

Thực hành HĐH MNM

Trịnh Tấn Đạt

Tuần 9

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

Nội dung

- Lập trình shell cơ bản (2)

Lập trình shell

- 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]`

			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]

Lập trình shell

Tham số dòng lệnh

- Giả sử ta có script tên test.sh, để thực thi script này ta cần truyền vào 2 tham số như sau :

```
./test.sh one two
```

Trong đó test.sh là tên script

one : tham số thứ nhất truyền vào script

two : tham số thứ hai

Trong shell, bạn truy xuất đến những tham số như sau :

test.sh là \$0

one là \$1

two là \$2

Và biến `##` (có sẵn trong shell) sẽ cho giá trị 2 (có 2 tham số one và two). Bạn có thể lấy tất cả các tham số bằng cách sử dụng biến `$@` hoặc `$*`

Vì dụ:

```
args=("$@") # 3 tham số đưa vào mảng  
echo ${args[0]} ${args[1]} ${args[2]}
```

File: test.sh

Run :

./test.sh 100 Hello World

#echo arguments to the shell

```
echo $1 $2 $3 '-> echo $1 $2 $3'  
args=("$@")
```

#echo arguments to the shell

```
echo ${args[0]} ${args[1]} ${args[2]} '-> args=("$@"); echo ${args[0]} ${args[1]} ${args[2]}
```

#use \$@ to print out all arguments at once

```
echo $@ "-> echo $@"  
echo $@ '-> echo $@'
```

use \$\$ variable to print out

number of arguments passed to the bash script

```
echo "Number of arguments passed:" $$ '-> echo Number of arguments passed: $#'  
echo "Number of arguments passed:" $$ "-> echo Number of arguments passed: $#"
```

Lập trình shell

- **Function**: shell cũng cho phép khai báo function để thực hiện một nhóm các lệnh và trả về cho chúng ta kết quả có thể sử dụng tùy từng mục đích.

```
function_name () {  
    list of commands  
}
```

- Tạo file test.sh và thực thi ./test.sh

```
said () {  
    echo "Hello"  
}  
said
```

Lập trình shell

- Truyền tham số cho function
- file test.sh

```
said () {  
  echo "Hello $1 $2"  
}  
said World NewWorld
```

Lập trình shell

Return kết quả từ một hàm

- File tinh tong.sh

```
sum () {  
    a=$(( $1 + $2 ))  
    return $a  
}  
sum 10 20  
ret=$?  
echo "value is $ret"
```

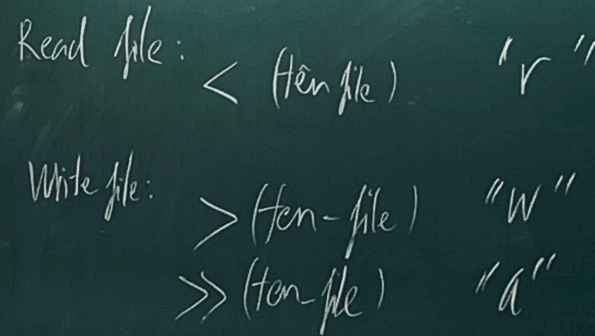

Lập trình shell

- Gọi một function trong một function

```
number_one () {  
    echo "This is the first function"  
    number_two  
}  
  
number_two () {  
    echo "This is now the second function"  
}  
  
number_one
```

Lập trình shell

Redirection



Read file: $<$ (ten file) 'r'
Write file: $>$ (ten file) 'w'
 $>>$ (ten file) 'a'

- Với Linux bạn còn có thể xuất dữ liệu vào file và đọc dữ liệu từ file
- Ví dụ: `ls > filename`
in kết quả lệnh ls vào file có tên filename.
- Có 3 ký hiệu redirection là $>$, $>>$ và $<$
 - (1). Ký hiệu $>$ cú pháp: `command > filename` Xuất output của lệnh ra file. Nếu file tồn tại thì nó sẽ ghi đè còn nếu file chưa có thì tạo file mới
 - (2). Ký hiệu $>>$ cú pháp: `command >> filename` Xuất output của lệnh vào cuối file nếu file đã tồn tại, còn nếu file chưa tồn tại thì tạo file mới.
 - (3). Ký hiệu $<$ cú pháp: `command < filename` lấy dữ liệu cho linux-command từ filename thay vì từ bàn phím.

Lập trình shell

- **Biến toàn cục và cục bộ: Global vs. Local variables**

ví dụ

```
#!/bin/bash
#Define bash global variable
VAR="global variable"
function bash {
#Define bash local variable
local VAR="local variable"
echo $VAR
}
echo $VAR
bash
# Note the bash global variable did not change
# "local" is bash reserved word
echo $VAR
```

Lập trình shell

- Lệnh `read` : đọc giá trị nhập từ bàn phím , file ...
- Dùng để lấy dữ liệu nhập từ bàn phím và lưu vào biến
- Cú pháp :

```
read var1 var2 var3 ... varN
```

- `read` không có tham số giá trị sẽ được chứa trong biến `$REPLY`

ví dụ :

```
read  
var="$REPLY"
```

Lập trình shell

- Bình thường thì dấu `\` cho phép xuống dòng để nhập tiếp dữ liệu trong `read`. Nếu `read -r` thì sẽ không có ý nghĩa đó.
- Ví dụ:

```
read var # nhập vào: first line \  
second line
```

```
echo "$var" kết quả: first line second line
```

- Nhưng với tham số `r` thì sao?

```
read -r var # nhập vào: first line \  
second line
```

```
echo "$var" kết quả: first line \  
second line
```

Lập trình shell

```
echo -e "Hi, please type the word: \c"
read word
echo "The word you entered is: $word"
echo -e "Can you please enter two words?"
read word1 word2
echo " Here is your input: \ " $word1\ "\ " $word2\ " "
echo -e "How do you feel about bash scripting?"
# read command now stores a reply into the default build-in variable $REPLY
read
echo "You said $REPLY, I'm glad to hear that! "
echo -e "What are your favorite colours ? "
# -a makes read command to read into an array
read -a colours
echo "My favorite colours are also ${colours[0] }, ${colours[1] } and ${colours[2] }:-)"
```

Lập trình shell

- Lệnh `read` có thể dùng để đọc file.
- Nếu file chứa nhiều hơn 1 dòng thì chỉ có dòng thứ nhất được gán cho biến. Nếu `read` với nhiều hơn 1 biến (`read var1 var2 ...`) thì `read` sẽ dựa vào biến `$IFS` để gán dữ liệu cho các biến.
 - Mặc định thì `IFS` là khoảng trắng.

File: test3.sh

Run: `./test3.sh test.sh`

```
while read line
do
  echo $line
done < $1
```

Lập trình shell

- Sử dụng \$IFS (Internal File Separator) để tách một dòng input của read, nếu bạn không muốn mặc định là khoảng trắng
- Ví dụ:

```
echo "liet ke tat ca user "  
OIFS=$IFS; IFS=:  
while read name passwd uid gid fullname ignore  
do  
    echo "$name $fullname"  
done < /etc/passwd  
IFS=$OIFS
```

Hoặc

```
while IFS=: read name passwd uid gid fullname ignore  
do  
    echo "$name $fullname"  
done < /etc/passwd
```


Lập trình shell

- Ví dụ: linecounting.sh ; run : ./linecounting.sh test.txt

```
echo "Chương trình đếm số dòng của tập tin $1"  
{  
n=0  
while read line  
do  
n=$((n + 1))  
done  
echo "Số dòng của tập tin $1 là : $n"  
exit 0
```

Lập trình shell

- Ví dụ: wordcounting.sh ; run : ./ wordcounting.sh test.txt

```
echo "Chương trình đếm số từ của tập tin $1"
{
n=0
while read line
do
    for wd in $line
    do
        n=$((n + 1))
    done
done
echo "Tổng số từ của tập tin $1 là : $n"
}<$1
exit 0
```

Lập trình shell

- Ví dụ: tìm dòng có độ dài lớn nhất trong một tập tin

```
echo "Chương trình tìm dòng dài nhất trong tập tin $1"
{
n=0
max=0
dong=""
while read line
do
    n=`expr length "$line"`
    if [ $n -gt $max ]
    then
        dong="$line"
        max=$n
    fi
done
echo "Dòng trong tập tin $1 có độ dài max = $max là :
$dong"
}<$1
exit 0
```

Lập trình shell

- Array:
- Ví dụ

```
array_var=(1 2 3 4 5 6)
echo ${array_var[0]}
index=4
echo ${array_var[$index]}
```

Hoặc

```
NAME[0]="Zara"
NAME[1]="Qadir"
NAME[2]="Mahnaz"
NAME[3]="Ayan"
NAME[4]="Daisy"
echo "First Method:${NAME[*]}"
echo "Second Method:${NAME[@]}"
```

Lập trình shell

- In chiều dài (số phần tử) của mảng

```
echo ${#array_var[*]}
```

- Định nghĩa mảng kết hợp

```
declare -A fruits_value  
fruits_value=([apple]='100 dollars' [orange]='150 dollars')  
echo "Apple costs ${fruits_value[apple]}"
```

- Liệt kê các chỉ mục của mảng

```
echo ${!array_var[*]} hoặc  
echo ${!array_var[@]}
```

Vi dụ :

```
echo ${!fruits_value[*]}
```

Lập trình shell

- Ví dụ: nhập mảng

```
declare -a a
echo -n "Nhập n: "
read n
for ((i=1;i<=n;i++));
do
    echo -n "a[$i]= "
    read m
    a[$i]=$m
done
echo -n "Mảng chưa sắp xếp: " echo ${a[*]}
```

Bài tập

1. Viết function Kiểm tra số nguyên tố
(dùng \$? để lấy giá trị trả về)
2. Nhập vào một mảng. Sắp xếp mảng tăng dần.

1. Viết function Kiểm tra số nguyên tố (dùng \$? để lấy giá trị trả về)

```
1  #!/bin/bash
2
3  function prime() {
4      local n=$1
5      if ((n<2)); then
6          return 0
7      else
8          for ((i=2; i<=((n/2)); i++)); do
9              if ((n%i==0)); then
10                 return 0
11             fi
12          done
13          return 1
14      fi
15  }
16
17  echo -n "N: "
18  read n
19  prime $n
20  if [ $? -eq 1 ]; then
21      echo "True"
22  else
23      echo "False"
24  fi
```


2. Nhập vào một mảng. Sắp xếp mảng tăng dần.

```
1  #!/bin/bash
2
3  function InterchangeSort() {
4      local n=${#a[@]}
5      for (( i=0; i<n-1; i++ )); do
6          for (( j=i+1; j<n; j++ )); do
7              if (( a[i] > a[j] )); then
8                  temp=${a[i]}
9                  a[i]=${a[j]}
10                 a[j]=$temp
11             fi
12         done
13     done
14 }
15
16 declare -a a
17 echo -n "N: "
18 read n
19
20 for (( i=0; i<n; i++ )); do
21     echo -n "a[$i] = "
22     read a[i]
23 done
24
25 echo "Before: ${a[*]}"
26 InterchangeSort a
27 echo "After: ${a[*]}"
```

Next

- Lập trình C trên ubuntu (linux)