Lập trình shell

Làm thế nào để viết shell script:

1. Sử dụng bất kỳ trình soạn thảo nào như: vi, mcedit...
2. Sau khi viết shell script thì thiết lập quyền thực thi cho nó theo cấu trúc:

chmod permission your-script-name

Ví dụ:

$ chmod +x your-script-name

$ chmod 755 your-script-name

3. Thực thi script bằng cấu trúc:

bash your-script-name

sh your-script-name

./your-script-name

Ví dụ:

$ bash bar

$ sh bar

$ ./bar

Ghi chú: Cấu trúc ./ có nghĩa là thư mục hiện hành, nhưng . (dot) nghĩa là thực thi lệnh trong shell với shell hiện hành. Cấu trúc dot như sau:

. shell_script

Ví dụ:

$. foo

Shell script sau sẽ in "Knowledge is Power" trên màn hình:

$ vi first

```
#
# My first shell script
#
clear
echo "Knowledge is Power"
```

Sau khi lưu file, thay đổi quyền (chmod 755 first) => chạy script này như sau:

$ ./first

Ghi chú: Shell script file có bằng phần mở rộng là .sh để dễ dàng xác định đó là shell script.

Các biến trong Shell:

Để xử lý dữ liệu/thông tin, dữ liệu phải được giữ trong bộ nhớ RAM của máy tính. RAM được chia thành nhiều vị trí nhỏ, và mỗi vị trí có một số duy nhất được gọi là địa chỉ bộ nhớ - được sử dụng để lưu dữ liệu. Bạn có thể gán tên cho vùng nhớ này => gọi là biến bộ nhớ hoặc biến.

Trong Linux (Shell), có hai loại biến:

1. Biến hệ thống - được tạo và duy trì bởi chính Linux. Loại biến này được định nghĩa bằng các mẫu tự hoa.

2. Biến do người dùng định nghĩa (UDV - User Defined variables) - được tạo và duy trì bởi người dùng. Loại biến này được định nghĩa bằng các mẫu tự thường.

Bạn có thể xem danh sách các biến hệ thống bằng lệnh set hoặc env. Một vài biến hệ thống quan trọng:
BASH=/bin/bash - tên shell.
BASH_VERSION=1.14.7(1) - phiên bản của shell.
COLUMNS=80 - số cột cho màn hình.
HOME=/home/vivek - thư mục nhà.
LINES=25 - số dòng của màn hình.
LOGNAME=students - tên đăng nhập của người dùng.
OSTYPE=Linux - Loại hệ điều hành.
PATH=/usr/bin:/sbin:/bin:/usr/sbin - Thiết lập đường dẫn.
PS1=[¥u@¥h ¥W]¥$ - thiết lập prompt.
PWD=/home/students/Common - thư mục hiện hành.
SHELL=/bin/bash - tên shell.
USERNAME=vivek - user nào hiện đang đăng nhập vào PC.
Bạn có thể in ra bất kỳ biến môi trường nào như sau:
$ echo $USERNAME
$ echo $HOME

Làm thế nào để định nghĩa UDV:
Sử dụng cấu trúc sau:
variable name=value - value được gán cho 'variable name' và value phải nằm bên phải dấu =
Ví dụ:
$ no=10
$ vech=Bus

Nguyên tắc đặt tên biến (cả UDV và biến hệ thống):
1. Biến phải bắt đầu bằng ký tự alphanumeric hoặc underscore (_), theo sau bởi một hoặc nhiều ký tự alphanumeric.
Ví dụ:
Các biến hợp lệ: HOME, SYSTEM_VERSION, vech, no…
2. Không có khoảng trắng giữa hai bên dấu bằng khi gán giá trị biến.
Ví dụ:
Các khai báo sau sẽ có lỗi:
$ no =10
$ no= 10
$ no = 10
3. Phân biệt chữ hoa và thường.
Ví dụ:
Các biến sau sẽ khác nhau:
$ no=10

$ No=11
$ NO=20
$ nO=2
4. Bạn có thể định nghĩa biến NULL như sau:
$ vech=
$ vech=""
Khi in ra các biến NULL này, thì không có gì trên màn hình bởi vì chúng không có giá trị.
5. Không sử dụng ?, *...để đặt tên cho biến.
In và truy cập giá trị của UDV:
$variablename
hoặc echo $variablename

Tính toán trong Shell:
Sử dụng các phép toán số sau theo cấu trúc sau:
expr op1 math-operator op2
Ví dụ:
$ expr 1 + 3
$ expr 2 - 1
$ expr 10 / 2
$ expr 20 % 3
$ expr 10 ¥* 3
$ echo `expr 6 + 3`

Trích dẫn (quote):
Có 3 loại quote:
1. " (double quote): Bất kỳ thứ gì trong đây đều bị hủy ý nghĩa của ký tự đó (trừ ¥ và $)
2. ' (single quote): Duy trì không thay đổi.
3. ` (back quote): thực thi lệnh.
Ví dụ:
$ echo "Today is date" => in ra Today is date
$ echo "Today is `date`". => in ra Today is Tue Jan....

Trạng thái kết thúc (exit status):
Mặc định trong Linux nếu lệnh/shell script được thực thi, nó sẽ trả về hai loại giá trị - được sử dụng để xem lệnh/shell script được thực thi thành công hay không.
1. Nếu trả về giá trị là 0 - thành công.
2. Không phải 0, lệnh không thành công hoặc có một vài lỗi khi thực thi lệnh/shell script.
=> giá trị này gọi exit status.
Ví dụ:
unknow1file không tồn tại trong đĩa cứng thì lệnh “rm unknow1file” sẽ in ra lỗi như sau:
rm: cannot remove `unkowm1file': No such file or directory.
Và sau đó nếu bạn gõ lệnh $ echo $? thì nó sẽ in ra giá trị nonzero để chỉ có lỗi.

Câu lệnh read:

Sử dụng để nhận dữ liệu từ bàn phím và lưu vào biến.

Cấu trúc:

read variable1, variable2,...variableN

Ví dụ:

Script sau sẽ hỏi tên người dùng và chờ họ nhập vào từ bàn phím. Sau khi người dùng nhập tên và nhấn Enter thì tên sẽ được lưu trong biến fname.

```
$ vi sayH
#
#Script to read your name from key-board
#
echo "Your first name please:"
read fname
echo "Hello $fname, Lets be friend!"
```

Chạy nó như sau:

```
$ chmod 755 sayH
$ ./sayH
Your first name please: vivek
Hello vivek, Lets be friend!
```

Wild cards:

1. *: phù hợp với bất kỳ chuỗi hoặc nhóm ký tự nào.

Ví dụ:

$ ls * - hiển thị tất cả các file.

$ ls a* - hiển thị tất cả các file mà tên của nó bắt đầu bằng mẫu tự 'a'.

$ ls *.c - hiển thị tất cả các file có bằng mở rộng là .c

2. ?: phù hợp bất kỳ 1 ký tự nào.

Ví dụ:

$ ls ? - hiển thị tất cả các file mà tên của nó có chiều dài 1 ký tự.

$ ls fo? - hiển thị tất cả các file mà tên của nó có 3 ký tự và tên file bắt đầu bằng fo.

3. [...]: phù hợp bất kỳ 1 trong các ký tự trong dấu ngoặc.

Ví dụ:

$ ls [abc]* - thể hiện tất cả các file bắt đầu với mẫu tự a, b, c.

Ghi chú:

[..-..]: Một cặp các ký tự được phân chia bởi dấu trừ để ghi nhận một dãy.

Ví dụ:

$ ls /bin/[a-c]* - hiển thị tất cả các file bắt đầu với mẫu tự a, b, c.

Nếu ký tự đầu tiên theo sau [là ! hoặc ^, thì bất kỳ ký tự nào không phù hợp sẽ được hiển thị.

$ ls /bin/[!a-o]
$ ls /bin/[^a-o]
=> sẽ hiển thị tất cả các file trong thư mục bin mà ký tự đầu tiên không phải là a, b, c...o.

Nhiều lệnh trên một dòng lệnh:
command1;command2
Ví dụ:
$ date;who - sẽ in ra ngày hiện tại theo sau là tên người dùng đăng nhập hiện tại.

Dòng lệnh:
Khi thực thi lệnh sau (giả sử file "grate_stories_of" không tồn tại trên hệ thống).
$ ls grate_stories_of - sẽ in ra thông báo “grate_stories_of: No such file or directory”
ls là tên của lệnh và shell sẽ thực thi lệnh này.
Từ đầu tiên của dòng lệnh là ls - tên của lệnh cần thực thi.
Phần còn lại là các tham số của lệnh này.
Ví dụ:
$ tail +10 myf - tên lệnh là tail và tham số là +10 và myf.
Ghi chú: $# nắm giữ các tham số của dòng lệnh. $* hoặc $@ tham chiếu đến các tham số được chuyển đến script.

Chuyển hướng:
Hầu hết các lệnh đều kết xuất ra màn hình hoặc nhận tham số từ bàn phím. Nhưng trong Linux thì có thể gửi kết xuất đến hoặc đọc dữ liệu từ tập tin.
Ví dụ:
$ ls cho kết xuất ra màn hình, để gửi kết xuất ra tập tin thì sử dụng lệnh sau:
$ ls > filename
1) Ký hiệu >:
command > filename - kết xuất kết quả của lệnh ra file. Nếu như file tồn tại, nó sẽ ghi đè lên, ngược lại file mới sẽ được tạo.
2) Ký hiệu >>:
command >> filename - kết xuất kết quả đến phần cuối của file. Nếu như file tồn tại, nó sẽ được mở và thông tin mới được ghi vào cuối file, không sợ mất dữ liệu/thông tin trước. Và nếu file không tồn tại, thì file mới được tạo.
Ví dụ:
$date >> myfiles
3) Ký hiệu <:
command < filename - nhận dữ liệu từ file thay vì bàn phím.
Ví dụ:
$ cat < myfiles
$ sort < sname > sorted_names - lệnh sort lấy dữ liệu từ file tên là sname và kết xuất kết quả ra file tên là sorted_names.

Pipes:

Định nghĩa: Một pipe là một nơi lưu tạm kết xuất của một lệnh và sau đó chuyển vào input của lệnh thứ hai. Pipe được sử dụng để chạy nhiều hơn 2 lệnh trên cùng một dòng lệnh.

Filter:

Nếu một lệnh chấp nhận input từ stdin và xuất nó ra stdout thì được gọi là filter. Một filter thực thi một vài xử lý trên input và cho ra ouput. Ví dụ: Giả sử bạn có một file được gọi là 'hotel.txt' với 100 dòng dữ liệu. Và bạn chỉ muốn in ra dòng từ 20 đến 30 và lưu kết qủa này vào file gọi là 'hlist' thì sử dụng lệnh sau:

$ tail +20 < hotel.txt | head -n30 >hlist1

=> Ở đây lệnh head là một filter nhận input từ lệnh tail (lệnh tail bắt đầu chọn từ dòng 20 trong file 'hotel.txt'. Và chuyển các dòng này làm input cho head, cuối cùng chuyển ouput đến file 'hlist'.)

Process:

Tiến trình là một chương trình (lệnh) để thực thi một công việc xác định. Trong Linux khi bạn khởi tạo một tiến trình, nó sẽ cho tiến trình đó một con số gọi là PIP hoặc process-id, PID khởi đầu từ 0 đến 65535.

Ví dụ:

$ls -lR - lệnh ls sẽ liệt kê các file trong thư mục hoặc tất cả thư mục con trong thư mục hiện hành - nó là một tiến trình.

Tại sao có tiến trình:

Vì Linux là hệ điều hành đa người dùng, đa nhiệm. Nghĩa là bạn có thể chạy nhiều hơn 2 tiến trình đồng thời nếu bạn muốn.

Ví dụ:

Để tìm nhiều file mà bạn có trên hệ thống, bạn có thể thực hiện lệnh sau:

$ ls / -R | wc -l

=> lệnh này sẽ mất nhiều thời gian để tìm kiếm tất cả các file trên hệ thống, vì thế bạn có thể chạy lệnh này ở background:

$ ls / -R | wc -l &

Do đó một thể hiện của lệnh đang chạy được gọi là process và một con số được in ra bởi shell gọi là process-id (PID), PID này có thể sử dụng để xác định tiến trình đang chạy.

Các lệnh liên quan đến process:

• ps - xem tiến trình đang chạy.
• kill - dừng bất kỳ tiến trình nào bởi PID của nó.
• killall- dừng tiến trình bởi tên của nó.
• ps -ag - lấy thông tin về tất cả các tiến trình đang chạy.
• kill 0 - dừng tất cả các tiến trình trừ shell của bạn.
• command & - chạy lệnh ở background.
• ps aux - hiển thị chủ sở hữu của các tiến trình cùng với các tiến trình đó.
• ps ax | grep process-U-want-to-see - xem từng tiến trình đang chạy hoặc không chạy.
• top - xem tiến trình đang chạy và các thông tin khác như bộ nhớ, CPU usage cùng với thời gian thực.

• pstree - hiển thị cây các tiến trình.