

CÁC LỆNH CƠ BẢN TRONG LINUX

***Lệnh liên quan đến hệ thống**

1. **exit:** thoát khỏi cửa sổ dòng lệnh.
2. **logout:** tương tự exit.
3. **reboot:** khởi động lại hệ thống.
4. **halt:** tắt máy.
5. **startx:** khởi động chế độ xwindows từ cửa sổ terminal.
6. **mount:** gắn hệ thống tập tin từ một thiết bị lưu trữ vào cây thư mục chính.
7. **umount:** ngược với lệnh mount.

***Lệnh thao tác trên tập tin**

1. **ls:** lấy danh sách tất cả các file và thư mục trong thư mục hiện hành.
2. **pwd:** xuất đường dẫn của thư mục làm việc.
3. **cd:** thay đổi thư mục làm việc đến một thư mục mới.
4. **mkdir:** tạo thư mục mới.
5. **rmdir:** xoá thư mục rỗng.
6. **cp:** copy một hay nhiều tập tin đến thư mục mới.
7. **mv:** đổi tên hay di chuyển tập tin, thư mục.
8. **rm:** xoá tập tin.
9. **wc:** đếm số dòng, số kí tự... trong tập tin.
10. **touch:** tạo một tập tin.
11. **cat:** xem nội dung tập tin.
12. **vi:** khởi động trình soạn thảo văn bản vi.
13. **df:** kiểm tra dung lượng đĩa.
14. **du:** xem dung lượng đĩa đã dùng cho một số tập tin nhất định
15. **nano:** Khởi động trình soạn thảo văn bản nano
16. **less:** Xem nội dung tập tin theo dòng
17. **tail:** Xem nội dung tập tin (mặc định xem 10 dòng cuối, muốn xem 100 dòng cuối thì dùng lệnh sau: tail 100 tenfile)
18. **more:** Xem nội dung tập tin theo trang
19. **head:** Xem nội dung tập tin (mặc định xem 10 dòng đầu, muốn xem 100 dòng đầu thì dùng lệnh sau: head 100 tenfile)

***Lệnh khi làm việc trên terminal**

1. **clear:** xoá trắng cửa sổ dòng lệnh.
2. **date:** xem ngày, giờ hệ thống.
3. **cal:** xem lịch hệ thống.

***Lệnh quản lí hệ thống**

1. **rpm:** kiểm tra gói đã cài đặt hay chưa, hoặc cài đặt một gói, hoặc sử dụng để gỡ bỏ một gói.
2. **ps:** kiểm tra hệ thống tiến trình đang chạy.

3. kill: dừng tiến trình khi tiến trình bị treo. Chỉ có người dùng super-user mới có thể dừng tất cả các tiến trình còn người dùng bình thường chỉ có thể dừng tiến trình mà mình tạo ra.
4. top: hiển thị sự hoạt động của các tiến trình, đặc biệt là thông tin về tài nguyên hệ thống và việc sử dụng các tài nguyên đó của từng tiến trình.
5. pstree: hiển thị tất cả các tiến trình dưới dạng cây.
6. sleep: cho hệ thống ngừng hoạt động trong một khoảng thời gian.
7. useradd: tạo một người dùng mới.
8. groupadd: tạo một nhóm người dùng mới.
9. passwd: thay đổi password cho người dùng.
10. userdel: xoá người dùng đã tạo.
11. groupdel: xoá nhóm người dùng đã tạo.
12. gpasswd: thay đổi password của một nhóm người dùng.
13. su: cho phép đăng nhập với tư cách người dùng khác.
14. groups: hiển thị nhóm của user hiện tại.
15. who: cho biết ai đang đăng nhập hệ thống.
16. w: tương tự như lệnh who.
17. man: xem hướng dẫn về dòng lệnh như cú pháp, các tham số...

Để hiểu và sử dụng tốt các câu lệnh trên, chúng ta có thể sử dụng lệnh man với cú pháp: `man tên_câu_lệnh` để có được những thông tin đầy đủ về chức năng cũng như cú pháp của câu lệnh.

* Các câu lệnh kiểm tra thông tin hệ thống (system information) trong Linux

Lệnh Linux	Mô tả
<code>cat /proc/cpuinfo</code>	Kiểm tra thông tin CPU (số core)
<code>cat /proc/meminfo</code>	Kiểm tra thông tin về RAM đang sử dụng
<code>cat /proc/version</code>	Kiểm tra phiên bản của Kernel Linux
<code>cat /proc/ioports</code>	Xem thông tin port I/O
<code>cat /etc/redhat-release</code>	Kiểm tra phiên bản Centos
<code>uname -a</code>	Kiểm tra các thông tin về Kernel
<code>free -m</code>	Kiểm tra dung lượng RAM còn trống
<code>init 0</code>	Tắt máy (tương đương lệnh <code>shutdown -h now</code> hoặc <code>telinit 0</code>)
<code>df -h</code>	Hiển thị thông tin những file hệ thống, nơi file được lưu hoặc tất cả những file mặc định. Lệnh này có thể xem được dung lượng ổ cứng đã sử dụng và còn trống.
<code>du -sh</code>	Kiểm tra dung lượng thư mục hiện tại
<code>du -ah</code>	Hiển thị dung lượng của thư mục con và các file trong thư mục hiện tại
<code>du -h -max-depth=1</code>	Hiển thị dung lượng các thư mục con ở cấp 1 (ngay trong thư mục hiện tại)
<code>df</code>	Kiểm tra dung lượng đĩa cứng, các phân vùng đĩa
<code>lspci</code>	Xem thông tin mainboard
<code>hostname</code>	Xem các địa chỉ IP của máy
	Xem tên máy (hostname)

finger user@server	Thu thập thông tin chi tiết về người dùng hiện đang dùng hệ thống
arch	Kiểm tra kiến trúc của máy (architech)
cat /proc/swaps	Kiểm tra thông tin SWAP của máy (tương tự như virtual RAM của Windows)
last reboot	Xem lịch sử reboot máy

*) Các lệnh shutdown, restart... trong Linux

Lệnh Linux	Mô tả
logout	Kết thúc session (phiên làm việc) hiện tại
reboot	Khởi động lại máy
shutdown -r now	Khởi động lại máy (tương đương với lệnh reboot)
shutdown -h now	Tắt máy (ngay lập tức)
shutdown -h 9:30	Hẹn giờ tắt máy (schedule) vào lúc 9h30 (tính theo khung 24h)
shutdown -c	Hủy bỏ tất cả các lệnh tắt máy trước đó (các lệnh tắt máy theo schedule)
telinit 0	Tắt máy (tương đương lệnh shutdown -h now)
init 0	Tắt máy (tương đương lệnh shutdown -h now hoặc telinit 0)
exit	Thoát khỏi terminal
halt	Tắt máy (tương tự shutdown)
sleep	Cho hệ thống ngừng hoạt động trong một thời gian (ngủ – tương tự Windows)

Các lệnh về quản lý user trong Linux

Lệnh Linux	Mô tả
passwd	Đổi mật khẩu (standard user có thể đổi pass của họ còn user root thì thay đổi được password của mọi user)
pwck	Kiểm tra syntax và định dạng của dữ liệu user/password (/etc/passwd)
useradd	Tạo user mới, ví dụ: useradd -c "test user 1" -g group1
userdel	Xóa User
usermod	Thay đổi thông tin user (group, name...)
groupadd	Tạo một nhóm user mới
groupdel	Xóa nhóm user
groupmod	Thay đổi thông tin group, ví dụ, groupmod -n "old group name" "new name"
who /w	Hiển thị những user đang đăng nhập hệ thống
uname	Hiển thị tên của hệ thống (host)
id	Hiển thị user ID (Chi danh của user)
logname	Hiển thị tên user đang login

<code>su</code>	Cho phép đăng nhập với tên user khác (tương tự secondary logon của Windows)
<code>groups</code>	Hiển thị nhóm của user hiện tại
<code>#vi /etc/passwd</code>	Xem danh sách user
<code>#vi /etc/group</code>	Xem danh sách nhóm (group)
<code>chmod [tên file="""]/[/tên]</code>	Thay đổi quyền cho file/thư mục (chỉ user sở hữu file mới thực hiện được)
<code>chown user [tên file="""]/[/tên]</code>	Thay đổi chủ sở hữu file/thư mục
<code>chgrp group [file] [/file]</code>	Thay đổi group sở hữu file/thư mục

*Các lệnh Quản lý services và process trong Linux

Lệnh Linux	Mô tả
<code>top</code>	Lệnh top khá giống như Task Manager trong Windows. Nó đưa ra thông tin về tất cả tài nguyên hệ thống, các process đang chạy, tốc độ load trung bình... Lệnh top -d thiết lập khoảng thời gian làm mới lại hệ thống
<code>ps -u username</code>	Kiểm tra những process được thực hiện bởi một user nhất định
<code>ps -U root</code>	Kiểm tra mọi process ngoại trừ những process hệ thống
<code>ps -A</code>	Kiểm tra mọi process trong hệ thống
<code>Ss</code>	Kiểm tra socket đang kết nối
<code>ss -l</code>	Hiển thị các công đang mở
<code>w username</code>	Kiểm tra user đăng nhập, lịch sử đăng nhập, các process user đó đang chạy
<code>vmstat3</code>	Kiểm soát hành vi hệ thống, phân cứng và thông tin hệ thống trong Linux
<code>ps</code>	Hiển thị các chương trình hiện đang chạy
<code>uptime</code>	Hiển thị thời gian đã vận hành của hệ thống trong bao lâu
<code>rpm</code>	Kiểm tra, gỡ bỏ hoặc cài đặt 1 gói .rpm
<code>yum</code>	Cài đặt các ứng dụng đóng gói (giống rpm)
<code>wget</code>	Tải các ứng dụng từ một website về
<code>sh</code>	Chạy một ứng dụng có đuôi .sh
<code>Startx</code>	Khởi động chế độ xwindows từ cửa sổ terminal
<code>yum update -y</code>	Update Linux (CentOS)
<code>stop/start/restart</code>	Dừng/ khởi động/khởi động lại một service hoặc ứng dụng, ví dụ: service mysql stop hoặc /etc/init.d/mysqld start
<code>kill</code>	Dừng process (thường dừng khi process bị treo). Chỉ có super-user mới có thể dừng tất cả các process còn user khác chỉ có thể dừng process mà user đó tạo ra
<code>kill PID hoặc %job</code>	Ngừng một process bằng số PID (Process Identification Number) hoặc số công việc
<code>pstree</code>	Hiển thị tất cả các process dưới dạng cây
<code>service -status-all</code>	Kiểm tra tất cả các service và tình trạng của nó

whereis mysql	Hiển thị nơi các file dịch vụ được cài đặt
service -status-all grep abc	Xem tình trạng của process abc
kill -9 PID	Bắt buộc đóng một process ID
kill -1 PID	Bắt buộc đóng một process ID và load lại cấu hình mặc định của process đó

*) Một số lệnh hữu ích khác trong Linux

Lệnh Linux	Mô tả
clear	Xoá trắng cửa sổ dòng lệnh
hwclock	Fix lịch của BIOS
cal	Xem lịch hệ thống
date	Xem lịch ngày, giờ hệ thống
date -s "1 JAN 2018 15:29:00"	Đặt ngày giờ hệ thống theo string
date +%Y%m%d -s "20180101"	Đặt ngày hệ thống (không thay đổi giờ)
date +%T -s "00:29:00"	Đặt giờ hệ thống, không thay đổi ngày

*) Các câu lệnh xem thông tin file và thư mục

Lệnh Linux	Mô tả
ls	Lấy danh sách tất cả các file và folder trong thư mục hiện hành
ls tenthumuc	Liệt kê nội dung bên trong một thư mục
ls -l	Như trên, nhưng liệt kê cả kích thước file, ngày cập nhật..
ls -a	Liệt kê tất cả các file ẩn
pwd	Xuất đường dẫn của folder đang làm việc
cd	Thay đổi folder làm việc đến một folder mới (tương tự như trong DOS)
df	Kiểm tra disk space

*) Lệnh nén và giải nén trong Linux

Lệnh Linux	Mô tả
tar -cvf	Nén file/thư mục sang định dạng .tar
tar -xvf	Giải nén file tar
gzip	Chuyển file .tar sang .tar.gz
gunzip	Chuyển file .tar.gz về .tar
tar -xzf	Giải nén file .tar.gz, ví dụ: tar -xvf archive.tar
tar -zxvf	Giải nén file .tar.bz2
tar -jxvf	Giải nén file .tar.gz2
unzip	Giải nén file zip

*) Lệnh sao lưu và phục hồi database

Lệnh Linux

mysqldump -u root -p[dbpass] [databasename] > [database].sql
 mysqldump -u root -p[dbpass] [databasename] | gzip -9 > [backupfile].sql.gz
 mysql -u username -p[dbpass] [databasename] < [database].sql

*) Một số dạng toán tử của Shell Script

Cấu trúc toán tử if trong Shell Script

Cú pháp dạng 1: if then fi

```
if [ expression ]then  
    # Thực hiện lệnh nếu expression true  
fi
```

Cú pháp dạng 2: if else fi

```
if [ expression ]then  
    # Thực hiện nếu expression true  
else  
    # Thực hiện nếu expression false  
fi
```

Cú pháp dạng 3: if elif fi

```
if [ expression 1 ] then  
    # Thực hiện nếu expression1 true  
elif [ expression 2 ]then  
    # Thực hiện nếu expression2 true  
elif [ expression 3 ]then  
    # Thực hiện nếu expression3 true  
else  
    # Thực hiện nếu cả 3 expression false  
fi
```

Cấu trúc toán tử for trong Shell Script

Cấu trúc for dạng 1

```
for value in listdo  
    statements using $value  
done
```

list là một danh sách các giá trị, ví dụ như là tên file.

Cấu trúc for dạng 2

```
for (( expr1; expr2; expr3 ))do  
    Thực hiện lặp lại cho đến khi bằng TRUE  
done
```

*) Một số dạng toán tử của Python

Câu trúc toán tử if trong Python

if dk: statements	if dk: statements_1 else: statements_1
	if dk1: statements_1 elif dk2: statements_2

Câu trúc toán tử for và toán tử while trong Python

statement	explaination
for biến in range(value1, value2): statements	Lặp statements với số lần biết trước là: value2 – value1 – 1 Lưu ý: s += a nghĩa là s = s + a
while dk: statements	Lặp với số lần không biết trước. Còn lặp khi dk sau while còn đúng. Dừng lặp khi dk sau while sai.

Truy cập file văn bản trong Python

Qui tắc chung

	Truy cập file để ĐỌC	Truy cập file để GHI
	Giả sử tên biến file là fi s là một biến kiểu string	Giả sử tên biến file là fo s là một biến có kiểu dữ liệu nào đó
Mở file	Mở file để đọc dữ liệu từ file ra biến Cách 1: fi = open('Tên file', 'r') Cách 2: with open('Tên file', 'r') as fi	Mở file để ghi dữ liệu từ biến vào file Cách 1: fo = open('Tên file', 'wt') Cách 2: with open('Tên file', 'wt') as fo
Truy cập file	Đọc tất cả dòng của file ra biến s = fi.read() Đọc dòng hiện tại của file ra biến s = fi.readline()	Ghi dữ liệu từ biến hoặc biểu thức vào dòng hiện tại của file print(s, file = fo) print(biểu_thức_có_thể, file = fo)
Đóng file	fi.close()	fo.close()

TÀI LIỆU LẬP TRÌNH SHELL SCRIPT TRÊN LINUX

(Dành cho sinh viên ngành CNTT, Khóa ĐH10C, Đại học TN & MT HN)

CHƯƠNG 1: CÁC DẠNG BÀI TOÁN LẬP TRÌNH SHELL SCRIPT LIÊN QUAN ĐẾN TOÁN HỌC

Bài 1: Viết chương trình ngôn ngữ Shell Script trên Linux cho phép nhập số tự nhiên N bất kỳ từ bàn phím. Yêu cầu tính tổng các số từ 1 đến số N nhập vào

```
#!/bin/bash
```

```
echo "Nhập số tự nhiên N: "
read N
# tính tong tu 1 den N
sum=0
for (( i=1; i<=N; i++ ))
do
    sum=$((sum+i))
done
# in ket qua
echo "Tổng từ 1 đến $N là: $sum"
```

Bài 2: Viết chương trình ngôn ngữ Shell Script trên Linux cho phép nhập số tự nhiên N bất kỳ từ bàn phím. Yêu cầu tính tổng các số chẵn, số lẻ, số chẵn chia hết

cho 2 từ 1 đến số N nhập vào

```
#!/bin/bash
```

```
# Nhập số N từ bàn phím
echo "Nhập số tự nhiên N: "
read N
```

Khởi tạo biến tổng cho từng loại số

```
even_sum=0
```

```
odd_sum=0
```

```
divisible_sum=0
```

```

# Lặp từ 1 đến N và tính tổng cho từng loại số
for ((i=1;i<=N;i++))
do
    # Nếu số là chẵn
    if ((i%2==0))
    then
        even_sum=$((even_sum+i))
    # Nếu số chẵn chia hết cho 2
    if ((i%4==0))
    then
        divisible_sum=$((divisible_sum+i))
    fi
    # Nếu số là lẻ
    else
        odd_sum=$((odd_sum+i))
    fi
done
# In kết quả
echo "Tổng các số chẵn từ 1 đến N là: $even_sum"
echo "Tổng các số lẻ từ 1 đến N là: $odd_sum"
echo "Tổng các số chẵn chia hết cho 2 từ 1 đến N là: $divisible_sum"

```

Bài 3: Viết chương trình giải và biện luận phương trình bậc nhất $Ax + B = 0$ bằng Shell Script trên Linux

```

#!/bin/bash

echo "Nhập hệ số a: "
read a
echo "Nhập hệ số b: "
read b

if [ $a -eq 0 ]

```

```

then
if [ $b -eq 0 ]
then
    echo "Phuong trinh dung voi moi x"
else
    echo "Phuong trinh vo nghiem"
fi
exit
fi
x=$(echo "scale=2; (-1)*$b/$a" | bc)
echo "Phuong trinh $a*x + $b = 0 co nghiem x = $x"

```

Bài 4: Viết chương trình giải và biện luận phương trình bậc 2: $Ax^2 + Bx + C = 0$ bằng Shell Script trên Linux

```
#!/bin/bash
```

```
# Lấy các hệ số từ người dùng
```

```
echo "Nhập hệ số A:"
```

```
read a
```

```
echo "Nhập hệ số B:"
```

```
read b
```

```
echo "Nhập hệ số C:"
```

```
read c
```

```
# Tính delta
```

```
delta=$((b*b-4*a*c))
```

```
# Kiểm tra điều kiện và giải phương trình
```

```
if [ $delta -gt 0 ]; then
```

```
    echo "Phương trình có hai nghiệm phân biệt:"
```

```
    x1=$(echo "scale=2; (-$b+sqrt($delta))/(2*$a)" | bc)
```

```
    x2=$(echo "scale=2; (-$b-sqrt($delta))/(2*$a)" | bc)
```

```

echo "x1 = $x1"
echo "x2 = $x2"
elif [ $delta -eq 0 ]; then
    echo "Phương trình có nghiệm kép:"
    x=$(echo "scale=2; (-$b)/(2*$a)" | bc)
    echo "x = $x"
else
    echo "Phương trình vô nghiệm"
fi

```

```

# Biện luận phương trình
if [ $a -eq 0 ]; then
    echo "Đây không phải phương trình bậc 2"
else
    echo "Đây là phương trình bậc 2"
    echo "Hệ số delta của phương trình là: $delta"
    if [ $delta -gt 0 ]; then
        echo "Phương trình có hai nghiệm phân biệt"
    elif [ $delta -eq 0 ]; then
        echo "Phương trình có nghiệm kép"
    else
        echo "Phương trình vô nghiệm"
    fi
fi

```

Bài 5: Viết chương trình Shell Script trên linux, cho phép nhập vào số tự nhiên N từ bàn phím. Hãy kiểm tra số nhập vào có phải là số nguyên tố hay không?

```

#!/bin/bash
is_prime(){
n=$1
if [ $n -lt 1 ];then
    return 0

```

```

    fi
    for ((i=2;i<n;i++));do
        let "k=$n%$i"
        if [ $k -eq 0 ];then
            return 0
        fi
    done
    return 1
}

echo -n "Nhập n: "
read n
is_prime $n
if [ $? -eq 0 ];then
    echo "$n không là số nguyên tố"
else
    echo "$n là số nguyên tố"
fi
exit 0

```

Bài 6: Viết chương trình Shell Script trên linux, cho phép nhập vào số tự nhiên N từ bàn phím. Hãy liệt kê các số nguyên tố nhỏ hơn N

```

#!/bin/bash
is_prime(){
    n=$1
    if [ $n -lt 2 ];then
        return 0
    fi
    for ((i=2;i<n;i++));do
        let "k=$n%$i"
        if [ $k -eq 0 ];then
            return 0

```

```

    fi
done
return 1
}
lietke(){
n=$1
if [ $n -lt 2 ]; then
    echo "khong co so nguyen to nao"
fi
for ((j=2;j<n;j++));do
    is_prime $j
    if [ $? -eq 1 ];then
        echo "$j"
    fi
done
}
echo -n "Nhap n: "
read n
lietke $n
exit 0

```

Bài 7: Viết Shell Script trên Linux tính tổng các ký số của một số được nhập vào, ví dụ nhập số tự nhiên 1234, kết quả tính tổng các số đó bằng 10

```

#!/bin/bash

echo -n "Nhập dãy số tự nhiên vào: "
read num

sum=0
while [ $num -gt 0 ]
do
    digit=$(( $num % 10 ))

```

```

sum=$(( $sum + $digit ))
num=$(( $num / 10 ))
done
echo "Tổng là: $sum"

```

Bài 8: Viết chương trình ngôn ngữ Shell Script trên Linux cho phép tạo tam giác vuông, đều, cân nếu 3 cạnh được nhập ngẫu nhiên từ bàn phím

Gợi ý:

Để tạo tam giác vuông, đều, cân, ta sẽ sử dụng câu lệnh read để lấy giá trị độ dài 3 cạnh a, b, c từ người dùng. Sau đó, ta sẽ kiểm tra các điều kiện để xác định loại tam giác.

Nếu tam giác là tam giác vuông, ta sẽ tính toán độ dài của cạnh huyền và in kết quả.

Nếu tam giác là tam giác đều hoặc cân, ta sẽ kiểm tra các cạnh và in kết quả.

```

#!/bin/bash
# Lấy độ dài 3 cạnh từ người dùng
read -p "Nhập độ dài cạnh a: " a
read -p "Nhập độ dài cạnh b: " b
read -p "Nhập độ dài cạnh c: " c

# Kiểm tra tam giác vuông
if [[ $a -gt 0 && $b -gt 0 && $c -gt 0 ]]; then
    if [[ $(($a*$a + $b*$b)) -eq $($c*$c) || $($b*$b + $c*$c)) -eq $($a*$a) || $($a*$a + $c*$c)) -eq $($b*$b) ]]; then
        echo "Đây là tam giác vuông"
    # Tính độ dài cạnh huyền
    if [[ $a -gt $b && $a -gt $c ]]; then
        huy = $a
        cat1 = $b
        cat2 = $c
    elif [[ $b -gt $a && $b -gt $c ]]; then

```

```

huy = $b
cat1 = $a
cat2 = $c
else
    huy = $c
    cat1 = $a
    cat2 = $b
fi
echo "Độ dài cạnh huyền: $huy"
fi

```

Kiểm tra tam giác đều hoặc cân

```

if [[ $a -eq $b && $b -eq $c ]]; then
    echo "Đây là tam giác đều"
elif [[ $a -eq $b || $b -eq $c || $a -eq $c ]]; then
    echo "Đây là tam giác cân"
fi

```

Bài 9: Nhập vào 1 mảng n phần tử. Hãy viết chương trình Shell script sắp xếp mảng tăng/ giảm dần

```

#!/bin/bash
declare -a a
echo -n "Nhập n: "
read n
for ((i=1;i<=n;i++));do
    echo -n "a[$i]= "
    read m
    a[$i]=$m
done
echo -n "Mảng chưa sắp xếp: "
echo ${a[*]}

```

```

for ((i=1;i<=n-1;i++));do
    for ((j=i+1;j<=n;j++));do
        if [ ${a[$i]} -gt ${a[$j]} ];then
            tam=${a[$i]}
            a[$i]=${a[$j]}
            a[$j]=$tam
        fi
    done
done
echo ${a[*]}

```

Bài 10: Nhập vào 1 mảng n phần tử. Hãy viết chương trình Shell script tính tổng các phần tử của mảng, tổng phần tử chẵn, tổng phần tử lẻ của mảng

`#!/bin/bash`

```

# Nhập vào mảng
echo -n "Nhập số phần tử của mảng vào: "
read n

echo "Nhập phần tử của mảng:"
for (( i=0; i<n; i++ ))
do
    read arr[$i]
done

# Tính tổng các phần tử trong mảng
sum=0
for (( i=0; i<n; i++ ))
do
    sum=$(( $sum + ${arr[$i]} ))
done

```

```
# Tính tổng các phần tử chẵn trong mảng
even_sum=0
for (( i=0; i<n; i++ ))
do
    if [ ${arr[$i]} % 2 -eq 0 ]
    then
        even_sum=$((even_sum + ${arr[$i]}))
    fi
done

# Tính tổng các phần tử lẻ trong mảng
odd_sum=0
for (( i=0; i<n; i++ ))
do
    if [ ${arr[$i]} % 2 -ne 0 ]
    then
        odd_sum=$((odd_sum + ${arr[$i]}))
    fi
done

# In kết quả
echo "Tổng các phần tử của mảng: $sum"
echo "Tổng các phần tử chẵn của mảng: $even_sum"
echo "Tổng các phần tử lẻ của mảng: $odd_sum"
```

Bài 11: Nhập vào 1 mảng N phần tử. Hãy viết chương trình Shell script in ra màn hình số lớn nhất, số nhỏ nhất của mảng đã nhập vào

```
#!/bin/bash
```

```
# Nhập số phần tử của mảng
read -p "Nhập số phần tử của mảng: " n
```

```
# Khởi tạo mảng với giá trị ban đầu là 0  
arr=()
```

```
# Nhập các phần tử của mảng  
echo "Nhập các phần tử của mảng:"  
for (( i=0; i<$n; i++ ))  
do  
    read -p "Phần tử thứ ${((i+1))}: " num  
    arr+=($num)  
done
```

```
# Khởi tạo giá trị lớn nhất và nhỏ nhất ban đầu  
max=${arr[0]}  
min=${arr[0]}
```

```
# Tìm giá trị lớn nhất và nhỏ nhất trong mảng  
for i in "${arr[@]}"  
do  
    if [[ "$i" -gt "$max" ]]; then  
        max="$i"  
    fi  
  
    if [[ "$i" -lt "$min" ]]; then  
        min="$i"  
    fi  
done
```

```
# In ra giá trị lớn nhất và nhỏ nhất của mảng  
echo "Số lớn nhất của mảng là: $max"  
echo "Số nhỏ nhất của mảng là: $min"
```

Bài 12: Nhập vào 1 mảng n phần tử. Hãy viết chương trình Shell script liệt kê các số nguyên tố có trong mảng đã nhập vào

Ví dụ:

Nhập số phần tử của mảng là: 7

Phần tử của mảng: 2 7 9 13 11 4 6

Số nguyên tố có trong mảng đã nhập là: 2 7 13 11

```
#!/bin/bash
```

```
# Nhập vào mảng
```

```
echo -n "Nhập số phần tử của mảng: "
```

```
read n
```

```
echo "Phần tử của mảng:"
```

```
for (( i=0; i<n; i++ ))
```

```
do
```

```
    read arr[$i]
```

```
done
```

```
# Tìm số nguyên tố trong mảng và in ra màn hình
```

```
echo "Số nguyên tố có trong đây:"
```

```
for (( i=0; i<n; i++ ))
```

```
do
```

```
    is_prime=1 # biến cờ để kiểm tra số nguyên tố
```

```
# Kiểm tra xem phần tử hiện tại có phải là số nguyên tố hay không
```

```
for (( j=2; j<${arr[$i]}; j++ ))
```

```
do
```

```
    if [ $(( ${arr[$i]} % $j )) -eq 0 ]
```

```
    then
```

```
        is_prime=0 # phát hiện số không phải nguyên tố, đổi biến cờ  
        break
```

```
    fi
```

```
done
```

```
# In ra phần tử nếu là số nguyên tố  
if [ $is_prime -eq 1 ] && [ ${arr[$i]} -gt 1 ]  
then  
    echo ${arr[$i]}\nfi  
done
```

Bài 13: Nhập vào 1 mảng n danh sách sinh viên gồm các thông tin mã sinh viên, họ và tên, ngày sinh, nơi sinh, lớp học, môn học, điểm thi. Hãy viết chương trình Shell script hiển thị danh sách sinh viên vừa nhập ra màn hình

```
#!/bin/bash\n\n# Nhập vào mảng các thông tin của sinh viên\necho -n "Nhập số lượng sinh viên: "\nread n\n\necho "Thông tin của 1 sinh viên:"\nfor (( i=0; i<n; i++ ))\ndo\n    echo -n "Nhập mã sv: "\n    read student_id\n    echo -n "Nhập tên sinh viên: "\n    read student_name\n    echo -n "Nhập ngày tháng năm sinh (dd/mm/yyyy): "\n    read dob\n    echo -n "Nơi sinh: "\n    read pob\n    echo -n "Lớp học: "\n    read class\n    echo -n "Nhập môn học: "\n
```

```

read subject
echo -n "Môn học: "
read score

# Lưu thông tin sinh viên vào mảng students
students[$i]="$student_id $student_name $dob $pob $class $subject $score"
done

# In ra danh sách sinh viên
echo "Thông tin sinh viên:"
echo "-----"
echo "MaSV    Họ Tên      Ngày Sinh      Nơi Sinh   Lớp   Điểm Môn Học"
echo "-----"
for (( i=0; i<n; i++ ))
do
  echo "${students[$i]}"
done

```

Bài 14: Viết chương trình Shell Script cho phép nhập vào 1 chuỗi và 1 ký tự, kiểm tra ký tự có trong chuỗi hay không, nếu có đưa ra số lần xuất hiện của ký tự đó trong chuỗi

```

#!/bin/bash
echo "Nhập vào một chuỗi: "
read string

echo "Nhập vào một ký tự: "
read char

# Sử dụng lệnh awk để đếm số lần xuất hiện của ký tự trong chuỗi
count=$(echo "$string" | awk -F"$char" '{print NF-1}')

```

```
if [ $count -eq 0 ]; then
    echo "Ký tự '$char' không có trong chuỗi '$string'"
else
    echo "Ký tự '$char' xuất hiện $count lần trong chuỗi '$string'"
fi
```



CHƯƠNG 2: CÁC DẠNG BÀI TOÁN LẬP TRÌNH SHELL SCRIPT TOÁN HỌC ỨNG DỤNG

Bài 1: Hãy viết file Bash script trong Linux, sử dụng ký tự “*” in ra hình tam giác vuông với kích thước hai cạnh góc vuông bằng nhau và được nhập từ bàn phím

```
#!/bin/bash
echo "Nhập kích thước hình tam giác"
read kich_thuoc
for ((i = 0; i < $kich_thuoc; i++))
do
    # tao khoang trong
    for ((k = 0; k < $kich_thuoc - $i + 1; k++))
    do
        echo -n " "
    done
    # tao *
    for ((k = 0; k < $i + 1; k++))
    do
        echo -n "*"
    done
    echo ""
done
```

Bài 2: Viết chương trình Shell Script trong Linux đếm số ký tự trong tên của bạn (tên viết liền, không dấu), so sánh với 10 sau đó in ra kết quả lớn hơn, nhỏ hơn hoặc bằng 10.

```
#!/bin/bash
echo "Nhập tên ban viet lien";
read ten_ban;
do_dai_ten_ban=${#ten_ban}
if [ $do_dai_ten_ban -gt 10 ]
```

esac

echo "Bạn \$ten và năm sinh \$nam có mệnh là: \$can \$chi"

CHƯƠNG 3: CÁC DẠNG BÀI TẬP LIÊN QUAN ĐẾN QUẢN LÝ THƯ MỤC/TẬP TIN TRONG HỆ THỐNG

Bài 1: Viết chương trình Shell Script cho phép hệ thống xóa nhiều thư mục liên tiếp đã được tạo trong hệ thống, với số thư mục được chỉ định từ bàn phím

```
#!/bin/bash
echo "chuong trinh tao thu muc trong Linux"
echo "Nhap so thu muc can tao:"
read N
for ((i=1;$i<=$N; i++))
do
    mkdir thumuc$i
done
```

Bài 2: Hãy viết chương trình Shell Script cho phép hệ thống xóa nhiều thư mục liên tiếp đã được tạo trong hệ thống, với số thư mục được chỉ định từ bàn phím

```
#!/bin/bash
echo "chuong trinh xoa nhieu thu muc trong Linux"
echo "Nhap so thu muc can xoa:"
read N
for ((i=1;$i<=$N; i++))
do
    rm -r thumuc$i
done
```

Bài 3: Nhập vào tên bất kỳ, viết chương trình Shell script trên linux kiểm tra xem tên đó có phải là thư mục hay tập tin trong hệ thống hay không? Và thư mục/ tập tin có tồn tại trên hệ thống chưa? Nếu có thì cho phép đọc/ghi/thực thi hay không?

```
#!/bin/bash
read -p "Nhập tên file hoặc tên thư mục: " name
```

```

        ;;
    "Di chuyển")
        echo -n "Nhập đường dẫn đến nơi di chuyển: "
        read dest
        mv "$file" "$dest"
        echo "Đã di chuyển $file đến $dest"
        break
    ;;
    "Thoát")
        exit 0
    ;;
    *)
        echo "Lựa chọn không hợp lệ. Vui lòng chọn lại."
    ;;
esac
done

```

Bài 5: Nhập vào tên file bất kỳ, viết chương trình Shell script trên Linux kiểm tra xem tệp tin hoặc thư mục vừa nhập vào có tồn tại trong hệ thống hay không, nếu không thì thực hiện tạo và yêu cầu người dùng nhập chuỗi văn bản vào ghi vào tệp tin và in nội dung của tệp tin vừa ghi ra màn hình.

```

#!/bin/bash

read -p "Nhập tên thư mục: " dir_name
read -p "Nhập tên tệp tin: " file_name
# Kiểm tra xem thư mục có tồn tại hay không
if [ ! -d "$dir_name" ]; then
    # Nếu không tồn tại, tạo mới thư mục
    echo "Thư mục không tồn tại. Tạo thư mục $dir_name"
    mkdir "$dir_name"
fi
# Kiểm tra xem tệp tin có tồn tại hay không
if [ ! -f "$dir_name/$file_name" ]; then

```

Nếu không tồn tại, tạo mới tệp tin

echo "Tệp tin không tồn tại. Tạo tệp tin \$file_name trong thư mục \$dir_name"

touch "\$dir_name/\$file_name"

if
Yêu cầu người dùng nhập chuỗi văn bản và ghi vào tệp tin

read -p "Nhập nội dung tệp tin: " file_content

echo "\$file_content" > "\$dir_name/\$file_name"

In nội dung của tệp tin ra màn hình

echo "Nội dung tệp tin \$file_name:"

cat "\$dir_name/\$file_name"

Bài 6: Viết chương trình Shell Script nhập vào tên bất kỳ từ bàn phím, kiểm tra xem tên đó có phải là thư mục hay tập tin trong hệ thống hay không?

#!/bin/sh

echo "Nhập vào tên thư mục:"

read file

if [-d \$file]

then

echo "Là thư mục"

elif [-f \$file]

then

echo "Là tập tin"

else

echo "File không tồn tại"

fi

Bài 7: Viết chương trình Shell Script cho phép thực hiện copy dữ liệu từ một file sang một file bất kỳ trong hệ thống

#!/bin/bash

echo "Thực hiện copy trong linux"

read -p "nhập tên file cần copy :" file1

read -p "nhập folder chứa file :" file2

```

cp $file1 $file2 > dev/null
if [ $? -eq 0 ]
then
    echo "copy Thực hiện thành công"
else
    echo "copy không thành công"
fi

```

Bài 8: Hãy viết chương trình Shell Script trên Linux để thực hiện quét các file, tập tin trong hệ thống và đếm tổng số file, tập tin quét được trong hệ thống.

```

#!/bin/bash

dir="/"
#Biến đếm
count=0
echo "Đang thực hiện quét file..."
#Đếm tất cả các file
total=$(find "$dir" 2>/dev/null |wc -l)
echo "Tổng số file: $total"
#Lặp qua từng lệnh
find "$dir" 2>/dev/null |while read f
do
    #Đếm tăng lên khi tìm thấy tệp file
    count=$((count+1))
#Ghi tệp vào nhật ký
    echo "$f" >> /tmp/file.log
#Lặp lại phần trăm của số tệp
    echo $(( 100*$count/$total ))
#Cập nhật lại tiến trình
done|zenity --progress \
--title="Thực hiện quét File" \
--text="Ghi nhận file..." \
--percentage=0

```

Bài 9: Viết chương trình Shell Script trên Linux cho phép cho phép đếm số từ của một tập tin bất kỳ trong hệ thống.

```
#!/bin/bash
echo "Chuong trinh dem so tu cua tap tin $1"
{
    n=0
    while read line
    do
        for wd in $line
        do
            n=$((n + 1))
        done
        done
        echo "Tong so tu cua tap tin $1 la : $n"
    }<$1
exit 0
```

CHƯƠNG 4: CÁC DẠNG BÀI TẬP LẬP TRÌNH SHELL SCRIPT LIÊN QUAN ĐẾN TẠO GIAO DIỆN, HÌNH ẢNH

Bài 1: Viết chương trình Shell Script cho phép tạo giao diện đồ họa bàn cờ vua 8x8 kích thước 8x8 với hai màu trắng và đen.



```
#!/bin/bash      # Tao ban co vua 8x8
for (( i=1; i<=8; i++ ))
do
    for ((j=1 ; j<= 8; j++ ))
        do
            total=$((i+j))
            tmp=$((total%2))
            if [ $tmp -eq 0 ]
            then
                echo -e -n "\033[47m "
            else
                echo -e -n "\033[40m "
            fi
            done
            echo -e "\033[0m"
        done
done
```

Bài 2: Viết chương trình Shell Script trên Linux để hệ thống cho phép tạo giao diện giống như đang cài đặt ứng dụng.

Vui long doi trong khi dang cai dat...

30%

```
#!/bin/bash
{
    for ((i=0;i<=100;i+=5))
    do
        sleep 1
        echo $i
    done
} |
whiptail --gauge "Vui long doi trong khi dang cai dat..." 6 60 0
```

CHƯƠNG 5: CÁC DẠNG BÀI TẬP LẬP TRÌNH SHELL SCRIPT LIÊN QUAN ĐẾN QUẢN LÝ TÀI KHOẢN NGƯỜI DÙNG, NHÓM NGƯỜI DÙNG TRONG HỆ THỐNG

Bài 1: Viết chương trình Shell Script trên Linux cho phép nhập vào tên một người dùng (user) bất kỳ từ bàn phím. Yêu cầu có kiểm tra tên user vừa nhập có trong hệ thống. Nếu có tài khoản user đó thì kiểm tra xem có đang đăng nhập (login) vào hệ thống hay không

```
#!/bin/bash
echo "Nhập user cần kiểm tra:"
read tenuser
tmp=$(grep $tenuser:x /etc/passwd | wc -l)
if [ $tmp -eq 0 ]
then
    echo "User $tenuser không có trong hệ thống"
else
    echo "User $tenuser có tồn tại trong hệ thống"
    grep $tenuser:x /etc/passwd
    kt=$(who | grep $tenuser | wc -l)
    if [ $kt -ne 0 ]
    then
        echo "User $tenuser đang Login vào hệ thống"
    else
        echo "User $tenuser không Login vào hệ thống"
    fi
fi
```

Bài 2: Viết chương trình Shell Script cho phép thêm một tài khoản người dùng mới, bao gồm tên đăng nhập và mật khẩu. Có kiểm tra tài khoản vừa nhập vào có tồn tại trong hệ thống hay không? Nếu không thì cho phép nhập tên đăng nhập và mật khẩu vào hệ thống ghi nhận.

```
#!/bin/bash
if [ $(id -u) -eq 0 ]
then
```

```

read -p "Nhập tên người dùng : " ten
grep $ten /etc/passwd >/dev/null
if [ $? -eq 0 ]
then
    echo "$ten đã tồn tại rồi. Vui lòng nhập tên khác!"
    exit 1
else
    read -s -p "Nhập mật khẩu của người dùng : " matkhau
    pass=$(perl -e 'printf crypt($ARGV[0], "password")' $matkhau)
    useradd -m -p $pass $ten
    [ $? -eq 0 ] && echo "User mới đã được thêm vào hệ thống!" || echo
    "Lỗi thêm người dùng!"
fi
else
    echo "Chỉ có admin (root) mới có quyền thêm user vào hệ thống!!!"
    exit 2
fi

```

Bài 3: Viết chương trình Shell Script cho phép hệ thống thêm một nhóm tài khoản người dùng Group mới. Có kiểm tra Group nhập vào có tồn tại trong hệ thống hay không? Nếu không thì cho phép tạo group mới và thông báo đã tạo group mới thành công, đồng thời hỏi bạn có muốn tiếp tục thêm mới các tài khoản user mới vào Group vừa tạo hay không? Nếu nhấn Y thì cho phép tạo (tên tài khoản và mật khẩu), ngược lại thoát khỏi chương trình.

```

#!/bin/bash
read -p "Nhập tên nhóm cần tạo mới : " nhom
egrep $nhom /etc/group >/dev/null
if [ $? -eq 0 ]; then
    echo "$nhom đã tồn tại rồi. Vui lòng nhập tên nhóm khác!"
else
    groupadd -r $nhom
    echo "Đã thêm nhóm thành công !"

```

```
read -p "Ban co muon tiep them user moi vao group vua tao hay khong. Hay  
nhan Y de tiep tuc?" ok  
if [ $ok == 'y' ]; then  
    read -p "Nhap ten user moi:" ten  
    read -s -p "Nhap mat khau cua nguoi dung :" matkhau  
  
    pass=$(perl -e 'printf crypt($ARGV[0], "password")' $matkhau)  
    useradd -g $nhom -m -p $pass $ten  
    echo "Da them $ten thanh cong vao nhom $nhom!"  
else  
    exit 0  
fi
```

CHƯƠNG 6: CÁC DẠNG BÀI TẬP LẬP TRÌNH SHELL SCRIPT LIÊN QUAN ĐẾN QUẢN TRỊ MẠNG HỆ THỐNG

Bài 1: Viết chương trình Shell Script trên Linux để tạo cây Menu và thực hiện các chức năng dịch vụ của mạng:

- Hiển thị các thư mục hiện tại.
- Xem địa chỉ IP của máy chủ.
- Hiển thị các dịch vụ mạng của máy chủ.
- Hiển thị tất cả các tài khoản của người dùng trong hệ thống.
- Thoát khỏi chương trình.

```
#!/bin/bash
while : #Khoi dong vong lap
do
clear
#Hien thi cay thu muc cho menu
echo "--CHON LUA DANH MUC CAN THUC HIEN--"
echo "-----"
echo "1.Hien thi cac thu muc hien tai"
echo "2. Xem dia chi IP cua may chu."
echo "3. Hiển thị các dịch vụ mạng của máy chủ."
echo "4. Hiển thị tất cả các tài khoản của người dùng trong hệ thống ."
echo "5. Thoát."
1) #Hien thi tat ca tap tin va thu muc
echo "Tat ca thu muc, tap tin:"
ls -al
read -p "Nhan phim Enter de tiep tuc.." readEnterKey
;;
2) #Xem dia chi IP cua may chu
ifconfig -a
read -p "Nhan phim Enter de tiep tuc.."
```

```

readEnterKey
;;
3) #Xem cac dich vu cua mang
    service --status-all

read -p "Nhan phim Enter de tiep tuc.."
readEnterKey
;;
4) #Hien thi tai khoan nguoi dung
cat /etc/passwd
read -p "Nhan phim Enter de tiep tuc.."
readEnterKey
;;
5)
echo "Tam biet!"
exit 0
;;
*)
echo "Loi: Gia tri lua chon khong dung.."
read -p "Nhan phim [Enter] de tiep tuc..."
readEnterKey
;;
esac
done

```

Bài 2: Viết chương trình Shell Script trên Linux để tạo cây Menu và thực hiện các chức năng cấu hình của hệ thống:

- Hiển thị ngày, thời gian, tên máy chủ và người dùng đăng nhập vào hệ thống.
- Tìm kiếm tất cả các tập tin cấu hình .conf trong hệ thống.
- Liệt các dải địa chỉ IP mạng con trong máy chủ.
- Thoát khỏi chương trình

```

#!/bin/bash
while :
do
clear
echo "Ten Server - $(hostname)"
echo "-----"
echo " M A I N - M E N U"
echo "-----"
echo "1. Hien thi ngay,thoi gian va ten may chu, nguoi dang nhap vao he thong"
echo "2. Tim kiem tat ca cac tap tin cau hinh .conf trong he thong"
echo "3. Liet các dia chi IP mang con trong may chua"
echo "4. Thoat"
#Đặt các sự lựa chọn
read -p "Lua chon cac gia tri tu [ 1 -4 ] " choice
case $choice in
1)
echo "Ngay hom nay la: $(date)"
uname -a
who
read -p "Nhan phim [Enter] de tiep tuc..." 
readEnterKey
;;
2)
i=1
for file in /etc/[abcd]*.conf
do
echo "File $((i++)) : $file"
done
read -p "Nhan phim [Enter] de tiep tuc..." 
readEnterKey
;;
esac
done

```

```

3)
is_alive_ping(){
    ping -c 1 $1 > /dev/null
    [ $? -eq 0 ] && echo Dai mang con IP: $i dang chay.
}

for i in 192.168.127.{1..255}
do
    is_alive_ping $i & disown
done
read -p "Nhan phim Enter de tiep tuc.."
readEnterKey
;;
4)
echo "Tam biet!"
exit 0
;;
*)
echo "Loi: Gia tri lua chon khong dung..."
read -p "Nhan phim [Enter] de tiep tuc..."
readEnterKey
;;
esac
done

```

Bài 3: Viết chương trình Shell Script trên Linux để tạo cây Menu và thực hiện các chức năng hệ thống của máy chủ:

- Thời gian, Tên máy chủ, Người đăng nhập
- Tài khoản người dùng
- Tài khoản nhóm người dùng
- Hiện thi thông tin địa chỉ IP
- Hiện thị kết nối mạng
- Cập nhật địa chỉ IP tự động

- Hiển thị các dịch vụ của mạng
- Xem các chế độ của tường lửa
- Thoát

```
#!/bin/bash
while :
do
clear
# Hiển thị menu
echo "Ten Server - $(hostname)"
echo "-----"
echo " M A I N - M E N U"
echo "-----"
echo "1. Hien thi ngay,thoi gian va ten may chu, nguoi dang nhap vao he thong"
echo "2. Hien thi tai khoan nguoi dung."
echo "3. Hien thi tai khoan nhom nguoi dung."
echo "4. Hien thi IP va card mang."
echo "5. Hien thi ket noi mang."
echo "6. Cap phat dia chi IP Client tu dong."
echo "7. Hien thi cac dich vu cua mang."
echo "8. Xem cac che do cua tuong lua."
echo "9. Thoat"
# Đặt các sự lựa chọn
read -p "Lua chon cac gia tri tu [ 1 - 9 ] " choice
# Chọn các trường hợp
case $choice in
1)
echo "Ngay hom nay la: $(date)"
uname -a
who
read -p "Nhan phim [Enter] de tiep tuc..." 
readEnterKey
esac
done
```

```
;;
2)
cat /etc/passwd
read -p "Nhan phim [Enter] de tiep tuc..." 
readEnterKey
;;
3)
cat /etc/group
read -p "Nhan phim [Enter] de tiep tuc..." 
readEnterKey
;;
4)
ifconfig -a
read -p "Nhan phim [Enter] de tiep tuc..." 
readEnterKey
;;
5)
netstat -nat
read -p "Nhan phim [Enter] de tiep tuc..." 
readEnterKey
;;
6)
dhclient
read -p "Nhan phim [Enter] de tiep tuc..." 
readEnterKey
;;
7)
service --status-all
read -p "Nhan phim [Enter] de tiep tuc..." 
readEnterKey
;;
8)
```

```

iptables -vL -t filter
iptables -vL -t nat
iptables -vL -t mangle
iptables -vL -t raw
iptables -vL -t security
read -p "Nhan phim [Enter]"
de tiep tuc..."
readEnterKey
;;
9)
echo "Tam biet!"
exit 0
;;
*)
echo "Loi: Gia tri lua chon khong dung..."
read -p "Nhan phim [Enter] de tiep tuc..."
readEnterKey
;;
esac
done

```

Bài 4: Hãy viết chương trình Shell Script trên Linux để tạo cây Menu và thực hiện các chức năng ứng dụng của hệ thống:

- Thêm tài khoản người dùng vào hệ thống (có kiểm tra)
- Liệt kê tất cả các tiến trình đang hoạt động
- Liệt kê những tiến trình nào đang bị ngưng hoạt động
- Cho phép cài đặt một ứng dụng cụ thể của chương trình
- Thoát

```

#!/bin/bash
temp="Ban lua chon:"

```

```

select ITEM in "Them tai khoan User" "Danh sach cac tien trinh" "Cac tien trinh
ngung" "Cap nhat chuong trinh" "Thoat"
do
if [[ $REPLY -eq 1 ]]
then
read -p "Nhap username: " username
output=$(grep -w $username /etc/passwd)
if [[ -n "$output" ]]
then
echo "Ten username $username nay da ton tai roi!"
else
sudo useradd -m -d /bin/bash "$username"
if [[ $? -eq 0 ]]
then
echo "Ten username $username da duoc them thanh cong!"
tail -n 1 /etc/passwd
else
echo "Them bi loi $username"
fi
fi
elif [[ $REPLY -eq 2 ]]
then
echo "Tat ca tien trinh...."
sleep 1
ps -ef
elif [[ $REPLY -eq 3 ]]
then
read -p "Nhap tien trinh bi ngung hoat dong:" progress
pkill $progress
elif [[ $REPLY -eq 4 ]]
then
read -p "Nhap chuong trinh de cai dat: " app

```

```
sudo apt update && sudo apt install $app  
elif [[ $REPLY -eq 5 ]]  
then  
    echo "Thoat..."  
    sleep 1  
    exit  
else  
    echo "Lua chon gia tri khong hop le!!!"  
fi  
done
```

CHƯƠNG 7: CÁC DẠNG BÀI TẬP LẬP TRÌNH LẬP TRÌNH SHELL SCRIPT VÀ PYTHON TRÊN LINUX VỀ ỨNG DỤNG MẠNG

Bài 1: Hãy viết chương trình bằng ngôn ngữ Python trên Linux để thực hiện quét các cổng (port) mạng của một địa chỉ IP hoặc trang website bất kỳ khi tham gia mạng internet

CHƯƠNG TRÌNH DÒ QUÉT CÁC CỔNG (PORT) CỦA WEB
Nhập địa chỉ IP vào để quét: www.hunre.edu.vn

```
import socket
import subprocess
import sys
subprocess.call('clear', shell=True)
print (" CHƯƠNG TRÌNH DÒ QUÉT CÁC CỔNG (PORT) CỦA WEB")
# Nhập IP vào để quét
temp = input("Nhập địa chỉ IP vào để quét: ")
IP = socket.gethostname(temp)
# In thông báo chương trình đang quét
print ("Vui lòng đợi, đang thực hiện quét địa chỉ IP vừa nhập", IP)
# Thực hiện quét các cổng đang mở
try:
    for port in range(1,1025):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        result = sock.connect_ex((IP, port))
        if result == 0:
            print ("Cổng {}: đang mở".format(port))
        sock.close()
# Bắt lỗi
except KeyboardInterrupt:
    print ("Nhấn phím Ctrl+C để thoát")
    sys.exit()
except socket.error:
```

```
print ("Không kết nối được đến máy chủ")
sys.exit()
```

Bài 2: Viết chương trình Shell Script thực hiện quét các cổng port của máy chủ đang hoạt động và cho biết những port nào đang mở

```
[root@kali ~] - [~/home/hacker/zphisher]
bash bl.sh
Nhập địa chỉ IP của máy chủ: 192.168.222.1
Các cổng mở trên máy chủ 192.168.222.1 là:
7680/tcp open pando-pub
```

```
#!/bin/bash
```

```
# Nhập địa chỉ IP của máy chủ cần quét
```

```
read -p "Nhập địa chỉ IP của máy chủ: " ip_address
```

```
# Sử dụng lệnh nmap để quét các cổng và lưu kết quả vào biến output
```

```
output=$(nmap -p- --min-rate=1000 -T4 $ip_address | grep 'open')
```

```
# Hiển thị kết quả các cổng mở
```

```
echo "Các cổng mở trên máy chủ $ip_address là:"
```

```
echo "$output"
```

Bài 3: Viết chương trình Python trên môi trường Linux cho phép hiển thị IP, Gateway, Host của server Linux.

```
#!/usr/bin/python3
```

```
import socket
```

```
import os
```

```
gw = os.popen("ip -4 route show default").read().split()
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

```
s.connect((gw[2], 0))
```

```
ipaddr = s.getsockname()[0]
```

```
gateway = gw[2]
```

```
host = socket.gethostname()
```

```
print ("IP:", ipaddr, "GW:", gateway, "Host:", host)
```

Bài 4: Viết chương trình Python trên môi trường Linux cho phép tạo mã QRCode để ứng dụng một ứng dụng cụ thể:



```
import qrcode  
data='dữ liệu nhập vào/ text/ url...'  
qr=qrcode.QRCode(  
    version=1,  
    box_size=15,  
    border=5  
)  
qr.add_data(data)  
qr.make(fit=True)  
img=qr.make_image(fill='black',back_color='white')  
img.save('tenfile.png')
```

Bài 5: Ứng dụng hệ thống quản lý mã nguồn, nhập địa chỉ IP bắt đầu, địa chỉ IP cuối của cùng dải mạng và cổng cần quét từ bàn phím. Hãy viết chương trình Python cho phép hiển thị những địa chỉ IP đang tham gia hoạt động trong dải mạng và cho biết cổng (port) đang đóng hay mở

```
#!/bin/bash  
cho "Nhập địa chỉ IP bắt đầu:"  
read ip1  
echo "Nhập địa chỉ IP cuối:"  
read ip2  
echo "Nhập số cổng cần quét:"  
read port
```

```
nmap -sT $ip1- $ip2 -p $port -oG web
```

```
cat web | grep open >web1
```

```
cat web1 | cut -f2 -d ":" | cut -f1 -d "(" >web2
```

```
cat web2
```

Bài 6: Viết chương trình Shell Script thực hiện quét lỗ hổng của website đang tham gia mạng internet

Chú ý: công cụ nikto. nikto là một công cụ mã nguồn mở và miễn phí, được sử dụng để quét lỗ hổng bảo mật trên các website.

```
#!/bin/bash
```

```
# Nhập địa chỉ website cần quét
```

```
read -p "Nhập địa chỉ website cần quét: " website
```

```
# Sử dụng công cụ nikto để quét lỗ hổng trên website
```

```
output=$(nikto -h $website)
```

```
# Hiển thị kết quả quét
```

```
echo "Kết quả quét lỗ hổng trên website $website là:"
```

```
echo "$output"
```

Bài 7: Viết chương trình Shell Script thực hiện quét lỗ hổng SQL Injection của website đang tham gia mạng internet

Chú ý: công cụ sqlmap. sqlmap là một công cụ mã nguồn mở và miễn phí, được sử dụng để phát hiện và khai thác lỗ hổng SQL Injection trên các website.

```
#!/bin/bash
```

```
# Nhập địa chỉ website cần quét
```

```
read -p "Nhập địa chỉ website cần quét: " website
```

```
# Sử dụng công cụ sqlmap để quét lỗ hổng SQL Injection trên website
```

```
output=$(sqlmap -u $website --batch --level=5 --risk=3)
```

```
# Hiển thị kết quả quét
```

```
echo "Kết quả quét lỗ hổng SQL Injection trên website $website là:"
```

```
echo "$output"
```

Bài 8: Viết chương trình Python trên Linux thực hiện giao tiếp giữa mô hình Client/ Server – Máy khách máy chủ trong mạng máy tính
➔**Tạo Máy chủ (server) - file server.py**

```
import socket  
  
# Tạo socket object  
  
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
  
# Lắng nghe kết nối tại địa chỉ và cổng được chỉ định  
server_address = ('localhost', 5000)  
  
print('starting up on %s port %s' % server_address)  
  
server_socket.bind(server_address)  
  
server_socket.listen(1)  
  
# Chờ kết nối từ một client  
  
print('waiting for a connection')  
  
client_socket, client_address = server_socket.accept()  
  
# Nhận dữ liệu từ client và gửi lại  
data = client_socket.recv(1024)  
  
print('received "%s"' % data.decode('utf-8'))  
  
client_socket.sendall(b'received: '+ data)  
  
# Đóng kết nối  
  
client_socket.close()  
  
server_socket.close()
```

➔**Tạo Máy khách (client) - file client.py**

```
import socket  
  
# Tạo socket object  
  
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
  
Kết nối đến server  
server_address = ('localhost', 5000)
```

```
print('connecting to %s port %s' % server_address)
client_socket.connect(server_address)
# Gửi dữ liệu đến server
message = 'Xin chào máy chủ. Máy khách đang giao tiếp...'
client_socket.sendall(message.encode('utf-8'))
# Nhận dữ liệu từ server và in ra màn hình
data = client_socket.recv(1024)
print('received "%s"' % data.decode('utf-8'))
# Đóng kết nối
client_socket.close()
```

CHƯƠNG 8: CÁC DẠNG BÀI TẬP LẬP TRÌNH SHELL SCRIPT VÀ LẬP TRÌNH PYTHON LIÊN QUAN ĐẾN ỨNG DỤNG

Bài 1: Dựa vào một số trang website dự báo thời tiết đã biết hoặc có thể lấy thông tin từ trang website sau: <https://wttr.in/?format=j1>

Hãy viết chương trình bằng Shell Script cho phép hiển thị thời tiết như: Nhiệt độ, độ ẩm của trang website trên.

```
#!/bin/bash
url="http://wttr.in/?format=j1"
json=$(wget -qO- "$url")
#Thu vien mau
default=`echo -en "\e[39m"`
red=`echo -en "\e[31m"`
orange=`echo -en "\e[33m"`
blue=`echo -en "\e[34m"`
bold=`echo -en "\e[1m"`
normal=`echo -en "\e[0m"`
temp=$(echo $json|jq -r ."current_condition[0]".temp_F")
humidity=$(echo $json|jq -r ."current_condition[0].humidity")
description=$(echo $json|jq -r ."current_condition[0]".weatherDesc[0].value)
echo "Thoi tiet hien tai ${bold}${description}${normal}"
echo "${color}Nhiет độ: ${temp}°F ${default}"
echo "Độ ẩm: ${humidity}%"
```

CHƯƠNG 9: CÁC DẠNG BÀI TẬP LẬP TRÌNH SHELL SCRIPT VÀ LẬP MÃY

Bài 1: Hãy viết chương trình Shell Script mô tả điện toán đám mây chia sẻ như: tình trạng máy chủ máy chủ, tốc độ CPU, RAM, ổ đĩa cứng.

```
#!/bin/bash
server_name="Web Server"
cpu=$(awk -v min=50 -v max=200 'BEGIN{ srand(); print int(min+rand()*(max-min+1)) }')
memory=$(awk -v min=1 -v max=8 'BEGIN{ srand(); print int(min+rand()*(max-min+1)) }')
echo "Starting $server_name with CPU usage of $cpu% and $memory GB of memory..."
sleep 2
echo "$server_name đang chạy."
echo "Kiểm tra tình trạng của máy chủ..."
sleep 2
echo "$server_name sử dụng hiện tại $(top -b -n 1 | grep %Cpu | awk '{print $2}')% CPU and $(free -g | grep Mem | awk '{print $3}') GB of memory."
echo "Stopping $server_name..."
sleep 2
echo "$server_name đã tạm ngưng."
```

Bài 2: Hãy viết chương trình bằng Python trên Linux mô tả các liên kết về điện toán đám mây như: khái niệm, các dịch vụ chia sẻ, một số nhà cung cấp điện toán đám mây hàng đầu thế giới.

```
# Định nghĩa khái niệm chính về điện toán đám mây
```

```
class CloudComputing:
    def __init__(self):
        self.services = []
        self.providers = []
    def add_service(self, service):
```

```

        self.services.append(service)
    def add_provider(self, provider):
        self.providers.append(provider)
    def describe(self):
        print("Điện toán đám mây là một mô hình cung cấp các dịch vụ công nghệ thông tin nơi các tài nguyên được cung cấp qua internet.")
        print("Các dịch vụ có sẵn trong điện toán đám mây bao gồm:", self.services)
        print("Điện toán đám mây được cung cấp bởi các nhà cung cấp như:", self.providers)
    # Xác định một số dịch vụ và nhà cung cấp phổ biến
    services = ["Khả năng tính toán", "Khả năng lưu trữ", "Khả năng truy xuất cơ sở dữ liệu", "Khả năng kết nối mạng", "Phần mềm"]
    providers = ["Amazon Web Services (AWS)", "Microsoft Azure", "Google Cloud Platform (GCP)", "IBM Cloud", "Oracle Cloud"]
    # Tạo một đối tượng Điện toán đám mây và thêm các dịch vụ cũng như nhà cung cấp
    cloud_computing = CloudComputing()
    for service in services:
        cloud_computing.add_service(service)
    for provider in providers:
        cloud_computing.add_provider(provider)
    # Gọi phương thức mô tả để in thông tin về điện toán đám mây
    cloud_computing.describe()

```

CHƯƠNG 10: CÁC DẠNG BÀI TẬP LẬP TRÌNH GAME BẰNG SHELL SCRIPT TRÊN LINUX

Bài 1: Viết chương trình tạo game trò chơi đoán số ngẫu nhiên trong khoảng từ 1-100 bằng ngôn ngữ Shell Script trên Linux

Chú ý: Để tạo một trò chơi đơn giản bằng ngôn ngữ Shell Script trên Linux, ta có thể sử dụng các lệnh thông thường của Shell Script để xây dựng các trò chơi như đồ vui, đoán số, ... Trong ví dụ này, ta sẽ tạo một trò chơi đoán số đơn giản, trong đó người chơi cần đoán được một số nguyên trong khoảng từ 1 đến 100.

```
#!/bin/bash
# Sinh số ngẫu nhiên trong khoảng từ 1 đến 100
number=$((RANDOM % 100 + 1))

# Khởi tạo biến đếm số lần đoán
count=0
echo "Chào mừng bạn đến với trò chơi đoán số!"
echo "Bạn cần đoán số nguyên trong khoảng từ 1 đến 100."
echo "Hãy bắt đầu nào!"

# Vòng lặp chính của trò chơi
while true
do
    read -p "Hãy đoán số: " guess
    # Kiểm tra số đoán của người chơi
    if [[ "$guess" -eq "$number" ]]; then
        echo "Chúc mừng! Bạn đã đoán đúng số trong $count lần đoán."
        exit
    elif [[ "$guess" -gt "$number" ]]; then
        echo "Số bạn đoán lớn hơn số cần tìm."
    else
        echo "Số bạn đoán nhỏ hơn số cần tìm."
    fi
    # Tăng biến đếm số lần đoán
    count=$((count + 1))
done
```

```
count=$((count+1))  
done
```

Bài 2: Viết chương trình Shell Script trên Linux tạo game trò chơi so sánh giữa người chơi và máy tính bằng cách lựa chọn từ ngẫu nhiên, trả ra kết quả thắng hoặc thua nếu chọn

```
#!/bin/bash  
  
# Khai báo biến  
computer_choice=$((RANDOM % 3))  
options=("cntr" "hunre" "hack")  
  
# Tạo hàm để hiển thị kết quả  
show_result() {  
    echo "Máy tính chọn: ${options[$computer_choice]}"  
    echo "Người chơi chọn: $player_choice"  
    if [ "$player_choice" == "${options[$computer_choice]}" ]; then  
        echo "Kết quả: Hòa nhau."  
    elif [ "$player_choice" == "cntr" ] && [ "${options[$computer_choice]}" ==  
"hunre" ]; then  
        echo "Kết quả: Người chơi thắng."  
    elif [ "$player_choice" == "hunre" ] && [ "${options[$computer_choice]}" ==  
"cntr" ]; then  
        echo "Kết quả: Người chơi thua."  
    elif [ "$player_choice" == "hack" ] && [ "${options[$computer_choice]}" ==  
"hunre" ]; then  
        echo "Kết quả: Người chơi thua."  
    else  
        echo "Kết quả: Máy tính thắng."  
    fi  
}
```

```
# Hiển thị các lựa chọn và yêu cầu người chơi chọn
echo "Lựa chọn của bạn:"
echo "1. Cntt"
echo "2. Hunre"
echo "3. Hack"
read -p "Hãy chọn một số từ 1-3: " choice
```

```
# Kiểm tra lựa chọn của người chơi
```

```
case $choice in
    1) player_choice="cntt";;
    2) player_choice="hunre";;
    3) player_choice="hack";;
    *) echo "Lựa chọn không hợp lệ"; exit 1;;
esac
```

```
# Hiển thị kết quả và kết thúc chương trình
```

```
show_result
```

```
exit 0
```