Thực hành HĐH MNM

Trịnh Tấn Đạt

Tuần 8

https://sites.google.com/site/ttdat88

Nội dung

- Lập trình shell cơ bản

Lập trình shell

- **Shell script** là một chuỗi các lệnh được viết trong plain text file.
- 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

Lập trình shell

Tạo và thực thi chương trình shell

- **Step 1**: tạo một file *.sh (ví dụ: temp.sh) sử dụng vi, gedit, …

```
#!/bin/bash <- shell mà script sẽ chạy
command … <- lệnh
command…
exit 0 <- thoát
```

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.

Lập trình shell

- **Step2**: Sau đó, để script có thể thực thi ta phải cấp quyền cho nó

chmod 0777 temp.sh

- **Step3**: Thực thi file shell.

// có thể chạy file bằng 1 số cách sau

// Open terminal , cd <đường dẫn chứa file>

bash hello.sh

sh hello.sh

./hello.sh

Lập trình shell

- Ví dụ : tạo file test.sh

```
#!/bin/bash
echo “Hello World !”
name=”name_user”
age=22
echo $name
echo $age
```

chmod 0777 test.sh

./test.sh

Các bạn lưu ý dấu = phải viết liền không được có dấu cách ví dụ age = 22, sẽ báo lỗi cú pháp.

Biến phân biệt chữ hoa và chữ thường, ví dụ biến NAME sẽ khác biến name

Lập trình shell

- Biến: có 2 loại
  - **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
- Định nghĩa biến: Cú pháp: tên biến=giá trị.
- Một số quy định về biến trong shell:
  - (1) Tên bắt đầu bằng ký tự hoặc dấu gạch chân (_).
  - (2) Không được có khoảng trắng trước và sau dấu bằng khi gán giá trị cho biến
  - (3) Biến có phân biệt chữ hoa chữ thường.
  - (4) Bạn có thể khai báo một biến có giá trị NULL như sau: var01= hoặc var01="" (5) Không dùng ?, * để đặt tên biến.

Lập trình shell

- Để truy xuất giá trị biến, dùng cú pháp sau: $tên_biến
- Ví dụ:

n=10

echo $n

- Lệnh echo: Dùng để hiển thị dòng văn bản, giá trị biến … Cú pháp: echo [options] [chuỗi, biến…] Các option:

```
-n: không in ký tự xuống dòng.
-e: cho phép hiểu những ký tự theo sau dấu \ trong chuỗi
\a: alert (tiếng chuông)
\b: backspace
\c: không xuống dòng
\n: xuống dòng
\r: về đầu dòng
\t: tab
\\: dấu \
```

ví dụ: echo –e “1 2 hello \a\t\t world \n"

Lập trình shell

- VD: file hello.sh

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

Lập trình shell

- Truyền tham số vào file

Ví dụ : tạo file test.sh

```
#!/bin/bash

name=$1
age=$2
echo “Name : ” $name
echo “Age : ” $age
```

./test.sh User_name 22

Lập trình shell

- Ví dụ tính bình phương một số bp.sh

```
#!/bin/bash
number=$(($1 * $1))
echo “Binh phuong cua $1 la : $number”

./bp.sh 5
```

Lập trình shell

- Tính toán: expr , let hoặc $((...))

```
expr 1 \+ 3
expr 2 \- 1
expr 10 \/ 2
expr 20 \% 3
expr 10 \* 3
echo `expr 6 \+ 3`
z=`epxr $z \+ 3`
```

```
let "z=$z+3"
let "z += 3"
let "z=$m*$n"
```

```
z=$((z+3))
z=$(($m*$n))
```

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

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

in kết quả ra màn hình: echo $z

Lập trình shell

- Trạng thái exit: khi một lệnh hoặc script thực thi , nó trả về 2 loạI giá trị để xác định xem lệnh hoặc script đó có thực thi thành công không.
  - (1). Nếu giá trị trả về là 0 (zero) -> lệnh thực thi thành công
  - (2). Nếu giá trị trả về khác 0 (nonzero) -> không thành công
- Để biết được giá trị trả về của một lệnh hay 1 script ? Rất đơn giản, chỉ cần sử dụng biến đặc biệt có sẵn của shell: $?
- Ví dụ : xoá 1 file không tồn tạI trên đĩa cứng

rm unknowfile

echo $?

sẽ in ra màn hình một giá trị khác 0

Lập trình shell

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
- Ví dụ

$ echo “ngay hom nay la: date"

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

$ echo `expr 1 + 2`

$echo “expr 1 + 2”

$echo “expr 1 \+ 2”

Lập trình shell

- Cấu trúc rẽ nhánh:

```
if [ expression 1 ]
then
   Statement(s) to be executed if expression 1 is true
elif [ expression 2 ]
then
   Statement(s) to be executed if expression 2 is true
elif [ expression 3 ]
then
   Statement(s) to be executed if expression 3 is true
else
   Statement(s) to be executed if no expression is true
fi
```

Lập trình shell

- Ví dụ: tao file test.sh

```
if [ -z $1 ]; then
    echo "Chua nhap tham so"
else
    number=$(($1 * $1))
    echo "Binh phuong cua $1 la : $number"
fi
```

-z là nếu không tồn tại tham số 1

./test.sh

./test.sh 10

Lập trình shell

- Ví dụ:

```
a=10
b=20
if  [ $a == $b ]
then
        echo "a is equal to b"
elif [ $a -gt $b ]
then
        echo "a is greater than b"
elif [ $a -lt $b ]
then
        echo "a is less than b"
else
        echo "None of the condition met"
fi
```

Lập trình shell

Lệnh so sánh với số

Cú pháp	Ý nghĩa
n1 -eq n2	Kiểm tra n1 = n2
n1 -ne n2	Kiểm tra n1 khác n2
n1 -lt n2	Kiểm tra n1 < n2
n1 -le n2	Kiểm tra n1 <= n2
n1 -gt n2	Kiểm tra n1 > n2
n1 -ge n2	Kiểm tra n1 >= n2

Lệnh so sánh với chuỗi

Cú pháp	Ý nghĩa
s1 = s2	Kiểm tra s1 = s2
s1 != s2	Kiểm tra s1 khác s2
-z s1	Kiểm tra s1 có kích thước bằng 0
-n s1	Kiểm tra s1 có kích thước khác 0
s1	Kiểm tra s1 khác rỗng

Lập trình shell

Toán tử kết hợp

Column 1	Column 2
!	Phủ định (not)
-a	Và (and)
-o	Hoặc (or)

Lệnh kiểm tra file (nói chung cho cả tệp và thư mục)

Cú pháp	Ý nghĩa
-f file	Kiểm tra xem file có phải là tệp hay không
-d file	Kiểm tra xem file có phải là thư mục hay không
-r file	Kiểm tra file có đọc (read) được hay không
-w file	Kiểm tra file có ghi (write) được hay không
-x file	Kiểm tra file có thực thi (execute) được hay không
-s file	Kiểm tra file có kích thước lớn hơn 0 hay không
-e file	Kiểm tra xem file có tồn tại hay không

Lập trình shell

- Vòng lặp for:

```
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
```

Lập trình shell

- Ví dụ:

```
#  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
```

Lập trình shell

- Ví dụ:

```
for (( i = 1; i <= 9; i++ ))
do
echo "Bảng $i:"
echo "-------"
for (( j = 1 ; j <= 10; j++ ))
do
echo "$i * $j = `expr $i \* $j`"
done
done
```

Lập trình shell

- Vòng lặp while:

```
while   [Điều kiện]
do
command1
command2
command3     ..   ....
done
```

Lập trình shell

- Ví dụ:

```
echo "Nhap vao cac so can tinh tong, nhap so am de exit"
sum=0
read i
while [ $i -ge 0 ]
do sum=`expr $sum + $i`
read i
echo "Total: $sum."
```

Lập trình shell

Chương trình tính tổng 1-> n

- Minh họa các cấu trúc while do done, và cách sử dụng [], $(()).
- Tập tin **tong1.sh**

```
#!/bin/sh
echo “Chuong trinh tinh tong 1- $1”
index=0
tong=0
while [ $index -lt $1 ]
do
    index=$(($index + 1))
    tong=$(($tong + $index))
done
echo "Tong 1-$1= $tong"
exit 0
```

- Chạy chương trình :

```
chmod a+x tong1.sh
./tong1.sh 100
```

Lập trình shell

Bài tập:

- Tính giai thừa một số nguyên N
- Kiểm tra một số có phải số nguyên tố
- Giải phương trình bậc 1/ bậc 2
- Đổi cơ sỗ từ hệ thập phân sang hệ nhị phân

Next

- Đọc/ghi từ file, redirection
- Array, String