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

Phần 1: Thực hành

a. Makefile

Makefile là gì?

+ Makefile là một file dạng script chứa các thông tin:
  - Cấu trúc project (file, sự phụ thuộc)
  - Các lệnh để tạo file
+ Lệnh make sẽ đọc nội dung Makefile, hiểu kiến trúc của project và thực thi các lệnh

Ví dụ:

- Project chứa 4 file: Shape.java, Rectangle.java, Square.java, Main.java
- 2 đối tượng Square và Rectangle cùng cài đặt giao diện Shape
- File thực thi là Main.java

Nội dung các file lần lượt như sau:

File Shape.java

```
public interface Shape{ double getArea();}
```

File Square.java

```
public class Square implements Shape{
    private double a;
    public Square(double a) {   this.a = a;}
    public double getArea(){return a*a;}
}
```

File Rectangle.java

```
public class Rectangle implements Shape{
    private double a, b;
    public Rectangle(double a, double b) {this.a = a;this.b = b;    }
    public double getArea(){return a*b;}
}
```

File Main.java

```
import java.util.ArrayList;
public class Main{
    public static void main(String[] args){
        ArrayList<Shape> shapes = new ArrayList<Shape>();
```

```
            shapes.add(new Square(5));
            shapes.add(new Rectangle(2, 3));
            System.out.println(shapes.get(0).getArea());
            System.out.println(shapes.get(1).getArea());
        }
    }
```

File Makefile có nội dung sau:

```
Main: main.class
    java Main
main.class: Main.java Rectangle.class Square.class
    javac Main.java
Rectangle.class: Rectangle.java Shape.class
    javac Rectangle.java
Square.class: Square.java Shape.class
    javac Square.java
Shape.class: Shape.java
    javac Shape.java
```

Thứ tự thực hiện:

Khi bạn thực hiện lệnh make, chương trình make sẽ nhảy đến phần đầu tiên là Main với mục đích để tạo ra nó, để làm được điều đó make đi kiểm tra lần lượt nội dung sau dấu : xem tệp tin main.class đã tồn tại chưa. Tệp đầu tiên là main.class chưa có, cần phải tìm câu lệnh nào đó mà ở đó main.class được tạo ra, make tìm ra dòng thứ 3 và nó nhảy đến thực hiện câu lệnh ở dòng này đề tạo ra main.class. Sau khi tạo ra main.class, make tiến hành các bước tương tự. Sau khi tất cả các tệp liên quan được tạo ra, make mới có thể tạo ra tệp chạy cuối cùng là Main.

Vậy make thực hiện theo thứ tự như sau:

+ Tạo ra các file object trước (main.class)
+ Tạo ra chương trình nhị phân cuối cùng từ các file object đã được tạo ra trước đó (sum)

Sau khi đã tạo các file với nội dung như trên, vào thư mục chứa các file trên thực hiện lệnh:

```
student@linux ~$make
```

Kết quả:

25
30

b. Biến môi trường

Linux Shell xác định các biến để điều khiển môi trường của người sử dụng đối với mỗi phiên sử dụng. Việc đặt các biến này sẽ xác định với hệ thống những tham số như thư mục nào được sử dụng làm thư mục chính, nơi đặt mail, những thư mục nào được sử dụng mặc định khi bạn gọi đến các lệnh Linux,… Một số biến hệ thống có thể được đặt trong tệp khởi động (startup file) và được đọc khi bạn đăng nhập. Trong tệp khởi động có thể đặt những câu lệnh Linux, và những lệnh này sẽ được thực hiện khi bạn login

Xem tất cả biến môi trường hiện có

env

Xem giá trị của 1 biến môi trường cụ thể

echo $ten_bien_moi_truong

- Ví dụ:
  student@linux ~$ echo $HOSTNAME -- Hiển thị tên máy tính

  Kết quả: linux

- *student@linux ~$ echo $HOME* -- Hiển thị thư mục mặc định của người dùng hiện tại

  Kết quả: /home/student

Đặt giá trị cho biến môi trường

Cú pháp: export ten_bien_moi_truong=gia_tri_cua_bien_moi_truong

- Ví dụ
  student@linux ~$ export TAILIEU=/home/student/tailieu
  student@linux ~$ echo $TAILIEU

  Kết quả: /home/student/tailieu

c. Biến trong Bash Shell

Giống như các ngôn ngữ lập trình khác, Bash shell có sử dụng biến, nhưng khác là không cần phải khai báo kiểu.

Cách gán giá trị cho biến

Cú pháp: *ten_bien=gia_tri*

(Chú ý: Không có dấu cách trong cú pháp này)

- Ví dụ: File test.sh
  #!/bin/bash
  var=100
  echo $var

Sử dụng Bash Shell rất linh hoạt, không những có thể gán 1 giá trị cụ thể vào biến và nó còn cho phép gán kết quả của một câu lệnh cho biến

Cú pháp: ten_bien=`cau_lenh` (Chú ý đây là ký tự ` nằm ở dưới phím ESC chứ không phải ký tự nháy đơn ‘)

- Ví dụ: File test.sh
  #!/bin/bash
  var=`pwd`
  echo $var

Để tăng tính tương tác, bạn có thể viết shell script cho phép yêu cầu nhập giá trị cho biến ngay khi chạy nó. Sử dụng câu lệnh read để đạt được điều này.

Cú pháp: read ten_bien

- Ví dụ:
  #!/bin/bash
  echo "Ban ten gi: "
  read ten
  echo "Chao ban $ten"

Bash shell không cho phép chúng ta thực hiện tính toán trực tiếp như những ngôn ngữ khác, do nó không phân biệt kiểu của biến. Thay vì đó, nó cung cấp lệnh expr khi chúng muốn thực hiện việc tính toán trên biến.

Cú pháp: expr $bien_1 + $bien_2

- Ví dụ:
  student@linux ~$ expr $a + $b

  Đặt trong file shell script test.sh

```
#!/bin/bash
var=`expr 1+3`
echo The result is $var
```

Lưu ý: Phải có khoảng trắng giữa toán tử và toán hạng. Nếu ở trên đổi thành 1+3 hoặc 1+ 3 hoặc 1 +3, thì đều không cho kết quả hoặc có thông báo sai.

Đối dòng lệnh

Đối dòng lệnh là các giá trị được đưa vào như là 1 tùy chọn (option) ở câu lệnh chạy.

Ví dụ: ./test.sh a b c

a,b,c là các đối dòng lệnh. a là đối dòng lệnh thứ nhất, b là đối dòng lệnh thứ 2, c là đối dòng lệnh thứ 3,…

Trong bash shell, chúng ta có thể lấy giá trị đối dòng lệnh theo cách sau:

```
#!/bin/sh
echo "Ten tep [$0]"
echo "Doi dong lenh thu nhat [$1]"
echo "Doi dong lenh thu hai [$2]"
echo "Doi dong lenh thu ba [$3]"
```

Ở ví dụ trên, $0 là 1 biến đặc biệt, nó cho phép ta lấy ra tên tệp chứa biến này. $1, $2,$3 tương ứng là giá trị của các đối dòng lệnh thứ 1,2,3.

d. Cấu trúc rẽ nhánh

if – else

Cú pháp của if-else trong bash shell cũng tương tự như những ngôn ngữ khác

```
if [ điều kiện ]
then
        <lệnh_cần_chạy>
 elif  [ điều_kiện ] then <lệnh_cần_chạy>...
else <lệnh_cần_chạy>
fi
```

- Ví dụ: So sánh 2 số a và b

```
#!/bin/sh
echo “Nhap so a:”
read a
```

```
echo “Nhap so b:”
read b
if [ $a -lt $b ]
then
        echo “so a nho hon so b.”
elif [ $a –eq $b ]
then
        echo “so a bang so b.”
else
        echo “so a lon hon so b.”
fi
```

case

Với 1 bài toán có nhiều trường hợp, ta có thể sử dụng cấu trúc case để giải quyết. Cấu trúc rẽ nhánh này tương tự như cấu trúc switch … case trong C

Cú pháp:

```
case <biến>
in
    giá_trị_1)
            <lệnh>
    giá_trị_2)
            <lệnh>
    giá_trị_3)
            <lệnh>
    ...
    *) #còn lại
    exit
esac
```

- Ví dụ:

```
#!/bin/bash
case $1
in
    1) echo "Ban vua chon option 1"
            exit;;
    2) echo "Ban vua chon option 2"
            exit;;
```

```
        3) echo "Ban vua chon option 3"
                exit;;
        *) echo "Exit";
                exit;;
    esac
```

e. Đặt lịch công việc tự động với crontab

Cron hoạt động bằng cách đọc từng dòng file crontab, mỗi dòng là 1 lịch làm việc hoàn chỉnh, được viết theo cú pháp của biểu thức CRON.
Biểu thức cron thường bao gồm 2 thành phần:

- Định nghĩa lịch trình chạy
- Tập lệnh muốn thực thi theo thời gian trên

Ví dụ sau đây là một tiến trình đã được đặt lịch chạy tự động trong crontab

```
1 2 12 5 */root/test.sh
```

(Phần lịch trình tô màu vàng, phần tập lệnh shell script sẽ thực thi tô màu xanh)

Phần lịch trình chạy có dạng như sau:
phút giờ ngày tháng thứ

- Phút có giá trị từ 0 đến 59 hoặc là * (tất cả các phút)
- Giờ có giá trị từ 0 đến 23 hoặc * (tất cả các giờ)
- Ngày có giá trị 1 đến 31 hoặc * (tất cả các ngày)
- Tháng có giá trị từ 1 đến 12 hoặc * (tất cả các tháng)
- Thứ có giá trị 0 (Chủ nhật), 1 (thứ hai)...6 (thứ 7) hoặc * (tất cả các thứ)

Ví dụ bạn muốn chạy 1 file tên test1.sh vào lúc 2g1' sáng ngày 12 tháng 5 (bất kể thứ) thì lịch chạy như sau:
*1 2 12 5 * /root/test1.sh*
(* biểu thị everything, tức là bất kể thứ nào)

Nếu bạn muốn chạy 1 file tên test2.sh vào 15g chiều vào chủ nhật (bất kể ngày tháng) thì lịch chạy như sau:
*0 15 * * 0 /root/test2.sh*

và nếu bạn muốn chạy 1 file tên test3.sh vào mỗi giờ trong ngày (bắt đầu ở phút thứ 8 của giờ đó) thì:
*8 * * * * /root/test3.sh*

Để xem lịch trình chạy đã được cấu hình từ trước

Cú pháp: *crontab –l*

Để sửa, thêm lịch trình chạy

Cú pháp: *crontab –e*

Phần 2: Bài tập thực hành

- Bài 1
  Viết 1 shell script in ra màn hình các thông tin sau:
  – Xâu: "I am a student"
  – Thư mục hiện hành
  – Tập tin và thư mục, kể cả các thư mục ẩn trong thư mục hiện hành
  – Ngày và giờ hiện tại
- Bài 2:
  Viết 1 shell script khi chạy cho phép người dùng nhập tên file, nếu file đó tồn tại thì sẽ in ra nội dung của file, nếu không hiển thị thông báo "File ten_file không tồn tại"
- Bài 3:
  In ra màn hình câu: hoặc Chào buổi sáng , hoặc chào buổi trưa, hoặc chào buổi tối, tùy thuộc vào thời điểm hiện tại
  Gợi ý: Lệnh date +%H sẽ in ra giờ hiện tại.
- Bài 4:
  Viết 1 shell script in ra màn hình nội dung:
  Xin moi ban chon hanh dong?
  1. Xem dung luong su dung cua may tinh
  2. Xem noi dung thu muc hien hanh
  3. Xem cac tien trinh dang chay tren may tinh duoi dang cay
  4. Xem ten nguoi dung dang nhap he thong

Khi người dùng nhập vào lựa chọn thì màn hình sẽ hiển thị kết quả tương ứng.
Gợi ý: Sử dụng câu lệnh free để xem hiện tại máy tính đang sử dụng RAM như thế nào

```
             total       used       free     shared    buffers     cached
Mem:       1027672     939560      88112      43652     132460     500160
-/+ buffers/cache:     306940     720732
Swap:            0          0          0
```

- Bài 5: Viết 1 shell script để lưu trữ tình hình sử dụng RAM của máy tính trong khoảng 1h vào 1 file đặt tên là memory_usage.log

Phần 3: Liên lạc

STT	Họ và tên	Email
1	Nguyễn Minh Hải	nguyenminhhai06@gmail.com
2	Nguyễn Thị Huyên	nthuyen.bmth@gmail.com