

Bài thực hành buổi 2

CƠ BẢN VỀ LẬP TRÌNH SHELL

Nội dung thực hành

- Tạo một shell script
- Thao tác biến
- Một số vấn đề liên quan khác

Trong phần này sinh viên sẽ được giới thiệu cơ bản các vấn đề về lập trình shell như tạo một shell script, thực thi một shell script và các vấn đề liên quan khác.

1 Viết một shell script như thế nào?

Để tạo một shell script ta thực hiện theo các bước sau:

B1: Sử dụng bất kỳ trình soạn thảo văn bản nào (text editor) để soạn thảo script, chẳng hạn gedit.

B2: Xác lập quyền thực thi cho shell script dùng lệnh chmod

Ví dụ: `chmod 755 script_file`

Lệnh này thiết lập quyền thực thi, đọc và thay đổi script cho chủ sở hữu và quyền đọc, thực thi cho những người dùng khác.

B3: Thực thi file script với một trong những cách thức sau:

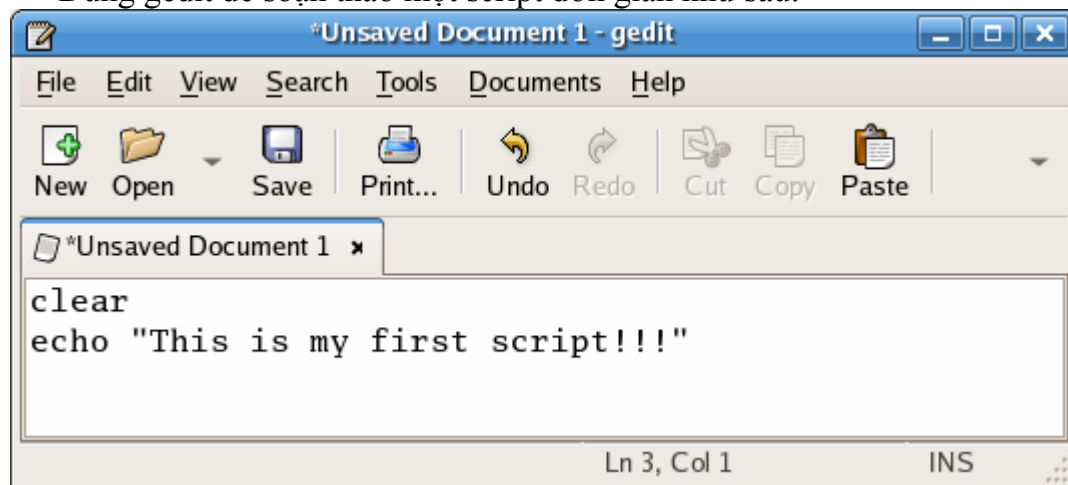
`bash script_file`

`sh script_file`

`./script_file`

Ví dụ sau đây mô tả việc tạo ra một script đơn giản và thực thi nó.

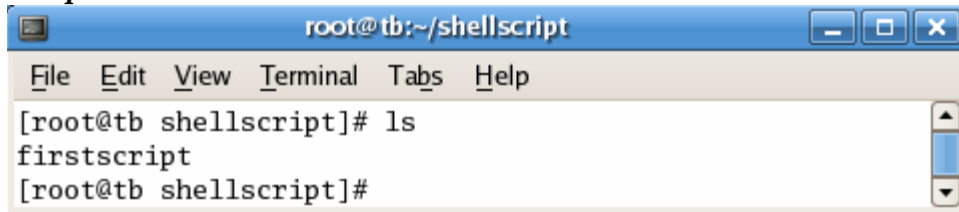
- Dùng gedit để soạn thảo một script đơn giản như sau:



Lệnh clear dùng để xóa màn hình hiện tại của shell

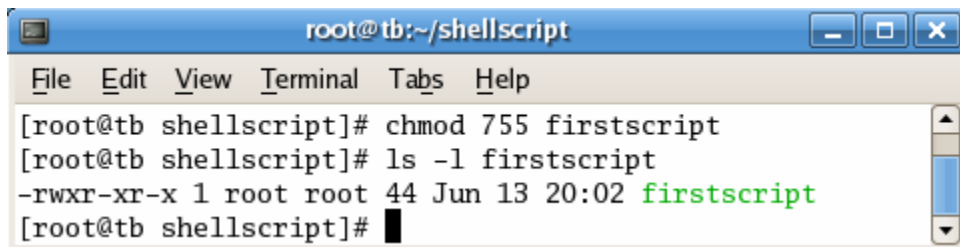
Lệnh echo “This is my first script!!!” dùng để xuất câu “This is my first script!!!” ra màn hình của shell.

- Click vào Save để tiến hành lưu file. Có thể lưu ở bất cứ thư mục nào và bất kỳ tên gì. Để cho tiện theo dõi trong ví dụ này tác giả lưu ở thư mục shellscript là thư mục con của thư mục home của người dùng hiện tại (root) với tên *firstscript*.



```
root@tb:~/shellscript
File Edit View Terminal Tabs Help
[root@tb shellscript]# ls
firstscript
[root@tb shellscript]#
```

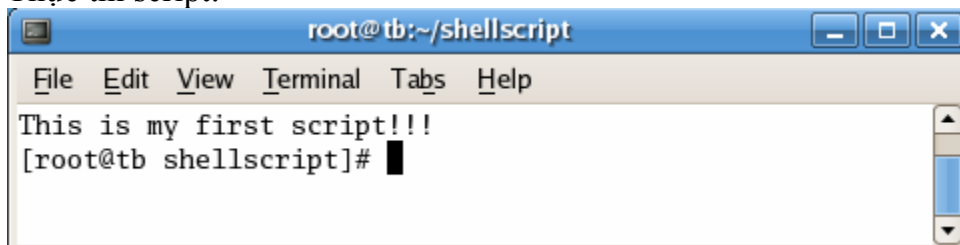
- Nếu tại thời điểm này chúng ta cho thực thi file firstscript sẽ nhận được thông báo lỗi tương tự như: Permission denied. Lý do là vì file script này chưa được thiết lập quyền thực thi.
- Thiết lập quyền thực thi cho script dùng lệnh chmod



```
root@tb:~/shellscript
File Edit View Terminal Tabs Help
[root@tb shellscript]# chmod 755 firstscript
[root@tb shellscript]# ls -l firstscript
-rwxr-xr-x 1 root root 44 Jun 13 20:02 firstscript
[root@tb shellscript]#
```

Sau khi dùng lệnh chmod 755 firstscript, quyền truy xuất trên file firstscript đã bị thay đổi (tên file cũng được đổi màu). Khi đó file firstscript có thể được thực thi bởi tất cả mọi người dùng.

- Thực thi script:



```
root@tb:~/shellscript
File Edit View Terminal Tabs Help
This is my first script!!!
[root@tb shellscript]#
```

❖ Làm thế nào shell có thể định vị được script cần thực thi?

Có 2 cách để shell có thể định vị được script cần thực thi

Cách 1: Sử dụng đường dẫn tuyệt đối để xác định cụ thể script cần thực thi nằm ở thư mục nào. Ví dụ:

```

root@tb:/
File Edit View Terminal Tabs Help
[root@tb /]# sh /root/shellscript/hello
Hello World!!
Today is
Tue Jun 13 20:45:32 ICT 2006
[root@tb /]#

```

Người dùng root trong ví dụ trên đang ở thư mục gốc “/” nhưng yêu cầu shell thực thi script hello trong thư mục /root/shellscript/.

Cách 2: Sử dụng biến môi trường PATH. Khai báo đường dẫn của script cần thực thi vào biến môi trường PATH.

Bài Tập

Hãy viết và thực thi script sau:

```

clear
echo "Hello $USER"
echo "Today is \c " ; date
echo "Number of user login : \c" ; who | wc -l
echo "Calendar"
cal
exit 0

```

2 Thao tác biến trong shell

Để lưu trữ và xử lý dữ liệu, dữ liệu cần được lưu vào bộ nhớ thường được gọi là RAM. Các ô nhớ trong RAM được định vị thông qua địa chỉ bộ nhớ. Đối với người lập trình việc truy xuất các dữ liệu này thường thông qua tên biến. Biến trong linux shell được phân thành 2 loại:

- Biến hệ thống (System variables): được tạo ra và quản lý bởi Linux, thường được đặt tên dưới dạng chữ HOA.
- Biến do người dùng định nghĩa (User defined variables - UDV): do người dùng tạo ra và quản lý, thường được đặt tên dưới dạng chữ thường.

Dưới đây là một số biến môi trường:

Biến môi trường	Ý nghĩa
BASH	Tên của shell
BASH_VERSION	Phiên bản của shell
COLUMNS	Số cột của màn hình
HOME	Thư mục chủ
LINES	Số dòng của màn hình
LOGNAME	Tên người dùng đã đăng nhập
OSTYPE	Loại của hệ điều hành
PATH	Danh sách các đường dẫn trong biến PATH
PS1	Cách thiết lập dấu nhắc
PWD	Thư mục làm việc hiện hành

SHELL	Tên của shell
-------	---------------

Để in nội dung của các biến trên, ta dùng lệnh echo như ví dụ sau:

\$echo \$BASH

\$echo \$PWD

3 Biến do người dùng định nghĩa

3.1 Cú pháp:

tên_biến=giá_trị_biến

Ví dụ:

\$ no=10

\$ n=100

3.2 Qui cách đặt tên biến

Tên biến phải bắt đầu bởi ký tự hoặc dấu gạch dưới “_” theo sau bởi một hoặc nhiều ký tự, ký số hoặc dấu gạch dưới “_”. Sau đây là vài tên biến hợp lệ: d100, a1_1, _head, _abc_.

3.3 Một số chú ý khi thao tác biến người dùng

- Chú ý: không được có khoảng trống 2 bên dấu bằng “=”, các ví dụ sau đây là sai cú pháp: n0 = 10 hay n= 100 hay n =100.
- Tên biến có phân biệt chữ HOA và chữ thường. Ví dụ *No* khác với *no*.
- Có thể định nghĩa một biến mà không có giá trị (NULL) như sau:
Ví dụ: vech= hoặc vech=""
- Để in giá trị một biến ta dùng dấu “\$” theo sau bởi tên biến.

Ví dụ:

\$echo \$n

\$echo “My name is \$LOGNAME”

Bài Tập

Chỉ ra các lỗi (nếu có) trong đoạn script sau:

```
myname=Vivek
myos = TroubleOS
myno=5
echo "My name is $myname"
echo "My os is $myos"
echo "My number is myno, can you see this number"
```

4 Một số vấn đề cơ bản khác trong lập trình shell

4.1 Sử dụng lệnh echo

Lệnh echo được sử dụng để hiển thị chuỗi hoặc giá trị của biến

Cú pháp: echo [option] [chuỗi, biến...]

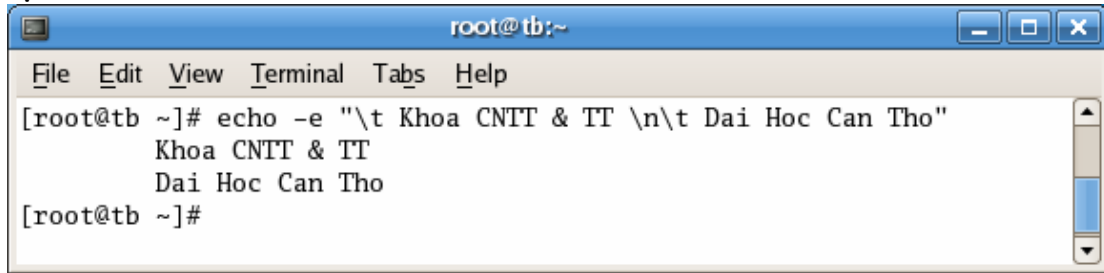
Các options:

- n: Dấu nhắc không xuống dòng sau khi hiển thị dòng cuối cùng
- e: Bật chế độ dịch các ký tự thoát

Các ký tự thoát:

- \a: phát ra âm thanh bíp
- \b: ký tự backspace
- \n: ký tự xuống dòng
- \t: ký tự tab
- \\: ký tự “\”

Ví dụ:



```
root@tb:~  
File Edit View Terminal Tabs Help  
[root@tb ~]# echo -e "\t Khoa CNTT & TT \n\t Dai Hoc Can Tho"  
    Khoa CNTT & TT  
    Dai Hoc Can Tho  
[root@tb ~]#
```

4.2 Thao tác biểu thức số học với expr

Để thao tác biểu thức số học trong shell ta dùng lệnh expr, với cú pháp như sau:

Cú pháp: `expr toán_hạng1 toán_tử toán_hạng2`

5 toán tử có thể dùng là:

- + : cộng
- : trừ
- * : nhân, khi sử dụng phép toán nhân phải dùng thêm ký tự thoát “*”.
- / : chia
- % : chia lấy dư

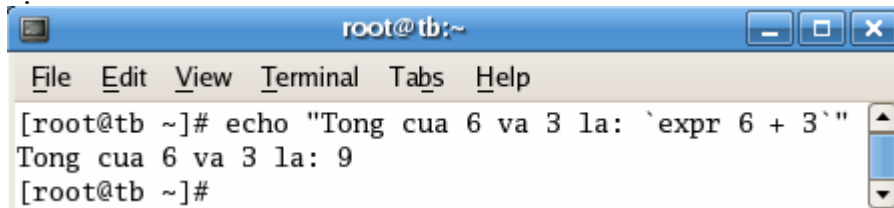
Ví dụ:

```
$ expr 1 + 3  
$ expr 2 - 1  
$ expr 10 / 2  
$ expr 20 % 3  
$ expr 10 \* 3
```

Chú ý: giữa toán_hạng1, toán_tử và toán_hạng2 phải có khoảng trắng.

expr có thể được sử dụng bên trong một chuỗi như là một biểu thức con, khi đó biểu thức sẽ được định giá trị và xuất kết quả cùng với chuỗi đó.

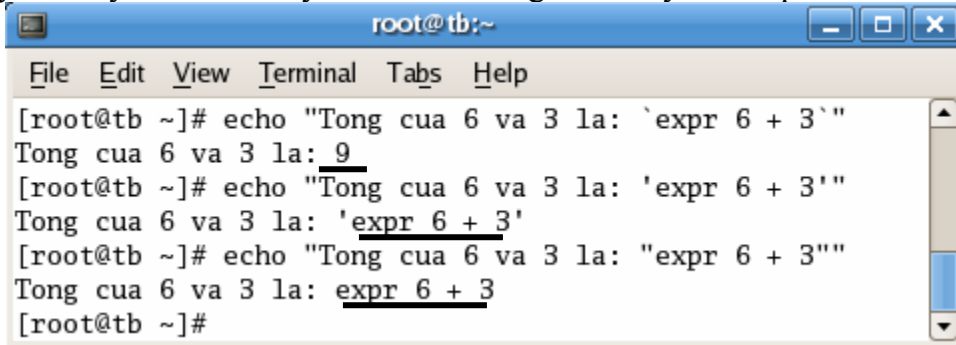
Ví dụ:



```
root@tb:~  
File Edit View Terminal Tabs Help  
[root@tb ~]# echo "Tong cua 6 va 3 la: `expr 6 + 3`"  
Tong cua 6 va 3 la: 9  
[root@tb ~]#
```

``expr 6 + 3`` đã được định giá là 9 và kết quả được xuất theo chuỗi “*Tong cua 6 va 3 la:*”. **Chú ý:** dấu nháy bao bọc ``expr 6 + 3`` là dấu nháy backquote (dấu nháy nằm cùng phím với dấu `~`, góc trên bên trái bàn phím).

Hãy xem kỹ ví dụ sau đây về việc sử dụng dấu nháy cho `expr`



```
root@tb:~  
File Edit View Terminal Tabs Help  
[root@tb ~]# echo "Tong cua 6 va 3 la: `expr 6 + 3`"  
Tong cua 6 va 3 la: 9  
[root@tb ~]# echo "Tong cua 6 va 3 la: 'expr 6 + 3'"  
Tong cua 6 va 3 la: 'expr 6 + 3'  
[root@tb ~]# echo "Tong cua 6 va 3 la: \"expr 6 + 3\""  
Tong cua 6 va 3 la: expr 6 + 3  
[root@tb ~]#
```

4.3 Trạng thái trả về sau khi kết thúc một lệnh (exit status)

Trong Linux khi một lệnh hoặc một script được thực thi xong nó sẽ trả về một giá trị trạng thái để xác định lệnh hoặc script đó thành công hay thất bại.

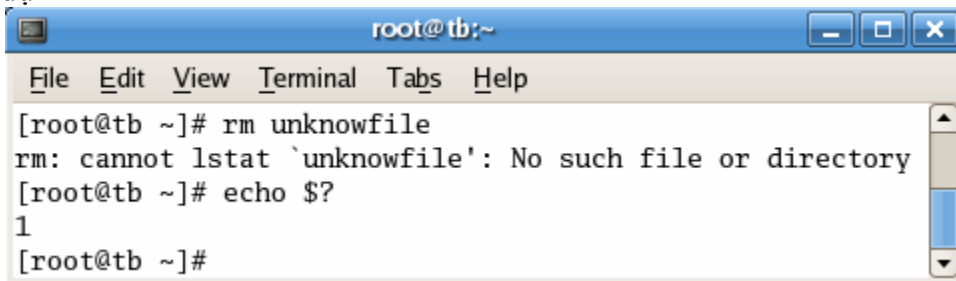
Giá trị đó là 0 thì lệnh được thực thi thành công

Giá trị đó khác 0 thì lệnh thực thi không thành công hoặc bị một lỗi nào đó

Giá trị này được gọi là giá trị trạng thái trả về (exit status)

Để xác định giá trị này shell cung cấp một biến đặc biệt là `$?`.

Ví dụ



```
root@tb:~  
File Edit View Terminal Tabs Help  
[root@tb ~]# rm unknowfile  
rm: cannot lstat `unknowfile': No such file or directory  
[root@tb ~]# echo $?  
1  
[root@tb ~]#
```

Ví dụ trên xóa một file không tồn tại (file có tên là unknowfile), do đó lệnh xóa không thành công. Và giá trị của exit status trả là 1.

Bài Tập:

Thử các lệnh sau đây và xem giá trị của exit status

```
$ expr 1 + 3  
$ echo $?  
  
$ echo Welcome  
$ echo $?  
$ wildwest canwork?  
$ echo $?  
$ date  
$ echo $?
```

4.4 Đọc dữ liệu từ bàn phím

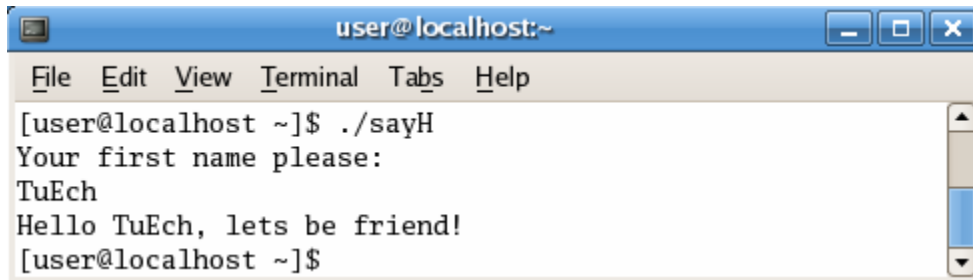
Để đọc dữ liệu nhập từ bàn phím và lưu vào biến ta dùng read có cú pháp sau:

read *biến*

Đoạn script sau yêu cầu người dùng nhập vào tên và được lưu vào biến fname, sau đó giá trị của biến này được in ra màn hình.

```
echo "Your first name please:"
read fname
echo "Hello $fname, Lets be friend!"
```

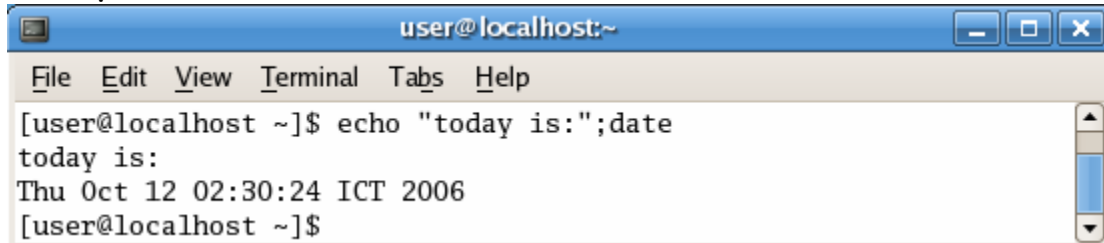
Lưu đoạn script trên với tên sayH, sau đó thực hiện chmod và thực thi. Ta được hình ảnh sau:



```
user@localhost:~
File Edit View Terminal Tabs Help
[user@localhost ~]$ ./sayH
Your first name please:
TuEch
Hello TuEch, lets be friend!
[user@localhost ~]$
```

4.5 Thực hiện nhiều lệnh trên một dòng

Để thực hiện nhiều lệnh trên một dòng ta dùng dấu chấm phẩy “;” để ngăn cách giữa các lệnh.



```
user@localhost:~
File Edit View Terminal Tabs Help
[user@localhost ~]$ echo "today is: ";date
today is:
Thu Oct 12 02:30:24 ICT 2006
[user@localhost ~]$
```

Lệnh echo và date được viết trên một dòng.

4.6 Truyền tham số cho shell script

Việc truyền tham số cho các shell script là rất cần thiết và làm cho script linh động hơn. Khi một shell script được truyền tham số thì tên của script và các tham số được lưu một cách tương ứng vào biến đặc biệt \$0, \$1, \$2...

Ví dụ:

\$myshell foo bar

Khi đó \$0 lưu giá trị myshell, \$1 lưu giá trị foo và \$2 lưu giá trị bar

Ngoài ra còn có một số biến đặc biệt khác như:

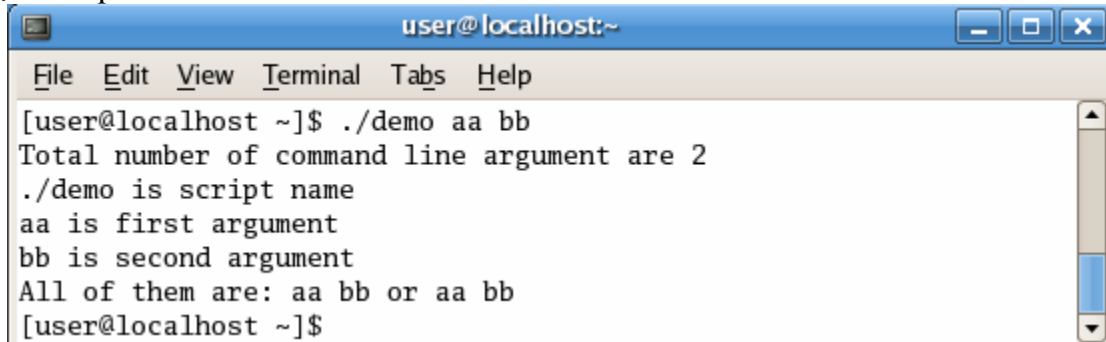
\$# : số lượng tham số truyền vào cho script

\$* hoặc @\$ lưu tất cả giá trị của các tham số

Đoạn script sau minh họa cho việc sử dụng tham số:

```
echo "Total number of command line argument are $#"  
echo "$0 is script name"  
echo "$1 is first argument"  
echo "$2 is second argument"  
echo "All of them are :- $* or $@"
```

Lưu đoạn script trên với tên file là demo và thực thi với 2 tham số là aa và bb ta được kết quả như sau:



```
user@localhost:~  
File Edit View Terminal Tabs Help  
[user@localhost ~]$ ./demo aa bb  
Total number of command line argument are 2  
./demo is script name  
aa is first argument  
bb is second argument  
All of them are: aa bb or aa bb  
[user@localhost ~]$
```

Bài tập:

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

Bài 2: Viết một script hiển thị các thông tin theo định dạng như sau:

```
\*****\  
Today is:  
<thông tin ngày giờ hiện tại>  
Hello <tên người dùng đang đăng nhập> !!!  
Your current working directory: <thư mục hiện hành>  
Your home directory: <thư mục cá nhân>  
Please press any key to finish: <chờ nhận một phím>  
Thank you very much!!  
\*****\  

```

Bài 3: Viết một script cho phép truyền vào 2 số nguyên như tham số của shell, sau đó thực hiện phép cộng, trừ, nhân, chia và chia lấy dư của 2 số và in ra kết quả theo định dạng như sau:

Tham số bạn đã truyền vào là 2 số: x và y
Tổng: $x + y = \langle x+y \rangle$
Hiệu: $x - y = \langle x-y \rangle$
Tích: $x * y = \langle x*y \rangle$
Thương: $x / y = \langle x/y \rangle$
Số dư: $x \% y = \langle x\%y \rangle$

Bài 4: Viết một script tên **taothumuc** cho phép người dùng nhập vào tên thư mục muốn tạo và thực hiện việc tạo thư mục. Trường hợp tạo không thành công thì in ra câu thông báo: “Khong the tao duoc thu muc!!!”.

(ghi chú: sử dụng thêm lệnh if trong bài thực hành buổi 3)

Bài 5: Viết một script với tên **thuchien** nhận vào 2 tham số **cmd1** và **arg1**. Trong đó cmd1 là lệnh cần thực hiện và arg1 là tham số của lệnh. Khi script được thực hiện thì lệnh cmd1 sẽ được thực hiện.

Ví dụ: thuchien mkdir aaa sẽ thực hiện tạo thư mục aaa.