



# LẬP TRÌNH SHELL CƠ BẢN

Trịnh Tấn Đạt

Khoa CNTT - Đại Học Sài Gòn

Email: [trinhtandat@sgu.edu.vn](mailto:trinhtandat@sgu.edu.vn)

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



# NỘI DUNG

Khái niệm Shell script

Các loại Shell trong Linux

Thông dịch

Tham biến trong Shell

Lệnh kiểm tra điều kiện

Cấu trúc điều khiển trong Shell

Phép toán số học trong Shell

# I. KHÁI NIỆM SHELL SCRIPT

- Máy tính chỉ có thể thực hiện các lệnh dạng nhị phân (bit 0, 1), còn gọi là mã nhị phân. Các máy tính muốn thực hiện được chương trình thì người dùng phải nạp chương trình dưới dạng các bit 0 và 1.
- Đây là một điều rất phiền toái và cực kỳ khó khăn với con người. Để khắc phục nhược điểm này các nhà thiết kế và xây dựng hệ điều hành đều có kèm theo một chương trình đặc biệt.
- Thông qua chương trình này người dùng có thể nhập các lệnh dưới dạng ngôn ngữ cấp cao (tiếng Anh) để yêu cầu hệ điều hành thực hiện một công việc nào đó. Chương trình đặc biệt này được gọi là **Shell** (Bộ thông dịch lệnh).

# I. KHÁI NIỆM SHELL SCRIPT

- Hệ điều hành MS-DOS và Windows thì có môi trường đánh lệnh command-line, hệ điều hành Unix và Linux có **môi trường Shell**
- *Shell không là một thành phần của hệ điều hành mà nó sử dụng hệ điều hành để thực thi lệnh, thao tác file...* Hệ điều hành Linux có thể có nhiều loại Shell khác nhau.
- Shell là nơi cho phép người dùng nhập lệnh (thông thường từ bàn phím) và thực thi lệnh. Nhưng thay vì người dùng nhập tuần tự các câu lệnh và thực thi chúng một cách tuần tự thì người dùng có thể lưu các lệnh này vào một file text và yêu cầu shell thực hiện file này. Điều này được gọi là **shell script**

# I. KHÁI NIỆM SHELL SCRIPT

- Những tính năng của shell
  - Xử lý tương tác (Interactive processing)
  - Chạy nền (Background)
  - Chuyển hướng (Redirection)
  - Ống dẫn (Pipe)
  - Tập tin lệnh (Shell scripts)
  - Biến shell (Shell variables)
  - Dừng lại các lệnh đã thực hiện (Command history)
  - Cấu trúc lệnh như ngôn ngữ lập trình
  - Tự động hoàn tất tên tập tin hoặc lệnh
  - Bí danh cho lệnh (Command alias)

# I. KHÁI NIỆM SHELL SCRIPT

Ví dụ:

tao file -sh vi nano  
#!/bin/bash sh gedit

```
#!/bin/bash
clear
echo "Hello $USER"
echo -n "Today is"
date "+%A %d %B %Y"
echo -n "There is/are"
who | wc -l
echo connection on $HOSTNAME
echo "Calendar" ; cal
exit 0
```

```
1#!/bin/bash
2clear
3echo "Hello $USER"
4echo -n "Today is"
5date "+%A %d %B %Y"
6echo -n "There is/are"
7who | wc -l
8echo connection on $HOSTNAME
9echo "Calendar" ; cal
10exit 0
```

```
dai@dai-VMware-Virtual-Platform: ~/Desktop/HDHMMN_DoAn
Hello dai
Today is Saturday 05 April 2025
There is/arewc: -l: No such file or directory
connection on dai-VMware-Virtual-Platform
Calendar
April 2025
Su Mo Tu We Th Fr Sa
        1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMMN_DoAn$
```

## II. CÁC LOẠI SHELL TRONG LINUX

- **Shell Bourne (sh)** do Steven Bourne viết, đó là Shell nguyên thủy có mặt trên hầu hết các hệ thống Unix/Linux...Nó rất hữu dụng cho việc lập trình Shell nhưng nó không xử lý tương tác người dùng như các Shell khác...
- **Bash shell (Bourne Again Shell)**, thường được biết những shell có đuôi .sh. Đây là phần mở rộng của sh, nó kế thừa những gì sh đã có và phát huy những gì sh chưa có...Nó có giao diện lập trình rất mạnh và linh hoạt...Cùng với giao diện lệnh dễ dung...Đây là Shell được cài đặt mặc định trên các hệ thống Linux.

## II. CÁC LOẠI SHELL TRONG LINUX

- **csh (C Shell)**, của đại học Berkeley, và tcsh là version cải tiến của csh. Đáp ứng tương thích cho người dùng...Nó hỗ trợ rất mạnh cho những Programmer C...và với đặc tính tự động hoàn thành dòng lệnh...
- **ksh (Korn Shell)** của David Korn. Có thể nói đây là một Shell tuyệt vời, nó kết hợp tính năng ưu việt của sh và csh...
- Ngoài ra còn có một số Shell khác như: ssh, nfssh, mcsh...



# III. THÔNG DỊCH

- Shell như là một thông dịch lệnh:

Login vào máy tính -> dấu nhắc shell -> yêu cầu lệnh -> shell đọc lệnh -> shell tìm kiếm tải tiện ích vào bộ nhớ ->

- **Tìm thấy** -> shell thực thi tiện ích -> trở lại dấu nhắc.
  - **Không tìm thấy** -> shell báo lỗi và hiển thị dấu nhắc.
- Môi trường làm việc gồm hai thành phần
    - Môi trường terminal.
    - Môi trường shell.

# III. THÔNG DỊCH

- Ta có thể tạo ra một file .sh để bắt đầu thực hiện shell script

- Một bash shell luôn luôn bắt đầu bằng:

`#!/bin/bash` hay `#!/bin/sh`

- Chạy một bash shell

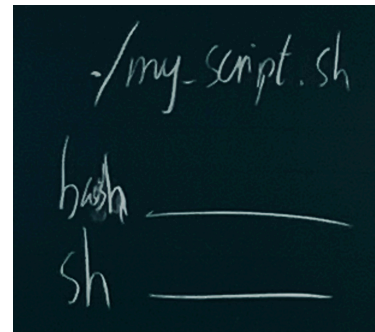
- Gán quyền execute cho file:

`chmod +x my-script`

- Chạy bằng 1 trong 2 câu lệnh sau :

`./my-script`

`bash my-script`



Trong Ubuntu

# IV. THAM BIẾN TRONG SHELL

- Trong Shell có các loại tham biến:
  - Biến thông thường
  - Biến môi trường
  - Tham số

# IV.1. BIẾN THÔNG THƯỜNG

- Không cần phải khởi tạo biến trước khi sử dụng
- Mặc định, giá trị trong biến luôn luôn là kiểu chuỗi
- Tên biến phân biệt hoa thường.
- Xuất biến ra màn hình, dùng lệnh **echo**
- Lấy giá trị của biến: **echo \$test**
- Gán giá trị cho biến:
  - **test = "welcome"**
- Cho người dùng nhập giá trị vào biến, dùng lệnh **read**
  - **read a**

# IV.1. BIẾN THÔNG THƯỜNG

- Phân biệt các VD sau:
  - `text="Monday"`
  - `echo $text` → Monday
  - `echo "Today is $text"` → Today is Monday
  - `echo 'Today is $text'` → Today is \$text
  - `echo "Today is \text"` → Today is \$text

## IV.2 BIẾN MÔI TRƯỜNG

- Các biến được khai báo sẵn và gán giá trị mặc định khi shell được khởi động
- Thường được viết hoa
- Xem danh sách biến môi trường:

**env** hay **printenv**

- Tạo biến môi trường:

**export para\_name=para\_value**

## IV.2 BIẾN MÔI TRƯỜNG

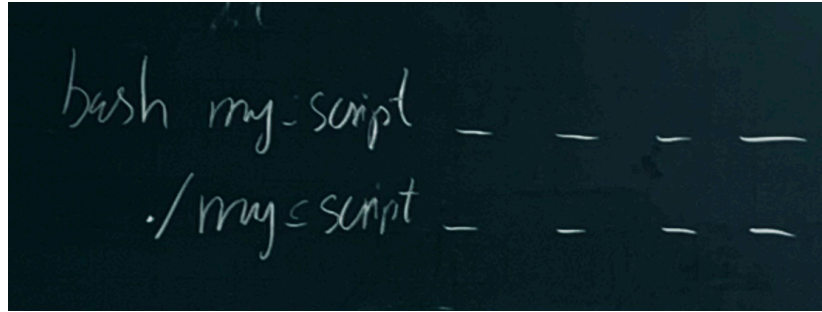
- HOME : Chứa thư mục người dùng.
- PATH : Danh sách thư mục tìm kiếm.
- PS1 : Dấu nhắc hiển thị lệnh.

## IV.3 THAM SỐ

- Những biến được xây dựng sẵn.
  - $\$ \#$  : tổng số tham số.
  - $\$ *$  : danh sách tham số đầy đủ.
  - $\$ 0$  : tên tập tin lệnh.
  - $\$ 1, \$ 2, \dots, \$ 9$  : giá trị các biến tham số thứ 1, thứ 2, ..., thứ 9



## IV.3 THAM SỐ



```
bash my-script _ _ _ _  
./my-script _ _ _ _
```

- `bash my-script Hanoi Paris Bordeaux "Ho Chi Minh City"` 4 tham số đầu vào
  - `$0="my-script"`
  - `$1="Hanoi"`, Tham số 1
  - `$2="Paris"`, Tham số 2
  - `$3="Bordeaux"`, Tham số 3
  - `$4="Ho Chi Minh City"` Tham số 4
  - `$*='Hanoi Paris Bordeaux "Ho Chi Minh City"'` Danh sách tham số đầy đủ
  - `$#=4` → số lượng tham số

# V. LỆNH KIỂM TRA ĐIỀU KIỆN

- Lệnh test:
  - Return 0 for true
  - Return 1 for false
- Có thể thay thế lệnh **test** bằng **[]**
- **VD:**
  - `test -f abc.txt`      `[ -f abc.txt ]`

# V. LỆNH KIỂM TRA ĐIỀU KIỆN

- Sử dụng:
  - `test -f name` : Kiểm tra name có phải là tập tin hay không?
  - `test -d name` : Kiểm tra name có phải là thư mục hay không?
  - `test String1=String2` : so sánh chuỗi
  - `test String1 != String2` : so sánh chuỗi
  - `test EXPR1 op EXPR2` : so sánh biểu thức với operation (op): -eq (equal) -ne (not equal) -lt (lesser than) -le (lesser or equal) -gt (greater than) -ge (greater or equal).

# V. LỆNH KIỂM TRA ĐIỀU KIỆN

## *Kiểm tra điều kiện trên tập tin*

-d file	→true nếu file là thư mục
-e file	→true nếu file tồn tại trên đĩa
-f file	→true nếu file là tập tin thông thường
-g file	→true nếu set-group-id được thiết lập trên file
-r file	→true nếu file cho phép đọc
-s file	→true nếu kích thước file khác 0
-u file	→true nếu set-ser-id được áp đặt trên file
-w file	→true nếu file cho phép ghi
-x file	→true nếu file được phép thực thi

# VI. CẤU TRÚC ĐIỀU KHIỂN TRONG SHELL

- Cấu trúc rẽ nhánh
- Cấu trúc lặp

# VI.1. CẤU TRÚC RỄ NHÁNH

## ➤ If

```
if condition
then
    statement
fi
```

```
if condition; then
    statement
fi
```

# VI.1. CẤU TRÚC RỄ NHÁNH

- So sánh hai số

**n <primitive> m**

- eq** : giá trị của n và m bằng nhau.
- ne** : giá trị của n và m không bằng nhau.
- gt** : giá trị của n lớn hơn m.
- lt** : giá trị của n nhỏ hơn m.
- ge** : giá trị của n lớn hơn hay bằng m.
- le** : giá trị n nhỏ hơn hay bằng m.

# VI.1. CẤU TRÚC RỄ NHÁNH

- So sánh hai chuỗi

**p <primitive> q**

**=** : kiểm tra rằng hai chuỗi bằng nhau. Một dấu "=" hay 2 dấu "==" đều được

**!=** : kiểm tra hai chuỗi không bằng nhau.

**<primitive> p1**

**-z** : đúng nếu chuỗi p1 có chiều dài là 0.

**-n** : đúng nếu chuỗi p1 có chiều dài khác 0.



# VI.1. CẤU TRÚC RẼ NHÁNH

- So sánh toán tử logic

! : để phủ định một mệnh đề logic.

-a : AND.

-o : OR.

# VI.1. CẤU TRÚC RỄ NHÁNH

**VD:**

```
if [ -d $r ]  
then  
    echo "The directory $r exists!"  
fi
```

# VI.1. CẤU TRÚC RỄ NHÁNH

- Cấu trúc rẽ nhánh **if**

```
if <biểu thức điều kiện>  
then  
    <lệnh 1>  
else  
    <lệnh 2>  
fi
```

Ví dụ: Nhập điểm cho môn học và xuất kết quả  
Tạo tập tin ketqua có nội dung sau:

```
echo "Chương trình kết quả môn học"  
echo "Nhập vào điểm"  
read diem  
if [ $diem -ge 5 ]  
then  
    echo "Đạt"  
else  
    echo "Rớt"  
fi
```

# VI.1. CẤU TRÚC RỄ NHÁNH

- elif

```
if condition
then
    statement
elif condition
then
    statement
else
    statement
fi
```

# VI.1. CẤU TRÚC RỄ NHÁNH

**VD:**

```
if [ -r $r ]  
then  
    echo "You can read file $r "  
elif [ -x $r ]  
then  
    echo "File $r can be executed"  
else  
    echo "Anything else"  
fi
```

# VI.1. CẤU TRÚC RỄ NHÁNH

- So sánh các đoạn chương trình sau:

```
#!/bin/sh

echo "Is it morning? Please answer yes or no"
read timeofday

if [ $timeofday = "yes" ]; then
    echo "Good morning"
else
    echo "Good afternoon"
fi

exit 0
```

# VI.1. CẤU TRÚC RỄ NHÁNH

```
#!/bin/sh

echo "Is it morning? Please answer yes or no"
read timeofday

if [ $timeofday = "yes" ]; then
    echo "Good morning"
elif [ $timeofday = "no" ]; then
    echo "Good afternoon"
else
    echo "Sorry, $timeofday not recognized. Enter yes or no"
    exit 1
fi

exit 0
```

# VI.1. CẤU TRÚC RỄ NHÁNH

```
#!/bin/sh

echo -n "Is it morning? Please answer yes or no:  "
read timeofday

if [ "$timeofday" = "yes" ]; then
    echo "Good morning"
elif [ "$timeofday" = "no" ]; then
    echo "Good afternoon"
else
    echo "Sorry, $timeofday not recognized. Enter yes or no"
    exit 1
fi

exit 0
```



# VI.1. CẤU TRÚC Rẽ NHÁNH

- Cấu trúc lựa chọn **case**

```
case <biến> in  
  <giá-trị-1>  
    <lệnh-1> ;;  
  <giá-trị-2>  
    <lệnh-2> ;;  
  ...  
  <giá-trị-N>  
    <lệnh-N> ;;  
esac
```

Ví dụ: Tạo menu cho phép người dùng chọn  
Tạo tập tin thucdon có nội dung sau:

```
echo "Thuc don"  
echo "1. Liet ke thu muc hen hanh"  
echo "2. Duong dan thu muc hien hanh"  
read chon  
case $chon in  
  1) ls -l ;;  
  2) pwd ;;  
  *) echo "Khong hop le" ;;  
esac
```

# VI.1. CẤU TRÚC RỄ NHÁNH

```
#!/bin/sh

echo "Is it morning? Please answer yes or no"
read timeofday

case "$timeofday" in
    "yes" | "y" | "Yes" | "YES" ) echo "Good Morning";;
    "n*" | "N*" ) echo "Good Afternoon";;
    * ) echo "Sorry, answer not
recognised";;
esac

exit 0
```

# VI.1. CẤU TRÚC RỄ NHÁNH

```
#!/bin/sh

echo "Is it morning? Please answer yes or no"
read timeofday

case "$timeofday" in
    "yes" | "y" | "Yes" | "YES" )
        echo "Good Morning"
        echo "Up bright and early this morning?"
        ;;
    "[nN]*" )
        echo "Good Afternoon"
        ;;
    * )
        echo "Sorry, answer not recognised"
        echo "Please answer yes or no"
        exit 1
        ;;
esac

exit 0
```

## VI.2. CẤU TRÚC LẶP

- Vòng lặp **for**

```
for <biến> in <giá-trị-1> <giá-trị-2> <giá-trị-3> ...  
do  
    <danh sách lệnh>;  
done
```

Ví dụ: Hiển thị các giá trị  
Tạo tập tin hienthi có nội dung sau:

```
for gt in apple banana 34  
do  
    echo $gt;  
done
```

Kết quả: apple  
          banana  
          34

## VI.2. CẤU TRÚC LẶP

- VD:

```
for a in {1..20}
do
    host= 192.168.100.$a
    ping -c2 $host
done
```

```
1 #!/bin/bash
2 for a in {1..20}
3 do
4     host = 192.168.100.$a
5     ping -c2 $host
6 done
```

```
dai@dai-VMware-Virtual-Platform: ~/Desktop/HDHMMN_DoAn
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMMN_DoAn$ bash demo.sh
;; communications error to 192.168.100.1#53: timed out
;; communications error to 192.168.100.1#53: timed out
;; no servers could be reached
ping: usage error: Destination address required
;; communications error to 192.168.100.2#53: timed out
;; communications error to 192.168.100.2#53: timed out
;; no servers could be reached
ping: usage error: Destination address required
;; communications error to 192.168.100.3#53: timed out
;; communications error to 192.168.100.3#53: timed out
;; no servers could be reached
ping: usage error: Destination address required
;; communications error to 192.168.100.4#53: timed out
;; communications error to 192.168.100.4#53: timed out
;; no servers could be reached
ping: usage error: Destination address required
;; communications error to 192.168.100.5#53: timed out
;; communications error to 192.168.100.5#53: timed out
;; no servers could be reached
ping: usage error: Destination address required
;; communications error to 192.168.100.6#53: timed out
;; communications error to 192.168.100.6#53: timed out
;; no servers could be reached
```

```
1 #!/bin/bash
2 for a in {1..20}
3 do
4     host=192.168.100.$a
5     ping -c2 $host
6 done
```

```
dai@dai-VMware-Virtual-Platform: ~/Desktop/HDHMMN_DoAn
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMMN_DoAn$ bash demo.sh
PING 192.168.100.1 (192.168.100.1) 56(84) bytes of data.
--- 192.168.100.1 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1016ms

PING 192.168.100.2 (192.168.100.2) 56(84) bytes of data.
--- 192.168.100.2 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1056ms

PING 192.168.100.3 (192.168.100.3) 56(84) bytes of data.
--- 192.168.100.3 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1060ms

PING 192.168.100.4 (192.168.100.4) 56(84) bytes of data.
--- 192.168.100.4 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1061ms

PING 192.168.100.5 (192.168.100.5) 56(84) bytes of data.
--- 192.168.100.5 ping statistics ---
```

## VI.2. CẤU TRÚC LẶP

```
dai@dai-VMware-Virtual-Platform: ~/Desktop/HDHMNM_DoAn
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMNM_DoAn$ bash demo.sh
ls: cannot access 'f*.sh': No such file or directory
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMNM_DoAn$
```

```
#!/bin/sh
```

```
for file in $(ls f*.sh); do
    lpr $file
done
```

```
1#!/bin/bash
2
3for file in $(ls f*.sh); do
4    lpr $file
5done
```

## VI.2. CẤU TRÚC LẶP

- Vòng lặp **while**: vòng lặp thực hiện khi **điều kiện còn đúng**

```
while <điều kiện>  
do  
    <danh sách lệnh>;  
done
```

Ví dụ: Tạo tập tin vònglapwhile có nội dung sau:

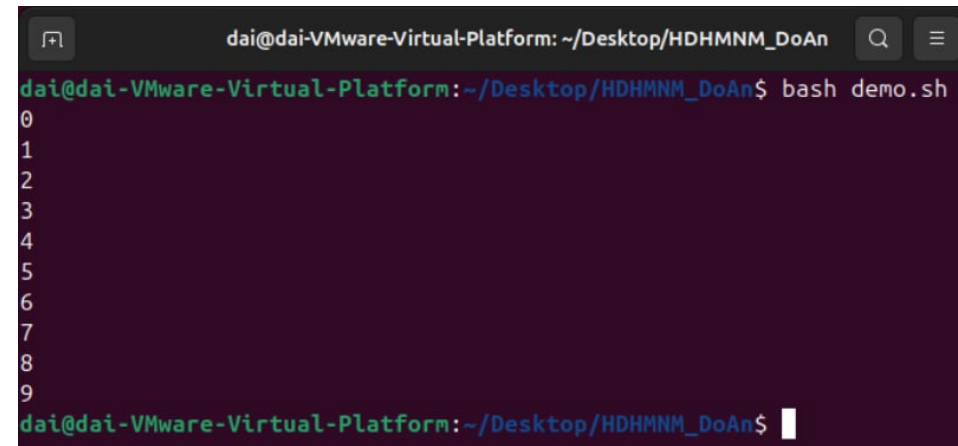
```
chon='y'  
while [ $chon = 'y' ] || [ $chon = 'Y' ]  
do  
    echo "Chao ban"  
    echo "An phim Y/y de tiep tục";  
    read chon  
done
```

## VI.2. CẤU TRÚC LẶP

- VD:

```
a=0
while [ $a -lt 10 ]
do
    echo $a
    let a=$a+1
done
```

```
1 #!/bin/bash
2 a=0
3 while [ $a -lt 10 ]
4 do
5     echo $a
6     let a=$a+1
7 done
```



```
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
6
7
8
9
dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMNM_DoAn$
```



## VI.2. CẤU TRÚC LẶP

```
#!/bin/sh

echo "Enter password"
read trythis

while [ "$trythis" != "secret" ]; do
    echo "Sorry, try again"
    read trythis
done

exit 0
```

## VI.2. CẤU TRÚC LẶP

- Vòng lặp **until**: vòng lặp kết thúc khi **điều kiện đúng**

```
until <điều kiện>  
do  
    <danh sách lệnh>;  
done
```

Ví dụ: Tạo tập tin vònglapuntil có nội dung sau:

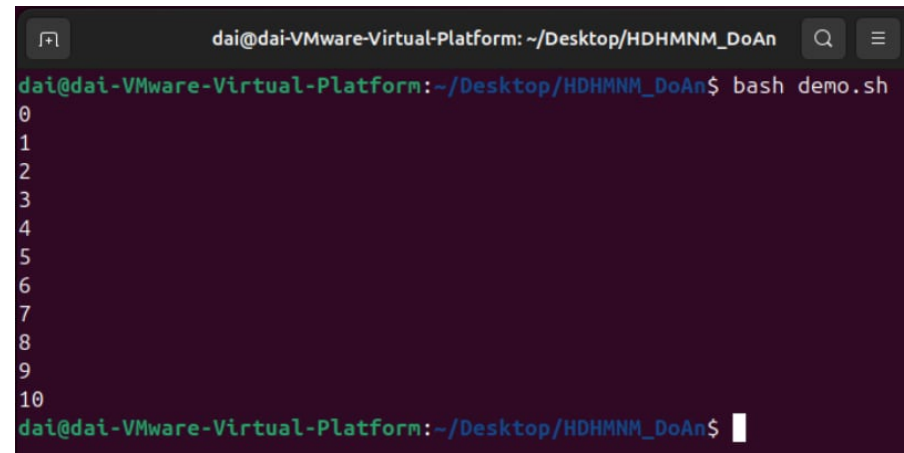
```
echo "Nhập vào số n: "  
read n  
until [ $n -le 10 ]  
do  
    echo "$n lớn hơn 10";  
    n=`expr $n -1`  
done
```

## VI.2. CẤU TRÚC LẶP

- VD:

```
a=0
until [$a -gt 10]
do
    echo $a
    let a=a+1
done
```

```
1 #!/bin/bash
2 a=0
3 until [ $a -gt 10 ]
4 do
5     echo $a
6     let a=a+1
7 done
```



A terminal window titled 'dai@dai-VMware-Virtual-Platform: ~/Desktop/HDHMMN\_DoAn'. The prompt is 'dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMMN\_DoAn\$'. The user has entered 'bash demo.sh'. The output shows a vertical list of numbers from 0 to 10, with each number on a new line. The prompt is now 'dai@dai-VMware-Virtual-Platform:~/Desktop/HDHMMN\_DoAn\$' with a cursor.

# VII. PHÉP TOÁN SỐ HỌC TRONG SHELL

- let
- Toán tử [ ]
- Toán tử (( ))
- expr
- bc

## VII.1 LET

- Lệnh **let** có thể được dùng để thực hiện trực tiếp các phép toán cơ bản. Trong khi sử dụng **let**, chúng ta sử dụng tên biến mà không cần có tiền tố \$

```
let result=no1+no2  
echo $result
```

- Toán tăng:

```
$ let no1++
```

- Toán giảm:

```
$ let no1--
```

```
$ let no+=6  
$ let no-=6
```

## VII.2. TOÁN TỬ [ ]

- Toán tử `[]` có thể được dùng giống như lệnh let

```
result=$(( no1 + no2 ))
```

- Sử dụng tiền tố \$ trong các toán hạng `[]` là hợp lệ

```
result=$(( $no1 + 5 ))
```

## VII.3. TOÁN TỬ (( ))

- Toán tử (( )) cũng có thể được dùng cho các phép toán số học. Tiền tố \$ với tên biến được dùng với toán hạng (( ))

```
result=$(( no1 + 50 ))
```

## VII.4. EXPR

- **expr** là 1 tiện ích được sử dụng cho các phép toán đơn giản:

```
expr toán_hạng_1 toán_tử toán_hạng_2
```

```
# 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
```



## VII.5. BC

- Tất cả các phương pháp trên không hỗ trợ số thực dấu chấm động, và chỉ hoạt động trên các số nguyên.
- **bc**, 1 tiện ích nâng cao cho các phép toán cao cấp. Nó có nhiều tùy chọn khác nhau. Chúng ta có thể thực hiện các phép toán dấu chấm động và sử dụng các hàm nâng cao
- Ví dụ:

```
echo "4 * 0.56" | bc  
2.24
```

```
no=54;  
result=`echo "$no * 1.5" | bc`  
echo $result  
81.0
```

## VII.5. BC

- **Lấy số thập phân với bc**
- Trong ví dụ sau, thông số `scale = 2` thiết lập số chữ số sau dấu thập phân là 2. Do đó, kết quả xuất ra của `bc` là 1 số với 2 số thập phân:

```
echo "scale=2;3/8" | bc  
0.37
```

## VII.5. BC

- **Chuyển đổi các hệ số với bc**
- Chúng ta có thể chuyển đổi qua lại giữa các hệ số như nhị phân, bát phân, thập phân ... với nhau qua bc:

```
#!/bin/bash
Desc: Number conversion
no=100
echo "obase=2;$no" | bc
1100100

no=1100100
echo "obase=10;ibase=2;$no" | bc
100
```

## VII.5. BC

- **Tính toán căn bậc 2 và lũy thừa**

```
echo "sqrt(100)" | bc  
echo "10^10" | bc
```