Bài Tập 5.2. Thực hành lập trình Shell

#Lê văn triệu (B1606947)

- Thực hành chủ đề Lập trình Shell:

https://sites.google.com/site/ptpmmnm/bai-tap/shell-script

- **Bài Tập 0:**

```
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$ nano bt0.s
h
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$ chmod 755
bt0.sh
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$ cat bt0.sh
#!/bin/bash
clear
echo "hello World"
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$ ./bt0.sh
hello World
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$
```

- **Bài tập 1:**

```
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$ ./bt1.sh
-ten nguoi dung la: levantrieu
-thu muc hien hanh: /home/levantrieu/PTPMMNM/levantrieu/baitap-shelcript
-tap tin va thu muc,tap tin an: . .. bt0.sh bt1.sh
-ngay gio hien tai: Thứ ba, 24 Tháng 3 năm 2020 17:44:49 +07
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$ ls
bt0.sh  bt1.sh
```

- **Bài tập 2:**

```
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$ cat bt2.sh
#!/bin/bash
str_ho="le"
str_ten="trieu"
echo -e "nhap ho cua ban: \c"
read ho
echo -e "nhap ten cua ban: \c"
read ten
if [ $ho = $str_ho ] && [ $ten = $str_ten ]
        then echo "=> ho&ten cua ban: "$ho $ten
fi

levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$ ./bt2.sh
nhap ho cua ban: le
nhap ten cua ban: trieu
=> ho&ten cua ban: le trieu
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$
```

- **Bài tập 3:**

```
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$ cat bt3.sh
#!/bin/bash
h=$(date +%H)
if [[ $h -gt 1 && $h -le 9 ]];
then
         echo "Chao buoi sang!"
elif [[ $h -gt 10 && $h -le 12 ]];
then
        echo "Chao buoi trua!"
elif [[ $h -gt 13 && $h -le 17 ]];
then
        echo "Chao buoi chieu!"
elif [[ $h -ge 18 && $h -le 24 ]];
then
         echo "Chao buoi toi!"
fi

echo "Bay h la: "$h
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$ ./bt3.sh
Chao buoi chieu!
Bay h la: 15
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$
```

- **Bài tập 4:**

```
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$ cat bt4.sh
#!/bin/bash
echo  "MENU"
echo "1.Cafe den"
echo "2.Cafe da"
echo "3.Cafe sua"
echo "4.Cafe cao"
echo "ban chon thuc uong nao (1-4)?"
read so
if [ $so = 1 ]
        then echo "ban da chon Cafe den"
elif [ $so = 2 ]
        then echo "ban da chon Cafe da"
elif [ $so = 3 ]
        then echo "ban da chon Cafe sua"
elif [ $so = 4 ]
        then echo "ban da chon Cafe cao"
fi
```

⇨ **Kết quả là:**

```
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$ ./bt4.sh
MENU
1.Cafe den
2.Cafe da
3.Cafe sua
4.Cafe cao
ban chon thuc uong nao (1-4)?
4
ban da chon Cafe cao
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$
```

- **Bài tập 5:**

```
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$ cat bt5.sh
#!/bin/bash
echo "nhap 2 tham so ban can tin:"
read a
read b
echo  "Cac phep toan"
echo "1.Cong (+)"
echo "2.Tru (-)"
echo "3.Nhan (*)"
echo "4.Chia (/)"
echo "q.Ket thuc (q)"
echo "ban chon phep tinh nao ?"
read pt
if [ $pt = 1 ]
        then echo "$a+$b=`expr $a + $b`"
elif [ $pt = 2 ]
        then echo "$a-$b=`expr $a - $b`"
elif [ $pt = 3 ]
        then echo "$a*$b=`expr $a \* $b`"
elif [ $pt = 4 ]
        then echo "$a/$b=`expr $a / $b`"
elif [ $pt = q ]
        then exit 0
fi
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$
```

⇨ **Kết quả:**

```
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$ ./bt5.sh
nhap 2 tham so ban can tin:
4
3
Cac phep toan
1.Cong (+)
2.Tru (-)
3.Nhan (*)
4.Chia (/)
q.Ket thuc (q)
ban chon phep tinh nao ?
2
4-3=1
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$
```

- **Bài tập 6:**

```
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$ c
at bt6.sh
echo -e "moi ban nhap vao chuoi can hien thi: \c"
read str
echo -e "so lan ban can in: \c"
read num
for ((i=1; i<=$num; i++))
do
        echo "$str $num"
done
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$ .
/bt6.sh
moi ban nhap vao chuoi can hien thi: chaoban
so lan ban can in: 4
chaoban 4
chaoban 4
chaoban 4
chaoban 4
levantrieu@levantrieu-VirtualBox:~/PTPMMNM/levantrieu/baitap-shelcript$
```

- Trả lời câu hỏi:

https://www.softwaretestinghelp.com/shell-scripting-interview-questions/

1. **Shell là gì?**
   - ⇨ Shell là một trình thông dịch lệnh, nó diễn giải lệnh mà người dùng đưa ra cho kernel. Nó cũng có thể được định nghĩa là một giao diện giữa người dùng và hệ điều hành.

2. **Shell Scripting là gì?**
   - ⇨ Shell scripting không có gì ngoài chuỗi hoặc chuỗi các lệnh UNIX được viết trong một tệp văn bản thuần túy. Thay vì chỉ định một công việc / lệnh tại một thời điểm, trong kịch bản lệnh shell, chúng tôi đưa ra một danh sách các lệnh UNIX như danh sách việc cần làm trong một tệp để thực thi nó.

3. **Tầm quan trọng của việc viết Shell ScScript là gì?**

- ⇨ Shell script lấy đầu vào từ người dùng, tập tin và hiển thị nó trên màn hình.
- ⇨ Shell scripting rất hữu ích trong việc tạo các lệnh của riêng bạn.
- ⇨ Nó rất hữu ích trong việc tự động hóa một số nhiệm vụ trong cuộc sống hàng ngày.
- ⇨ Nó rất hữu ích để tự động hóa các nhiệm vụ quản trị hệ thống.
- ⇨ Chủ yếu là tiết kiệm thời gian.

4. Liệt kê một số lệnh UNIX phổ biến và được sử dụng rộng rãi nhất.

⇨ Ls, cd, mkdir, rmdir, cp, rm, mv, more, touch, cat, date, diff, find, finger, who, kill, ps, …

5. Các chương trình Shell được lưu trữ trong tập tin nào?

⇨ Các chương trình Shell được lưu trữ trong một tệp gọi là sh.

6. Các loại shell khác nhau ?

⇨ Chủ yếu có 4 loại vỏ quan trọng được sử dụng rộng rãi:

Bourne Shell (sh), C Shell (csh), Korn Shell (ksh), Bourne Again Shell (bash).

7. Ưu điểm của C Shell so với Bourne Shell là gì?

⇨ -C shell cho phép đặt bí danh cho các lệnh, tức là người dùng có thể đặt bất kỳ tên nào cho sự lựa chọn của mình cho lệnh. C shell cho phép đặt bí danh cho các lệnh, tức là người dùng có thể đặt bất kỳ tên nào cho sự lựa chọn của mình cho lệnh.

- C shell cung cấp tính năng lịch sử lệnh. C shell nhớ lệnh đã gõ trước đó. Vì vậy, nó tránh gõ lệnh nhiều lần.

8. Trong môi trương Unix có bao nhiêu kernels và shells hiện tại có sẳn?

⇨ Trong một môi trường UNIX điển hình, chỉ có một kernel và nhiều shell.

9. Là 1 trình biên dịch riêng biệt cần thiết để thực hiện một chương trình shell?

⇨ Một trình biên dịch riêng biệt là không cần thiết để thực hiện một chương trình shell. Shell tự diễn giải lệnh trong chương trình shell và thực thi chúng

10. Có bao nhiêu tập lệnh shell đi kèm với hệ điều hành UNIX?

⇨ Có khoảng 280 tập lệnh shell đi kèm với hệ điều hành UNIX.

11. Khi nào không nên sử dụng lập trình shell / scripting?

⇨ không nên sử dụng lập trình / kịch bản shell trong các trường hợp dưới đây:

- Khi nhiệm vụ rất phức tạp như viết toàn bộ hệ thống xử lý bảng lương
- Trường hợp có mức độ năng suất cao cần thiết.
- Khi nó cần hoặc liên quan đến các công cụ phần mềm khác nhau.

12. Cơ sở của chương trình shell dựa vào thực tế gì?

⇨ Cơ sở của lập trình shell dựa trên thực tế là shell UNIX có thể chấp nhận các lệnh không chỉ từ bàn phím mà còn từ một tệp.

13. Các quyền mặc định của tệp khi nó được tạo là gì?

⇨ 666 tức là rw-rw-rw- là quyền mặc định của tệp khi nó được tạo.

14) Điều gì có thể được sử dụng để sửa đổi quyền của Tệp?

=> **Quyền truy cập** tệp có thể được sửa đổi bằng cách sử dụng **umask** .

15. Làm thế nào để hoàn thành bất kỳ nhiệm vụ nào thông qua shell script?

=> Bất kỳ tác vụ nào cũng có thể được thực hiện thông qua tập lệnh shell tại dấu nhắc đô la ($) và ngược lại.

16. Biến Shell là gì?

=> Biến Shell là phần chính của lập trình shell hoặc script. Chúng chủ yếu cung cấp khả năng lưu trữ và thao tác thông tin trong một chương trình shell.

17. Hai loại biến Shell là gì? Giải thích ngắn gọn.

- Biến được xác định Unix hoặc biến hệ thống

+ Đây là các biến được xác định theo tiêu chuẩn hoặc hệ vỏ. Nói chung, chúng được định nghĩa bằng chữ VỐN.

- **Biến** do người dùng xác định

+ Chúng được xác định bởi người dùng. Nói chung, chúng được định nghĩa bằng chữ thấp hơn

18. Các biến shell được lưu trữ như thế nào? Giải thích với một ví dụ đơn giản.

=>Biến Shell được lưu trữ dưới dạng biến chuỗi.

Ví dụ: $ a = 10;

19. Tuổi thọ của một biến trong tập lệnh shell là gì?

tuổi thọ của một biến trong tập lệnh shell chỉ cho đến khi kết thúc thực thi.

20. Làm thế nào để biến các biến thành không thể thay đổi?

=>Biến có thể được thực hiện không thể thay đổi bằng cách sử dụng **chỉ đọc** . Chẳng hạn, nếu chúng ta muốn biến **một** giá trị duy trì là **10** và không bị thay đổi thì chúng ta có thể đạt được điều này bằng cách sử dụng **chỉ đọc** .

21. Làm thế nào các biến có thể bị xóa sổ?

Biến có thể bị xóa hoặc xóa bằng lệnh **unset** .

22. Tham số vị trí là gì? Giải thích với một ví dụ.

=>Tham số vị trí là các biến được xác định bởi shell. Và chúng được sử dụng bất cứ khi nào chúng ta cần truyền đạt thông tin đến chương trình. Và điều này có thể được thực hiện bằng cách chỉ định các đối số tại dòng lệnh.

23.Cái gì . (dot) cho biết ở đầu tên tệp và làm thế nào để được liệt kê?

=> Tên tệp bắt đầu bằng một. (Dấu chấm) được gọi là tệp ẩn. Bất cứ khi nào chúng tôi cố gắng liệt kê các tệp, nó sẽ liệt kê tất cả các tệp trừ tệp ẩn.

24. mỗi khối trong UNIX là bao nhiêu byte?

=> mỗi khối trong UNIX là 1024 byte.

25. Theo mặc định, một tệp mới và một thư mục mới đang được tạo sẽ có bao nhiêu liên kết?

Tập tin mới chứa **một** liên kết

26. Giải thích về quyền truy cập tệp.

=> Có 3 loại quyền truy cập tệp như dưới đây:
r - đọc, w - viết, x - thực hiện.

27. Hệ thống tập tin là gì?

- **Siêu khối** : Khối này chủ yếu nói về trạng thái của hệ thống tệp như độ lớn của nó, tối đa có thể chứa bao nhiêu tệp, v.v.
- **Khối khởi động** : Điều này thể hiện sự khởi đầu của một hệ thống tệp. Nó chứa chương trình nạp bootstrap, được thực thi khi chúng ta khởi động máy chủ.
- **Bảng Inode** : Như chúng ta đã biết tất cả các thực thể trong UNIX được coi là các tệp. Vì vậy, thông tin liên quan đến các tệp này được lưu trữ trong bảng Inode.
- **Khối dữ liệu** : Khối này chứa nội dung tệp thực tế.

29. Ba điều khoản bảo mật khác nhau được UNIX cung cấp cho một tệp hoặc dữ liệu là gì?

- Nó cung cấp một id người dùng và mật khẩu duy nhất cho người dùng, do đó người không biết hoặc không được phép không thể truy cập nó.
- Ở cấp độ tệp, nó cung cấp bảo mật bằng cách cung cấp các quyền đọc, ghi và thực thi để truy cập các tệp.
- Cuối cùng, nó cung cấp bảo mật bằng cách sử dụng mã hóa tập tin. Phương pháp này cho phép mã hóa một tập tin ở định dạng không thể đọc được. Ngay cả khi ai đó thành công trong việc mở tệp, nhưng họ không thể đọc nội dung của tệp cho đến khi và trừ khi được giải mã

30. Ba trình soạn thảo có sẵn trong hầu hết các phiên bản của UNIX là gì?

=> Ba biên tập viên là ed, ex & vi.

31. Ba chế độ hoạt động của trình soạn thảo vi là gì? Giải thích ngắn gọn.

=> Ba chế độ hoạt động của các **biên tập viên vi** là,
(i) Chế độ lệnh : Trong chế độ này, tất cả các phím được nhấn bởi người dùng được hiểu là các lệnh biên tập.

(ii) Chế độ chèn : **Chế độ** này cho phép chèn một văn bản mới và chỉnh sửa văn bản hiện có, v.v.
(iii) Chế độ lệnh cũ : **Chế độ** này cho phép người dùng nhập các lệnh tại một dòng lệnh.

32. Lệnh thay thế có sẵn để lặp lại và nó làm gì?

=> tput là một lệnh thay thế cho **echo** .
- Sử dụng điều này, chúng ta có thể kiểm soát cách hiển thị đầu ra trên màn hình.

34. Hướng dẫn điều khiển là gì và có bao nhiêu loại hướng dẫn điều khiển có sẵn trong một vỏ? Giải thích ngắn gọn.

=> Hướng dẫn điều khiển là: những **hướng dẫn** , cho phép chúng tôi chỉ định thứ tự mà các hướng dẫn khác nhau trong chương trình / tập lệnh sẽ được máy tính thực thi. Về cơ bản, họ xác định một luồng kiểm soát trong một chương trình.
Có 4 loại hướng dẫn điều khiển có sẵn trong vỏ.

- **Hướng dẫn điều khiển tuần tự** - Điều này đảm bảo rằng các hướng dẫn được thực hiện theo cùng thứ tự xuất hiện trong chương trình.
- **Hướng dẫn lựa chọn hoặc điều khiển quyết định** - Nó cho phép máy tính đưa ra quyết định về việc lệnh nào sẽ được thực hiện tiếp theo.
- **Lặp lại hoặc Hướng dẫn điều khiển vòng lặp** - Nó giúp máy tính thực hiện một nhóm các câu lệnh liên tục.
- **Hướng dẫn điều khiển trường hợp** - Điều này được sử dụng khi chúng ta cần chọn từ một số lựa chọn thay thế.

35. Vòng lặp là gì và giải thích ngắn gọn ba phương pháp vòng lặp khác nhau?

=> Vòng lặp là những cái, liên quan đến việc lặp lại một phần của chương trình / tập lệnh hoặc một số lần xác định hoặc cho đến khi một điều kiện cụ thể được thỏa mãn.
3 phương pháp lặp là:

- **Đối với** vòng lặp - Đây là vòng lặp được sử dụng phổ biến nhất. Đối với vòng lặp cho phép chỉ định danh sách các giá trị mà biến điều khiển trong vòng lặp có thể lấy. Vòng lặp sau đó được thực thi cho từng giá trị được đề cập trong danh sách.
- Vòng lặp **While** - Cái này được sử dụng trong một chương trình khi chúng ta muốn làm một cái gì đó trong một số lần cố định. Vòng lặp while được thực thi cho đến khi nó trả về giá trị 0.
- Vòng lặp **Until** - Điều này tương tự như vòng lặp while ngoại trừ vòng lặp thực thi cho đến khi điều kiện là đúng. Cho đến khi vòng lặp được thực thi ít nhất một lần cho đến khi nó trả về giá trị khác không.

36. IFS là gì?

=> IFS là viết tắt của **Dấu tách trường nội bộ** . Và nó là một trong những biến hệ thống. Theo mặc định, giá trị của nó là không gian, tab và một dòng mới.
Nó biểu thị rằng trong một dòng nơi một trường hoặc từ kết thúc và một trường khác bắt đầu.

37. Tuyên bố Break là gì và được sử dụng để làm gì?

=> Dấu ngắt là một từ khóa và được sử dụng bất cứ khi nào chúng tôi muốn nhảy ra khỏi vòng lặp ngay lập tức mà không cần chờ để quay lại lệnh điều khiển.

Khi gặp từ khóa bên trong bất kỳ vòng lặp nào trong chương trình, điều khiển sẽ tự động được chuyển đến câu lệnh đầu tiên sau một vòng lặp. Một break thường được liên kết với một if.

38. Tuyên bố Tiếp tục là gì và được sử dụng để làm gì?

=> Tiếp tục là một từ khóa và được sử dụng bất cứ khi nào chúng tôi muốn đưa điều khiển đến đầu vòng lặp, bằng cách chuyển các câu lệnh bên trong vòng lặp chưa được thực thi. Khi từ khóa tiếp tục gặp phải bên trong bất kỳ vòng lặp nào trong chương trình, điều khiển sẽ tự động chuyển đến đầu vòng lặp. Tiếp tục thường được liên kết với một if.

39. Metachar character trong shell là gì? Giải thích với một số ví dụ.

=> Siêu nhân vật là các ký tự đặc biệt trong trường chương trình hoặc dữ liệu cung cấp thông tin về các ký tự khác. Chúng cũng được gọi là, biểu thức chính quy trong một vỏ.

40. Làm thế nào để thực thi nhiều tập lệnh? Giải thích với một ví dụ.

=> Trong một trình bao, chúng ta có thể dễ dàng thực thi nhiều tập lệnh, tức là một tập lệnh có thể được gọi từ tập lệnh kia. Những gì chúng ta phải làm là, chúng ta cần đề cập đến tên của một kịch bản sẽ được gọi khi chúng ta muốn gọi nó.

41. Lệnh nào cần được sử dụng để biết hệ thống đã chạy được bao lâu?

=> lệnh **uptime** cần được sử dụng để biết hệ thống đã chạy được bao lâu.

42. Làm thế nào để tìm shell hiện tại mà bạn đang sử dụng?

Chúng tôi có thể tìm thấy trình bao hiện tại mà chúng tôi đang sử dụng với echo $ SHELL.

43. Làm thế nào để tìm tất cả các shell có sẵn trong hệ thống của bạn?

=> Chúng tôi có thể tìm thấy tất cả các shell có sẵn trong hệ thống của chúng tôi với $ cat /etc/ shell.

44. Làm thế nào để đọc đầu vào bàn phím trong tập lệnh shell?

=> Đầu vào bàn phím có thể được đọc trong các kịch bản shell như dưới đây,

45. Có bao nhiêu trường có trong tệp crontab và mỗi trường chỉ định điều gì?

=> Tệp **crontab** có sáu trường. Năm lĩnh vực đầu tiên nói với **cron** khi thực hiện lệnh: phút (0-59), giờ (0-23), ngày (1-31), tháng (1-12), và ngày trong tuần (0-6, Chủ nhật = 0).Và trường thứ sáu chứa lệnh được thực thi.

46. Hai tập tin của lệnh crontab là gì?

Hai tệp của lệnh crontab là :

- cron.allow - Nó quyết định người dùng nào cần được phép sử dụng lệnh crontab.
- cron.deny - Nó quyết định người dùng nào cần được ngăn chặn sử dụng lệnh crontab.
-

47. Lệnh nào cần được sử dụng để sao lưu?

=> Có ba lệnh khác nhau có sẵn để kiểm tra việc sử dụng đĩa.

df - Lệnh này được sử dụng để kiểm tra dung lượng đĩa trống.
du - Lệnh này được sử dụng để kiểm tra thư mục sử dụng đĩa khôn ngoan.
dfspace - Lệnh này được sử dụng để kiểm tra dung lượng đĩa trống theo MB.

49. Các lệnh giao tiếp khác nhau có sẵn trong Unix / shell là gì?

=> Về cơ bản, có 4 lệnh giao tiếp khác nhau có sẵn trong Unix / shell. Và họ là thư, tin tức, tường và motd.

50. Làm thế nào để tìm ra tổng dung lượng đĩa được sử dụng bởi một người dùng cụ thể, ví dụ tên người dùng là John?

=> Tổng dung lượng đĩa được John sử dụng có thể được tìm thấy như hình bên dưới. du –s/home/John

51. Shebang trong kịch bản shell là gì?

=> Shebang là một dấu # theo sau là một câu cảm thán tức là! Nói chung, điều này có thể được nhìn thấy ở đầu hoặc đầu của kịch bản / chương trình. Thông thường, một nhà phát triển sử dụng điều này để tránh công việc lặp đi lặp lại. Shebang chủ yếu xác định vị trí của động cơ sẽ được sử dụng để thực thi tập lệnh.
Ở đây biểu tượng '#' được gọi là hàm băm và '!' được gọi là một tiếng nổ.
Ví dụ: #! / Bin / bash

52. Lệnh được sử dụng để hiển thị các biến môi trường của shell là gì?

=> Lệnh được sử dụng để hiển thị các biến môi trường của shell là **env** hoặc **printenv**.

53. Làm thế nào để gỡ lỗi các vấn đề gặp phải trong shell script / chương trình?

=> Mặc dù nhìn chung nó phụ thuộc vào loại vấn đề gặp phải. Đưa ra dưới đây là một số phương pháp phổ biến được sử dụng để gỡ lỗi các vấn đề trong kịch bản.

- Các câu lệnh gỡ lỗi có thể được chèn vào tập lệnh shell để xuất / hiển thị thông tin giúp xác định vấn đề.
- Sử dụng bộ set -x, chúng tôi có thể cho phép gỡ lỗi trong tập lệnh.

54. Làm thế nào để biết chiều dài thay đổi?

=> Có thể kiểm tra độ dài biến: $ {#variable}

55. Sự khác biệt giữa = và == là gì?

= Cái này được sử dụng để gán giá trị cho biến.
== Điều này được sử dụng để so sánh chuỗi.

56. Làm cách nào để mở tệp chỉ đọc trong Unix / shell?

=> Tập tin chỉ đọc có thể được mở như hình dưới đây:
vi –R <File Name>

57. Làm thế nào có thể đọc nội dung của tệp trong jar mà không cần trích xuất trong tập lệnh shell?

=> Có thể đọc nội dung của tệp bên trong tệp jar mà không cần trích xuất trong tập lệnh shell như dưới đây.
tar –tvf <File Name>.tar

58. Sự khác biệt giữa các lệnh diff và cmp là gì?

=> diff - Về cơ bản, nó nói về những thay đổi cần được thực hiện để làm cho các tệp giống hệt nhau.
cmp - Về cơ bản, nó so sánh hai tệp byte theo byte và hiển thị sự không khớp đầu tiên.

59. Giải thích ngắn gọn về lệnh sed với một ví dụ.

=> sed là viết tắt của **trình chỉnh sửa luồng** . Và nó được sử dụng để chỉnh sửa một tập tin mà không cần sử dụng trình chỉnh sửa. Nó được sử dụng để chỉnh sửa một luồng nhất định, tức là một tệp hoặc đầu vào từ một đường ống.

60. Giải thích ngắn gọn về lệnh awk với một ví dụ.

=> awk là một tiện ích hoặc lệnh thao tác dữ liệu. Do đó, nó được sử dụng để thao tác dữ liệu.