----------oOo----------

Phần 1: Thực hành

a. Phạm vi biến trong shell

Trong shell có hai loại biến: Biến cục bộ và biến toàn cục

- Biến cục bộ: Chỉ có thể truy cập bởi shell hiện thời
- Biến toàn cục: Có thể truy cập bởi shell hiện thời và tất cả các tiến trình con của shell này.

Ví dụ: *student@linux$ a=1*
student@linux$ vi testVar.sh
Gõ nội dung của tệp tin *testVar.sh* như sau:
!#/bin/bash
echo "a=$a"
a=2
echo "a=$a"
student@linux$ bash testVar.sh
Kết quả: a=
a=2

Để đặt một biến là toàn cục, dùng lệnh *export*
Cú pháp: export var1, var2, …, varN
Ví dụ: *student@linux$ a=1*
student@linux$ export a
student@linux$ vi testVar.sh
Gõ nội dung của tệp tin *testVar.sh* như sau:
!#/bin/bash
echo "a=$a"
a=2
echo "a=$a"
student@linux$ bash testVar.sh
Kết quả: a=1
a=2

b. Một số biến đặc biệt trong lập trình shell

$0	Tên file của script hiện tại
$n	Những biến tương ứng với các tham số mà một script được gọi. Ở đây n là một số thập phân dương ứng với vị trí của tham số (tham số đầu tiên là $1, tham số thứ 2 là $2,…)
$#	Số tham số tương ứng với một script
$* hoặc $@	Tất cả các đối số được trích dẫn, nếu một script gọi đến hai đối số, thì $* tương ứng với $1 $2
$?	Trạng thái exit của câu lệnh cuối cùng thực hiện

c. Lệnh exit và exit status

Các lệnh exit thường dùng để kết thúc một script, giống như trong chương trình C. Mỗi lệnh exit trả về một exit status. Trong Linux, khi một lệnh hoặc một script thực hiện, nó trả về hai giá trị. Nếu lệnh thực thi thành công sẽ trả về 0, ngược lại sẽ trả về khác 0 (thông thường là một mã lỗi).

Tương tự như vậy, một hàm trong một script sẽ trả về một exit status. Lệnh cuối cùng thực hiện trong hàm sẽ xác định exit status. Để biết được một giá trị trả về của một lệnh hay script, dùng biến đặc biệt có sẵn của shell là : **$?**. Sau khi một hàm trả về, **$?** sẽ đưa ra exit status của lệnh được thực hiện cuối cùng của hàm đó. Đây là cách trả về một giá trị trong một hàm của Linux shell.

Ví dụ:

```
echo Hello
echo $?
abeckdf
echo $?
```

d. Hàm trong lập trình shell

Hàm trong lập trình shell có cú pháp:

```
function-name()
{
    command1
    command2
    ...
    commandN
    return
}
```

Gọi hàm:

```
function-name arg1 arg2 arg3 argN
```

Ví dụ:

```
#!/bin/bash
sayHello()
{
        echo “Hello $LOGNAME, Toi la An!”
}
sayHello
```

Ví dụ:

```
cal()
{
        n1=$1
        op=$2
        n2=$3
        ans=0
        if [ $# -eq 3 ]; then
                ans=$(( $n1 $op $n2 ))
                echo "$n1 $op $n2 = $ans"
        else
                echo "Function cal requires at least three args"
        fi
        return
}
cal 5 + 10
cal 10 - 2
cal 10 / 2
```

Biến cục bộ và biến toàn cục trong một shell script

- Biến cục bộ: Chỉ có hiệu lực trong hàm. Để khai báo biến cục bộ, dùng từ khóa local
- Biến toàn cục: Có hiệu lực tại tất cả các hàm trong cùng script. Nếu không có từ khóa local, các biến sẽ được coi là biến toàn cục

Trường hợp đã có biến toàn cục, nhưng trong hàm lại khai báo biến cục bộ cùng tên, biến cục bộ sẽ có giá trị ưu tiên và hiệu lực cho đến khi hàm chấm dứt

Ví dụ: Viết một script có nội dung như sau:

```
#!bin/bash
sample_text=“global variable”
foo() {
        local sample_text=“local variable”
        local sample_text1=“local variable 1”
```

```
        echo "sample_text = $sample_text"
}
echo "sample_text = $sample_text"
foo
echo "sample_text = $sample_text"
echo "sample_text1 = $sample_text1"
```

Kết quả sau khi thực hiện:

sample_text = global variable
sample_text = local variable
sample_text = global variable
sample_text1 =

Trả về giá trị từ một hàm

Nếu muốn trả về giá trị cho hàm, dùng biến toàn cục để lưu lại giá trị hoặc *echo* in ra giá trị cần trả về trong hàm và dùng lệnh $(function_name) để lấy lại giá trị đó.

Ví dụ:

```
#!/bin/bash
sum()
{
        var=0
        for((i = 1; i<=10; i++))
        do
                var=`expr $var + $i`
        done
        echo $var
}
tong=$(sum)
echo "Tong la $tong"
```

Kết quả: Tong la 55

Ngoài ra, có thể dùng lệnh *return* để trả về giá trị của một hàm. Thông thường giá trị trả về của hàm được lưu trữ trong biến *$*?

Ví dụ:

```
#!/bin/bash
sum()
{
        var=0
        for i in 1 2 3 4
        do
```

```
        var=`expr $var + $i`
    done
    return $var
}
sum
echo "Tong la $?"
```

Một hàm cũng có thể gọi lại chính nó (đệ quy) hoặc có thể gọi đến các hàm khác.

Ví dụ:

```
#!/bin/sh
number_one () {
    echo "This is the first function speaking..."
     number_two
}
number_two () {
    echo "This is now the second function speaking..."
}
number_one
```

Phần 2: Bài tập:

1. Viết 1 shell trong đó có hàm sum() trả về tổng các đối số truyền vào của nó. In tổng vừa tính được ra màn hình?
2. Viết 1 shell trong đó chứa hàm count() có đối số truyền vào là tên của một thư mục, và trả về số lượng file trong thư mục đó?
3. Viết 1 shell tính và in ra tổng giai thừa sau: Sum= 1! + 2! + .. + n!. Với n nhập vào từ bàn phím hoặc đối dòng lệ
4. Viết 1 shell thực hiện các yêu cầu sau:
    1. Hiển thị menu để lựa chọn
    2. Nhập vào một lựa chọn từ bàn phím
    3. Thực hiện tạo menu theo lựa chọn vừa nhập:

        Lựa chọn 1: Hiển thị thư mục đang hiện hành

        Lựa chọn 2: Hiển thị tất cả nội dung của thư mục hiện hành

        Lựa chọn 3: Hiển thị các tiến trình đang hoạt động

        Lựa chọn 4: Đổi tên tất cả các tệp tin có đuôi .doc sang .txt trong thư mục hiện hành

        Lựa chọn 5: Tìm kiếm và hiển thị các tệp tin có tên bắt đầu bằng chuỗi abc trong thư mục hiện hành

        Lựa chọn 6: Tìm kiếm và hiển thị các dòng có chứa chữ số trong thư mục hiện hành.

Lựa chọn 0: Thoát khỏi menu

Tổ chức chương trình thành các hàm con tương ứng với mỗi lựa chọn. Và thử nghiệm các hàm con đó.

Các bài tập ôn luyện

1. Viết chương trình shell tìm và in ra số nguyên tố trong một dãy số đầu vào.
2. Viết chương trình shell tìm số đảo ngược của một số

Phần 3: Liên lạc

STT	Họ và tên	Email	ĐT
1	Hà Mỹ Linh	Halinh.hus@gmail.com	
3	Trần Thị Hương	Tranthihuong.hus@gmail.com	