HỆ ĐIỀU HÀNH MÃ NGUỒN MỞ

TUẦN 9

LẬP TRÌNH SHELL CĂN BẢN

Hướng dẫn làm bài:

I. Shell script là gì

Shell là chương trình giao tiếp với người dùng. Có nghĩa là shell chấp nhận các lệnh từ bạn (keyboard) và thực thi nó. Nhưng nếu bạn muốn sử dụng nhiều lệnh chỉ bằng một lệnh, thì bạn có thể lưu chuỗi lệnh vào text file và bảo shell thực thi text file này thay vì nhập vào các lệnh. Điều này gọi là shell script.

Định nghĩa: **Shell script** là một chuỗi các lệnh được viết trong plain text file. Shell script thì giống như batch file trong MS-DOS nhưng mạnh hơn.

Tại sao phải viết shell script:

• Shell script có thể nhận input từ user, file hoặc output từ màn hình.

• Tiện lợi để tạo nhóm lệnh riêng.

• Tiết kiệm thời gian.

• Tự động làm một vài công việc thường xuyên.

II. Hướng dẫn tạo và thực thi chương trình shell

step1: Tạo file hello.sh (trong thư mục cd /home/tuanvh/) nội dung như sau:

sử dụng vi, emacs, gedit... để soạn thảo nội dung

```
#!/bin/bash

echo "hello world"
```

Dòng đầu tiên chúng ta luôn đặt #!/bin/bash, đây là cú pháp bắt buộc. Sau # được hiểu là comment, chú thích của các đoạn mã.

step2: Sau đó, để script có thể thực thi ta phải cấp quyền cho nó

```
chmod 0777 hello.sh
```

step3: Thực thi file shell.

```
// có thể chạy file bằng 1 số cách sau
– bash hello.sh
– sh hello.sh
– ./hello.sh
```

III. Biến trong shell

Trong Linux shell có 2 loại biến:

Biến hệ thống :

- Tạo ra và quản lý bởi Linux.
- Tên biến là CHỮ HOA

Biến do người dùng định nghĩa

-Tạo ra và quản lý bởi người dùng -Tên biến là chữ thường

1. Một số biến hệ thống

VD: file hello.sh

```
#!/bin/bash
echo "hello"
echo $BASH_VERSION
echo $BASH
echo $HOME
echo $PATH
```

Kết quả: run ./hello.sh

```
hello
4.2.24(1)-release
/bin/bash
/home/vu.huy.tuan
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
```

2. Biến người dùng, cú pháp, quy tắc đặt tên

cú pháp:

```
tên_biến=value
```

- **tên_biến** phải bắt đầu bằng ký tự
- Không có dấu cách 2 bên toán tử = khi gán giá trị cho biến

```
#Đúng
a=1
#sai
a = 1
#sai
a= 1
```

- Tên biến có phân biệt chữ hoa, thường

```
#các biến sau đây là khác nhau
a=1
A=2
```

- Một biến không có giá trị khởi tạo thì bằng NULL
- Không được dùng dấu ?, * để đặt tên các biến

ECHO Để in giá trị của biến

Cú pháp:

```
echo [option][string,variables…]
#example
echo $tên_biến
```

- In một số ký tự đặc biệt trong tham số với tùy chọn -e:

```
\a          alert       (bell)
\b          backspace
\c          suppress trailing new           line
\n          new         line
\r          carriage return
\t          horizontal tab
\\          backslash


//example
$ echo -e "Hello\tTuan"
#output:  Hello   Tuan
$ echo -e "Hello\nTuan"
#output
Hello
Tuan
```

IV: Các phép toán số học

Shell cung cấp cho ta một số biểu thức toán học.

Cú pháp:

```
expr toán_hạng_1 toán_tử toán_hạng_2
```

example:

```
# phép cộng
$expr 1 + 2
# phép trừ
$expr 5 - 1
# phép chia
$expr 8 / 3   # output =2 phép chia chỉ lấy phần nguyên
$expr 8 % 5  # output =3 phép chia lấy phần dư
$expr 10 \* 2 # output = 20 phép nhân
```

Chú ý: Phải có dấu cách trước và sau toán tử.

```
# example sai cú pháp
$expr 1+2
$expr 5- 1
```

Các dấu ngoặc

- Tất cả các ký tự trong dấu ngoặc kép đều không có ý nghĩa ñnh toán, trừ những ký tự sau \ hoặc $
- Dấu nháy ngược (`): nghĩa là yêu cầu thực thi lệnh

```
#example

$ echo "ngay hom nay la: `date`"

#ouput: ngay hom nay la: Wed Apr 27 10:43:59 ICT 2016

$ echo `expr 1 + 2`

#output = 3

$echo "expr 1 + 2"

#ouput: expr 1 + 2
```

Kiểm tra trạng thái trả về của 1 câu lệnh **cú pháp

```
$echo $?
```

– Trạng thái 0 nếu câu lệnh kết thúc thành công. – Khác 0 nếu kết thúc có lỗi

```
# xóa file không tồn tại

rm abc.txt #output messge:( rm: cannot remove `abc.txt': No such file or directory )

# kiểm tra trạng thái câu lệnh rm abc.txt

$echo $? #output 1 nghĩa là có lỗi


$ echo "ngay hom nay la: `date`"

#ouput: ngay hom nay la: Wed Apr 27 10:43:59 ICT 2016

$echo $? #output 0, nghĩa là thành công
```

V: Cấu trúc điều khiển trong shell script

Cũng giống như các ngôn ngữ lập trình khác, Shell Scripts cũng cung cấp các vòng lặp: "for", "while"; và lệnh rẽ nhánh "if", "case".

1. Cú pháp rẽ nhánh If Cú pháp:

```
if điều_kiện
then
câu lệnh 1
…
fi
```

if...else...fi

Cú pháp:

```
if điều_kiện then
        câu_lệnh_1
….
else
        câu_lệnh_2
fi
```

Vòng lặp For

Cú pháp:

```
for { tên biến } in { danh sách }
do
# Khối lệnh
# Thực hiện từng mục trong danh sách cho đến cho đến hết
# (Và lặp lại tất cả các lệnh nằm trong "do" và "done")
done
```

```
#hoặc sử dụng for
for (( expr1; expr2; expr3 ))
do
# Lặp cho đến khi biểu thức expr2 trả về giá trị TRUE
done
```

example

```
#  for 1
for i in 1 2 3 4 5
do
 echo $i
done


#output: 1 2 3 4 5


#for 2
for (( i = 0 ; i <= 5; i++ )) # bao quanh bằng (())
do
echo $i
done
#ouput 1 2 3 4 5
```

3. Vòng lặp While

```
while      [Điều kiện]
do
```

```
command1
command2
command3 .. ....
done
```

example demo1.sh

```
#!/bin/sh
echo "Nhap vao cac so can tinh tong, nhap so am de exit"
sum=0
read i
while [ $i -ge 0 ] # nếu i >= 0
do
sum=`expr $sum + $i`
read i # nhận giá trị từ người dùng
done
echo "Total: $sum."
```

Kết quả sau khi chạy ./demo1.sh

```
#ouput
./demo1.sh
Nhap vao cac so can tinh tong, nhap so am de exit
1
5
4
```

```
-1

Total= 10.
```

VI: Lệnh test

Lệnh test được dùng để kiểm tra một biểu thức là đúng hay không và trả lại

– 0 nếu biểu thức đúng

– khác 0 sai

Cú pháp:

```
test biểu_thức HOẶC  [biểu thức]
```

Các phép toán kiểm tra

Mathematical Operator in,Shell Scrip	Meaning	Normal Arithmetical/ Mathematical Statements		
			For test statement with if command	For [expr] statement with if command
-eq	is equal to	5 == 6	if test 5 -eq 6	if [5 -eq 6]
-ne	is not equal to	5 != 6	if test 5 -ne 6	if [5 -ne 6]
-lt	is less than	5 < 6	if test 5 -lt 6	if [5 -lt 6]
-le	is less than or equal to	5 <= 6	if test 5 -le 6	if [5 -le 6]
-gt	is greater than	5 > 6	if test 5 -gt 6	if [5 -gt 6]

Mathematical Operator in,Shell Scrip	Meaning	Normal Arithmetical/ Mathematical Statements		
-ge	is greater than or equal to	5 >= 6	if test 5 -ge 6	if [5 -ge 6]

NOTE: == is equal, != is not equal.

For string Comparisons use

Operator	Meaning
string1 = string2	string1 is equal to string2
string1 != string2	string1 is NOT equal to string2
string1	string1 is NOT NULL or not defined
-n string1	string1 is NOT NULL and does exist
-z string1	string1 is NULL and does exist

Toán tử logic

Operator	Meaning
! expression	Logical NOT
expression1,-a,expression2	Logical AND
expression1,-o,expression2	Logical OR

kiểm tra file, thư mục

Test	Meaning
-s file	Non empty file
-f file	Is File exist or normal file and not a directory
-d dir	Is Directory exist and not a file
-w file	Is writeable file
-r file	Is read-only file
-x file	Is file is executable