

# Thực hành HỆ THỐNG 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.

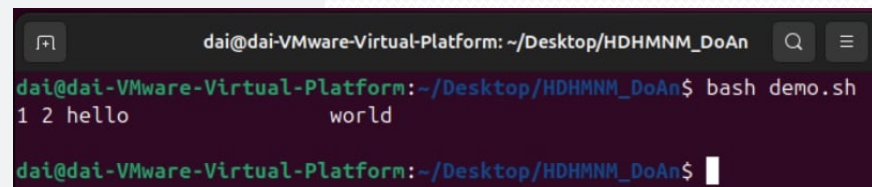
# Lập trình shell

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

```
n=10  
echo $n
```

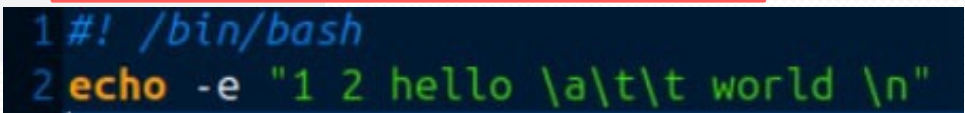
- **Lệnh `echo`:** Dùng để hiển thị dòng văn bản, giá trị biến ... Cú pháp: `echo [options] [chuỗi, biến...]` Các option:

```
-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 \
```



```
dai@dai-VMware-Virtual-Platform: ~/Desktop/HDHMMN_DoAn  
dai@dai-VMware-Virtual-Platform: ~/Desktop/HDHMMN_DoAn$ bash demo.sh  
1 2 hello  
world  
dai@dai-VMware-Virtual-Platform: ~/Desktop/HDHMMN_DoAn$
```

ví dụ: `echo -e "1 2 hello \a\t\t world \n"`



```
1 #! /bin/bash  
2 echo -e "1 2 hello \a\t\t world \n"
```



```
dai@dai-VMware-Virtual-Platform: ~/Desktop/HDHMMN_DoAn
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMMN_DoAn$ bash demo.sh
hello
5.2.21(1)-release
/usr/bin/bash
/home/dai
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMMN_DoAn$
```

# Lập trình shell

- VD: file hello.sh

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

```
1 #! /bin/bash
2 echo "hello"
3 echo $BASH_VERSION
4 echo $BASH
5 echo $HOME
6 echo $PATH
```

```
dai@dai-VMware-Virtual-Platform: ~/Desktop/HDHMMN_DoAn
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMMN_DoAn$ bash demo.sh User_name 22
Name : User_name
Age : 22
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMMN_DoAn$
```

# 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
```

```
1 #! /bin/bash
```

```
2
```

```
3 name=$1
```

```
4 age=$2
```

```
5 echo "Name : " $name
```

```
6 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 "Bình phương của $1 là : $number"
```

```
./bp.sh 5
```

```
1 #! bin/bash
2
3 echo `expr 6 \+ 3`
```

```
dai@dai-VMware-Virtual-Platform: ~/Desktop/HDHMNM_DoAn
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMNM_DoAn$ bash demo.sh
9
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMNM_DoAn$
```

# 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: xóa 1 file không tồn tại trên đĩa cứng

```
rm unknowfile  
echo $?
```

```
1 #! bin/bash  
2  
3 rm unknowfile  
4 echo $?
```

```
dai@dai-VMware-Virtual-Platform: ~/Desktop/HDHMMN_DoAn  
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMMN_DoAn$ bash demo.sh  
rm: cannot remove 'unknowfile': No such file or directory  
1  
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMMN_DoAn$
```

### 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 ãnh 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”
```

```
1 #! bin/bash
2
3 echo `date`
```

```
dai@dai-VMware-Virtual-Platform: ~/Desktop/HDHMMN_DoAn
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMMN_DoAn$ bash demo.sh
Sat Apr 5 11:46:27 PM +07 2025
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMMN_DoAn$
```

# 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: tạo 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

Nếu chiều dài tham số là 0

./test.sh

./test.sh 10



```
dai@dai-VMware-Virtual-Platform: ~/Desktop/HDHMNM_DoAn
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMNM_DoAn$ bash demo.sh
a is less than b
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMNM_DoAn$
```

# 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
```

```
1 #! /bin/bash
2
3 a=10
4 b=20
5 if [ $a == $b ]
6 then
7     echo "a is equal to b"
8 elif [ $a -gt $b ]
9 then
10    echo "a is greater than b"
11 elif [ $a -lt $b ]
12 then
13    echo "a is less than b"
14 else
15    echo "None of the condition met"
16 fi
```

# Lập trình shell

## Lệnh so sánh với số

Cú pháp	Ý nghĩa
<code>n1 -eq n2</code>	Kiểm tra $n1 = n2$
<code>n1 -ne n2</code>	Kiểm tra $n1$ khác $n2$
<code>n1 -lt n2</code>	Kiểm tra $n1 < n2$
<code>n1 -le n2</code>	Kiểm tra $n1 \leq n2$
<code>n1 -gt n2</code>	Kiểm tra $n1 > n2$
<code>n1 -ge n2</code>	Kiểm tra $n1 \geq n2$

## Lệnh so sánh với chuỗi

Cú pháp	Ý nghĩa
<code>s1 = s2</code>	Kiểm tra $s1 = s2$
<code>s1 != s2</code>	Kiểm tra $s1$ khác $s2$
<code>-z s1</code>	Kiểm tra $s1$ có kích thước bằng 0
<code>-n s1</code>	Kiểm tra $s1$ có kích thước khác 0
<code>s1</code>	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  
# Khởi 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
```

```
dai@dai-VMware-Virtual-Platform: ~/Desktop/HDHMNM_DoAn
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMNM_DoAn$ bash demo.sh
0
1
2
3
4
5
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMNM_DoAn$
```

```
1 #! bin/bash
2
3 for (( i=0; i<=5; i++))
4 do
5     echo $i
6 done
```

# 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."
```

done



# 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 "Chương trình tính 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 số từ hệ thập phân sang hệ nhị phân

## Tính giai thừa một số nguyên N

```
1  #!/bin/bash
2
3  echo -n "N: "
4  read n
5  if (( n==0 || n==1 )); then
6      echo "$n! = 1"
7  else
8      s=$n
9      for ((i=2; i<n; i++)); do
10         let s=$s*i
11     done
12     echo "$n! = $s"
13 fi
```

~~echo "Thank you very much!!"~~  
~~#nhập n~~

1. Tính giai thừa  
factorial = 1  
for ((i=1, i<=n, i++))  
do  
factorial = \$(( factorial \* i ))  
done  
echo "Giai thừa của \$n là: \$factorial"

## Kiểm tra một số có phải số nguyên tố

```
1  #! bin/bash
2
3  echo -n "N: "
4  read n
5  if [ $n -lt 2 ]; then
6      echo "no"
7  else
8      for ((i=2; i<=((n/2)); i++)); do
9          if ((n%i==0)); then
10             echo "no"
11             exit 1
12          fi
13      done
14      echo "yes"
15  fi
```

### 2 Kiểm tra snt

#!/bin/bash

echo -n "N: "

read n

if [ \$n -lt 2 ]; then

echo "No"

else

for ((i=2; i<=((n/2)); i++)); do

if ((n%i==0)); then

echo "No"

exit 1

fi

done

echo "Yes"

fi

# Giải phương trình bậc 1/ bậc 2

```
1  #!/bin/bash
2
3  echo "1)PT bac 1"
4  echo "2)PT bac 2"
5  echo -n "Choose: "
6  read choose
7  case $choose in
8      1)
9      echo -n "a: "
10     read a
11     echo -n "b: "
12     read b
13     echo "y = $a*x + $b"
14     echo "$a*x + $b = 0"
15     if [ $a -eq 0 ]; then
16         if [ $b -ne 0 ]; then
17             echo "VN"
18         else
19             echo "VSN"
20         fi
21     else
22         echo -n "x = "
23         echo "scale=2;$b*(-1)/$a" | bc
24     fi
25 ;;
26 2)
27     echo -n "a: "
28     read a
29     echo -n "b: "
30     read b
31     echo -n "c: "
32     read c
33     echo "y = $a*x^2 + $b*x + $c"
34     echo "$a*x^2 + $b*x + $c = 0"
35     if [ $a -eq 0 ]; then
36         if [ $b -eq 0 ]; then
37             if [ $c -ne 0 ]; then
38                 echo "VN"
39             else
40                 echo "VSN"
41             fi
42         else
43             echo -n "x = "
44             echo "scale=2;$c*(-1)/$b" | bc
45         fi
46     else
47         delta=$((b*b-4*a*c))
48         if [ $delta -lt 0 ]; then
49             echo "VN"
50         elif [ $delta -eq 0 ]; then
51             echo -n "x1 = x2 = "
52             echo "scale=2;$b*(-1)/2*$a" | bc
53         else
54             echo -n "x1 = "
55             echo "scale=2;($b*(-1)+sqrt($delta))/2*$a" | bc
56             echo -n "x2 = "
57             echo "scale=2;($b*(-1)-sqrt($delta))/2*$a" | bc
58         fi
59     fi
60 ;;
61 *)
62     echo "Invalid input"
63 ;;
64 esac
```

## Đổi cơ số từ hệ thập phân sang hệ nhị phân

```
1  #!/bin/bash
2
3  echo -n "N: "
4  read n
5  echo "obase=2;$n" | bc
```

4. Bài cơ số từ thập phân sang nhị phân

```
#!/bin/bash
read -p "Nhập vào số thập phân dec" dec
binary=$(echo "obase=2;$dec" | bc)
echo "$dec trong hệ nhị phân là $binary"
```

# Next

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