, you chose Salad.";;
    "3") echo "Thanks, you chose Noodle";;
    *) echo "Sorry, we have only Fried Eggs, Salad, Noodle";;
esac
Ln 11, Col 5  INS

```

Lưu script trên với tên casetest và thực thi, ta được kết quả như sau:



```

user@localhost:~
File Edit View Terminal Tabs Help
./casetest: line 1: Please: command not found
1: Fried Eggs
2: Salad
3: Noodle
5
Sorry, we have only Fried Eggs, Salad, Noodle
[user@localhost ~]$
    
```

Khi được thực thi, đoạn script trên sẽ yêu cầu người dùng nhập vào một số tương ứng với món ăn được chọn. Sau đó tùy vào người dùng nhập vào số nào mà chương trình sẽ in ra câu tương ứng với món được chọn và kèm lời cảm ơn. Nếu chọn những số khác, chương trình sẽ in câu sorry.

Bài Tập

Bài 1: Thực hiện và kiểm tra tất cả các ví dụ phía trên.

Bài 2: Viết một script hiển thị trên màn hình hình sau:

```

*
**
***
****
*****
*****
*****
*****
*****
*****
*****
    
```

Và

```

1      2      3      4      5
1      2      3      4      5
1      2      3      4      5
1      2      3      4      5
1      2      3      4      5
    
```

Bài 3: Viết script thực hiện giải phương trình bậc nhất với hệ số a và b được người dùng nhập vào từ bàn phím.

Bài 4: Viết script thực hiện tổng dãy số nguyên từ 1 đến n , n được người dùng nhập vào từ bàn phím.

Bài 5: Viết một script hiển thị bảng cửu chương cho một số được nhập vào từ bàn phím. Ví dụ nhập vào 5, kết quả sẽ là:

$$1 \times 5 = 5$$

$$2 \times 5 = 10$$

$$3 \times 5 = 15$$

$$4 \times 5 = 20$$

$$5 \times 5 = 25$$

$$6 \times 5 = 30$$

$$7 \times 5 = 35$$

$$8 \times 5 = 40$$

$$9 \times 5 = 45$$

$$10 \times 5 = 50$$

Khi kết thúc sẽ hỏi người dùng có muốn tiếp tục không? Nếu tiếp tục thì lặp lại quá trình trên, ngược lại thì kết thúc.

Bài 6: Viết một script thực hiện tạo n thư mục cho n người dùng với tên thư mục là `user_i` ($i=1..n$). n được nhập vào từ bàn phím.