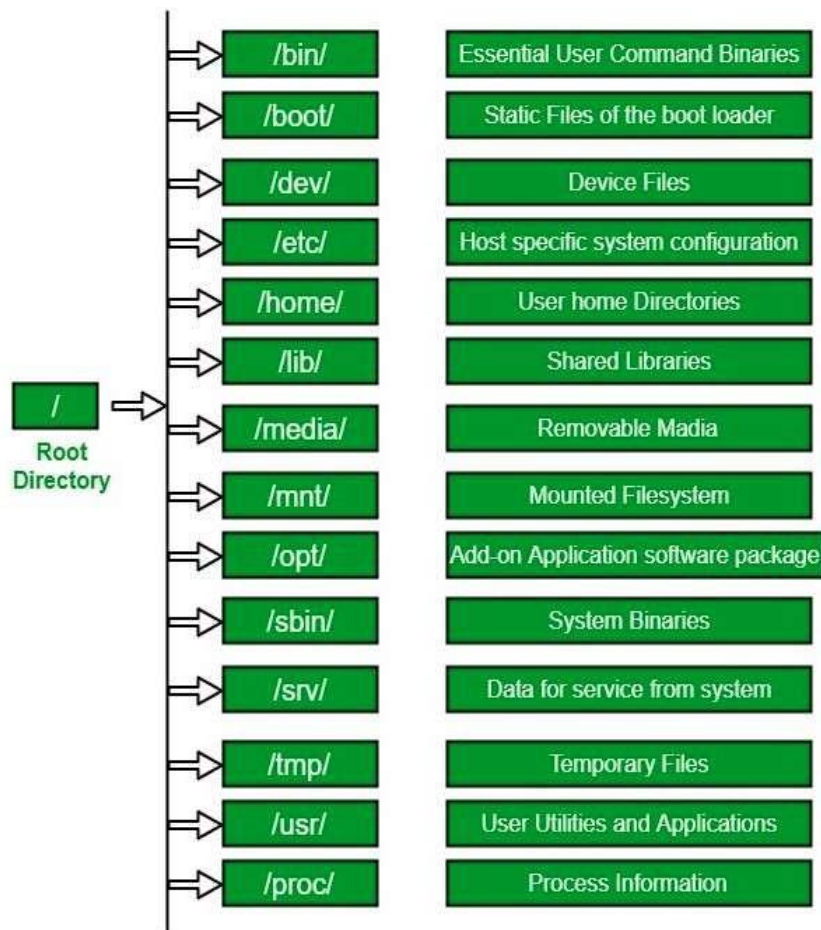


NETWORKING AND SYSTEM
ADMINISTRATION LAB RECORD

1. Write short note on linux file system hierarchy

The Linux File Hierarchy Structure or the Filesystem Hierarchy Standard (FHS) defines the directory structure and directory contents in Unix-like operating systems. It is maintained by the Linux Foundation.

- In the FHS, all files and directories appear under the root directory /, even if they are stored on different physical or virtual devices.
- Some of these directories only exist on a particular system if certain subsystems, such as the X Window System, are installed.
- Most of these directories exist in all UNIX operating systems and are generally used in much the same way; however, the descriptions here are those used specifically for the FHS, and are not considered authoritative for platforms other than Linux.



The Root Directory

Everything on your Linux system is located under the / directory, known as the root directory. You can think of the / directory as being similar to the C:\ directory on Windows – but this isn't strictly true, as Linux doesn't have drive letters. While another partition would be located at D:\ on Windows, this other partition would appear in another folder under / on Linux.

/bin : Essential command binaries that need to be available in single user mode; for all users, e.g., cat, ls, cp

- Contains binary executables
- Common linux commands you need to use in single-user modes are located under this directory.
- Commands used by all the users of the system are located here e.g. ps, ls, ping, grep, cp

The /bin directory contains the essential user binaries (programs) that must be present when the system is mounted in single-user mode. Applications such as Firefox are stored in /usr/bin, while important system programs and utilities such as the bash shell are located in /bin. The /usr directory may be stored on another partition – placing these files in the /bin directory ensures the system will have these important utilities even if no other file systems are mounted. The /sbin directory is similar – it contains essential system administration binaries.

/boot – Static Boot Files

The /boot directory contains the files needed to boot the system – for example, the GRUB boot loader's files and your Linux kernels are stored here. The boot loader's configuration files aren't located here, though – they're in /etc with the other configuration files.

/cdrom – Historical Mount Point for CD-ROMs

The /cdrom directory isn't part of the FHS standard, but you'll still find it on Ubuntu and other operating systems. It's a temporary location for CD-ROMs inserted in the system. However, the standard location for temporary media is inside the /media directory

/dev – Device Files

Linux exposes devices as files, and the /dev directory contains a number of special files that represent devices. These are not actual files as we know them, but they appear as files

– for example, `/dev/sda` represents the first SATA drive in the system. If you wanted to partition it, you could start a partition editor and tell it to edit `/dev/sda`.

This directory also contains pseudo-devices, which are virtual devices that don't actually correspond to hardware. For example, `/dev/random` produces random numbers. `/dev/null` is a special device that produces no output and automatically discards all input – when you pipe the output of a command to `/dev/null`, you discard it.

`/etc` – Configuration Files

The `/etc` directory contains configuration files, which can generally be edited by hand in a text editor. Note that the `/etc/` directory contains system-wide configuration files – userspecific configuration files are located in each user's home directory.

`/home` – Home Folders

The `/home` directory contains a home folder for each user. For example, if your user name is bob, you have a home folder located at `/home/bob`. This home folder contains the user's data files and user-specific configuration files. Each user only has write access to their own home folder and must obtain elevated permissions (become the root user) to modify other files on the system.

`/lib` – Essential Shared Libraries

The `/lib` directory contains libraries needed by the essential binaries in the `/bin` and `/sbin` folder. Libraries needed by the binaries in the `/usr/bin` folder are located in `/usr/lib`.

`/lost+found` – Recovered Files

Each Linux file system has a `lost+found` directory. If the file system crashes, a file system check will be performed at next boot. Any corrupted files found will be placed in the `lost+found` directory, so you can attempt to recover as much data as possible.

`/media` – Removable Media

The `/media` directory contains subdirectories where removable media devices inserted into the computer are mounted. For example, when you insert a CD into your Linux system, a directory will automatically be created inside the `/media` directory. You can access the contents of the CD inside this directory.

/mnt – Temporary Mount Points

Historically speaking, the /mnt directory is where system administrators mounted temporary file systems while using them. For example, if you're mounting a Windows partition to perform some file recovery operations, you might mount it at /mnt/windows. However, you can mount other file systems anywhere on the system.

/opt – Optional Packages

The /opt directory contains subdirectories for optional software packages. It's commonly used by proprietary software that doesn't obey the standard file system hierarchy – for example, a proprietary program might dump its files in /opt/application when you install it.

/proc – Kernel & Process Files

The /proc directory is similar to the /dev directory because it doesn't contain standard files. It contains special files that represent system and process information.

/root – Root Home Directory

The /root directory is the home directory of the root user. Instead of being located at /home/root, it's located at /root. This is distinct from /, which is the system root directory.

/run – Application State Files

The /run directory is fairly new, and gives applications a standard place to store transient files they require like sockets and process IDs. These files can't be stored in /tmp because files in /tmp may be deleted.

/sbin – System Administration Binaries

The /sbin directory is similar to the /bin directory. It contains essential binaries that are generally intended to be run by the root user for system administration.

/selinux – SELinux Virtual File System

If your Linux distribution uses SELinux for security (Fedora and Red Hat, for example), the /selinux directory contains special files used by SELinux. It's similar to /proc. Ubuntu doesn't use SELinux, so the presence of this folder on Ubuntu appears to be a bug.

/srv – Service Data

The /srv directory contains “data for services provided by the system.” If you were using the Apache HTTP server to serve a website, you’d likely store your website’s files in a directory inside the /srv directory.

/tmp – Temporary Files

Applications store temporary files in the /tmp directory. These files are generally deleted whenever your system is restarted and may be deleted at any time by utilities such as tmpwatch.

/usr – User Binaries & Read-Only Data

The /usr directory contains applications and files used by users, as opposed to applications and files used by the system. For example, non-essential applications are located inside the /usr/bin directory instead of the /bin directory and non-essential system administration binaries are located in the /usr/sbin directory instead of the /sbin directory. Libraries for each are located inside the /usr/lib directory. The /usr directory also contains other directories – for example, architecture-independent files like graphics are located in /usr/share. The /usr/local directory is where locally compiled applications install to by default – this prevents them from mucking up the rest of the system.

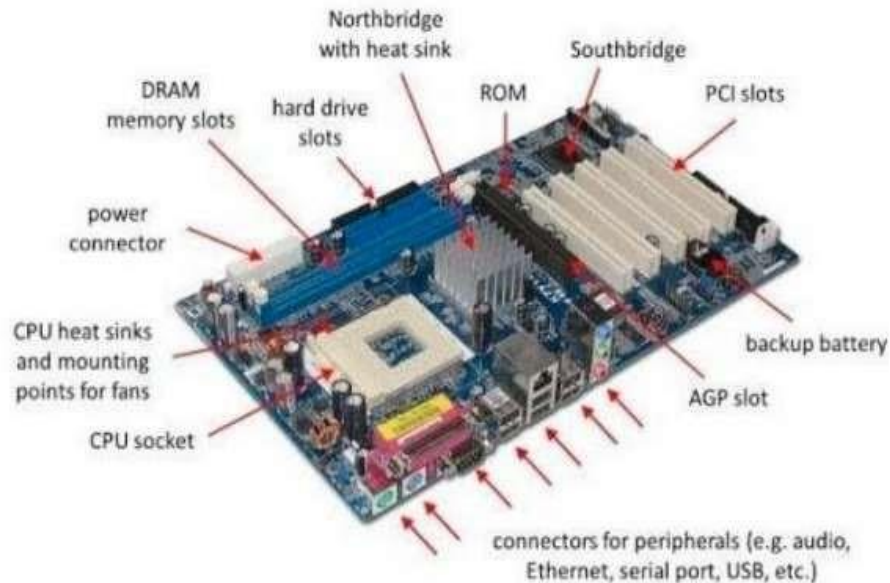
/var – Variable Data Files

The /var directory is the writable counterpart to the /usr directory, which must be read-only in normal operation. Log files and everything else that would normally be written to /usr during normal operation are written to the /var directory. For example, you’ll find log files in /var/log.

Write short note on motherboard, Ram, Daughter cards, bus slotted, SMPS, internal storage devices, interfacing port

Physical identification of major components of a computer system such as motherboard, RAM modules, daughter cards, bus slots, SMPS, internal storage devices, interfacing ports. Specifications of desktop and server class computers. Installation of common operating systems for desktop and server use. (Students may be asked to formulate specification for computer to be used as Desktop, Web server)

MOTHERBOARD



A motherboard (also called mainboard, main circuit board, or mobo) is the main printed circuit board (PCB) in general-purpose computers and other expandable systems. It holds and allows communication between many of the crucial electronic components of a system, such as the central processing unit (CPU) and memory, and provides connectors for other peripherals. Unlike a backplane, a motherboard usually contains significant sub-systems, such as the central processor, the chipset's input/output and memory controllers, interface connectors, and other components integrated for general use. Motherboard means specifically a PCB with expansion capabilities. As the name suggests, this board is often referred to as the "mother" of all components attached to it, which often include peripherals, interface cards, and daughterboards: sound cards, video cards, network cards, host bus adapters, TV tuner cards, IEEE 1394 cards; and a variety of other custom components.

RAM modules



In computing, a memory module or RAM (random-access memory) stick is a printed circuit board on which memory integrated circuits are mounted. Memory modules permit easy installation and replacement in electronic systems, especially computers such as personal computers, workstations, and servers. The first memory modules were proprietary designs that were specific to a model of computer from a specific manufacturer. Later, memory modules were standardized by organizations such as JEDEC and could be used in any system designed to use them.

Types of memory module include:

- TransFlash Memory Module
- SIMM, a single in-line memory module
- DIMM, dual in-line memory module
- Rambus memory modules are a subset of DIMMs, but are normally referred to as RIMMs
- SO-DIMM, small outline DIMM, a smaller version of the DIMM, used in laptops

Distinguishing characteristics of computer memory modules include voltage, capacity, speed (i.e., bit rate), and form factor. For economic reasons, the large (main) memories found in personal computers, workstations, and non-handheld game-consoles (such as PlayStation and Xbox) normally consist of dynamic RAM (DRAM). Other parts of the computer, such as cache memories normally use static RAM (SRAM). Small amounts of SRAM are sometimes used in the same package as DRAM.[2] However, since SRAM has high leakage power and low density, die-stacked DRAM has recently been used for designing multi-megabyte sized processor caches.

DAUGHTER BOARD



The daughter board is a computer hardware. It is also known as the piggyback board, riser card, daughter board, daughtercard or daughter card. A daughter board is a printed circuit board which is connected to the motherboard or expansion card. As compared to the motherboard, it is smaller in size. A daughter board does not act as an expansion card. An expansion card adds extra new functions to the computer. But a daughter board that is connected to the motherboard adds or supports the main functions of the motherboard. Daughter boards are directly connected to the motherboards. You know that expansion cards are connected to the motherboard by using the bus and other serial interfaces. But daughter board is directly connected to the board by soldering. As an update of the motherboard or expansion card, daughter boards are released to extend the features and services of the motherboard or expansion cards.

BUS SLOTS



Alternatively known as a bus slot or expansion port, an expansion slot is a connection or port inside a computer on the motherboard or riser card. It provides an installation point for a hardware expansion card to be connected, which provides additional features to a computer such as video, sound, advanced graphics, Ethernet or memory. The expansion card has an edge connector that fits precisely into the expansion slot as well as a row of contacts that is designed to establish an electrical connection between the motherboard and the electronics on the card, which are mostly integrated circuits. Depending on the form factor of the case and motherboard, a computer system generally can have anywhere from one to seven expansion slots. With a backplane system, up to 19 expansion cards can be installed. Expansion cards can provide various functions including:

SMPS



A switched-mode power supply (SMPS) is an electronic circuit that converts power using switching devices that are turned on and off at high frequencies, and storage components such as inductors or capacitors to supply power when the switching device is in its non-conduction state.

- Solid-state drive
- Power-on self-test
- Advanced multirate codec
- Basic input/output system (BIOS)
- Expansion read-only memory (ROM)
- Security devices
- RAM memory
- Sound
- Modems
- Network
- Interface adapters
- TV and radio tuning

- Video processing
 - Host adapting such as redundant array of independent disks or small computer system interface
- Switching power supplies have high efficiency and are widely used in a variety of electronic equipment, including computers and other sensitive equipment requiring stable and efficient power supply. A switched-mode power supply is also known as a switch-mode power supply or switching-mode power supply. Switched-mode power supplies are classified according to the type of input and output voltages.

The four major categories are:

- AC to DC
- DC to DC
- DC to AC
- AC to AC

A basic isolated AC to DC switched-mode power supply consists of:

- Input rectifier and filter
- Inverter consisting of switching devices such as MOSFETs
- Transformer
- Output rectifier and filter
- Feedback and control circuit

INTERNAL STORAGE DEVICES



A storage unit is a part of the computer system which is employed to store the information and instructions to be processed. A storage device is an integral part of the computer hardware which stores information/data to process the result of any computational work. Without a storage device, a computer would not be able to run or even boot up. Or in other words, we can say that a storage device is hardware that is used for storing, porting, or extracting data files. It can also store information/data both temporarily and permanently. Computer storage is of two types:

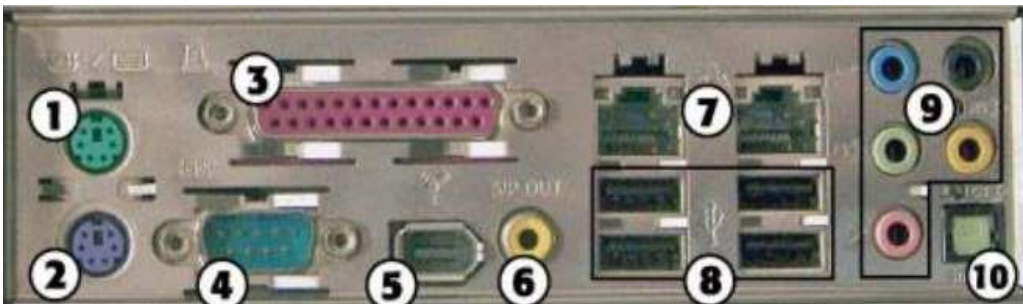
Primary Storage Devices: It is also known as internal memory and main memory. This is a section of the CPU that holds program instructions, input data, and intermediate results.

It is generally smaller in size. RAM (Random Access Memory) and ROM (Read Only Memory) are examples of primary storage.

Secondary Storage Devices: Secondary storage is a memory that is stored external to the computer. It is mainly used for the permanent and long-term storage of programs and data.

Hard Disk, CD, DVD, Pen/Flash drive, SSD, etc, are examples of secondary storage.

INTERFACING PORTS



A port is basically a physical docking point which is basically used to connect the external devices to the computer, or we can say that A port act as an interface between the computer and the external devices, e.g., we can connect hard drives, printers to the computer with the help of ports.

Characteristics of Ports

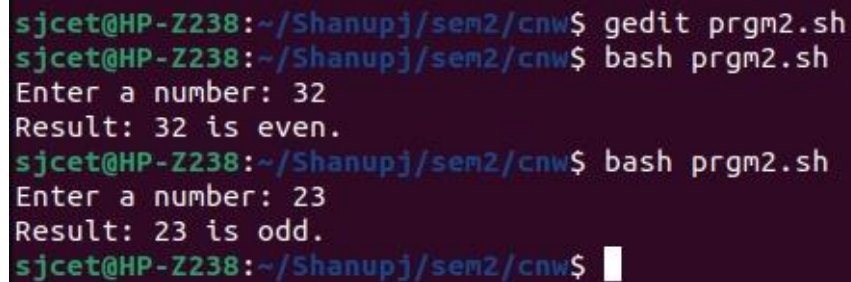
A port has the following characteristics –

- External devices are connected to a computer using cables and ports.
- Ports are slots on the motherboard into which a cable of an external device is plugged in.

- Examples of external devices attached via ports are the mouse, keyboard, monitor, microphone, speakers, etc.

2. Write a Shell program to check if the given number is even or odd.

```
echo -n "Enter a number: "  
read n  
echo -n "Result: "  
if [ `expr $n % 2` == 0 ]  
then  
echo "$n is even."  
else  
echo "$n is odd."  
fi
```



```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm2.sh  
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm2.sh  
Enter a number: 32  
Result: 32 is even.  
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm2.sh  
Enter a number: 23  
Result: 23 is odd.  
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

3. Write a Shell program to check a leap year.

```
echo "Enter any year: "  
read y  
year=$y  
y=$(( $y % 4 ))  
if [ $y -eq 0 ]  
then  
echo "$year is leap year."  
else  
echo "$year is not a leap year."  
fi
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm3.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm3.sh
Enter any year:
2012
2012 is leap year.
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm3.sh
Enter any year:
2009
2009 is not a leap year.
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

4. Write a Shell program to find the area and circumference of a circle.

```
echo "Enter the radius of the circle: "
read r
area=$(echo "3.14*$r*$r" | bc )
circum=$(echo "3.14*2*$r" | bc)
echo "area of the circle is: " $area
echo "circumference of the circle is: " $circum
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm4.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm4.sh
Enter the radius of the circle:
3.5
area of the circle is: 38.46
circumference of the circle is: 21.98
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

5. Write a Shell program to check the given number and its reverse are the same.

```
echo "Enter a number: "
read n
rev=$(echo $n | rev)
if [ $n -eq $rev ]; then
    echo "Number is palindrome"
else
    echo "Number is not palindrome"
fi
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm5.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm5.sh
Enter a number:
12321
Number is palindrome
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm5.sh
Enter a number:
345
Number is not palindrome
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm5.sh
Enter a number:
54
Number is not palindrome
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

6. Write a Shell program to check if the given string is palindrome or not.

```
echo "Enter a String"
read input
reverse=""

len=${#input}
for (( i=len-1; i>=0; i-- ))
do
reverse="$reverse${input:$i:1}"
done
if [ $input == $reverse ]
then
echo "$input is palindrome"
else
echo "$input is not palindrome"
fi
```



```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm6.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm6.sh
Enter a String
malayalam
malayalam is palindrome
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm6.sh
Enter a String
sona
sona is not palindrome
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

7. Write a Shell program to find the sum of odd and even numbers from a set of numbers.

```
echo "enter"
read num
rev=0
even=0
odd=0
while [ $num -gt 0 ]
do
tmp=$(( $num % 10 ))
if(( $tmp % 2 == 0 ))
then
even=$(( $even + $tmp ))
else
odd=$(( $odd + $tmp ))
fi
rev=$(( $rev * 10 + $tmp ))
num=$(( $num / 10 ))
done
echo the sum of even number $even
echo the sum of odd number $odd
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm7.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm7.sh
enter
987654321
the sum of even number 20
the sum of odd number 25
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```


8. Write a Shell program to find the roots of a quadratic equation.

```

echo "Enter the coefficients of the quadratic equation (a, b, c): "
read a b c
discriminant=$(echo "$b^2-4*$a*$c" | bc)
if [ $discriminant -gt 0 ]
then
    root1=$(echo "scale=2; (-$b + sqrt($discriminant)) / (2*$a)" | bc)
    root2=$(echo "scale=2; (-$b - sqrt($discriminant)) / (2*$a)" | bc)
    echo "The roots are $root1 and $root2"
elif [ $discriminant -eq 0 ]
then
    root=$(echo "scale=2; (-$b / (2*$a))" | bc)
    echo "The root is $root"
else
    real=$(echo "scale=2; (-$b / (2*$a))" | bc)
    imag=$(echo "scale=2; sqrt(-$discriminant) / (2*$a)" | bc)
    echo "The roots are $real + i$imag and $real - i$imag"
fi

```

```

sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm8.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm8.sh
Enter the coefficients of the quadratic equation (a, b, c):
4 0 -25
The roots are 2.50 and -2.50

```

9. Write a Shell program to check if the given integer is an Armstrong number or not.

```

echo "Enter a number: "
read c

x=$c
sum=0
r=0
n=0
while [ $x -gt 0 ]
do
    r=`expr $x % 10`
    n=`expr $r \* $r \* $r`
    sum=`expr $sum + $n`
    x=`expr $x / 10`

```

```
done
```

```
if [ $sum -eq $c ]
```

```
then
```

```
echo "It is an Armstrong Number."
```

```
else
```

```
echo "It is not an Armstrong Number."
```

```
fi
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm9.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm9.sh
Enter a number:
153
It is an Armstrong Number.
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm9.sh
Enter a number:
234
It is not an Armstrong Number.
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

10. Write a Shell program to check if the given integer is prime or not.

```
echo -e "Enter Number : \c"
```

```
read n
```

```
for((i=2; i<=$n/2; i++))
```

```
do
```

```
ans=$(( n%i ))
```

```
if [ $ans -eq 0 ]
```

```
then
```

```
echo "$n is not a prime number."
```

```
exit 0
```

```
fi
```

```
done
```

```
echo "$n is a prime number."
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm10.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm10.sh
Enter Number : 9
9 is not a prime number.
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm10.sh
Enter Number : 7
7 is a prime number.
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

11. Write a Shell program to generate prime numbers between 1 and 50.

```
echo enter m and n
read m n

for a in $(seq $m $n)
do
    k=0
    for i in $(seq 2 $(expr $a - 1))
    do
        if [ $(expr $a % $i) -eq 0 ]
        then
            k=1
            break
        fi
    done
    if [ $k -eq 0 ]
    then
        echo $a
    fi
done
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm11.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm11.sh
enter m and n
1 50
1
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

12. Write a Shell program to find the sum of squares of individual digits of a number.

```
echo "Enter a number: "
read number
sum=0
while [[ $number -gt 0 ]]; do
    digit=$((number%10))
    sum=$((sum + digit*digit))
    number=$(( number / 10 ))
done
echo "The sum of the squares of the digits is: $sum"
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm12.sh
Enter a number:
325
The sum of the squares of the digits is: 38
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

13. Write a Shell program to count the number of vowels in a line of text.

```
echo "Enter a line of text: "
```

```

read line
vowel_count=0
for (( i=0; i<${#line}; i++ )); do
char=${line:i:1}
case $char in
    [aeiouAEIOU])
        vowel_count=$((vowel_count + 1))
        ;;
    esac
done
echo "The number of vowels in the line of text is: $vowel_count"

```

```

sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm13.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm13.sh
Enter a line of text:
hello welcome
The number of vowels in the line of text is: 5
sjcet@HP-Z238:~/Shanupj/sem2/cnw$

```

14. Write a Shell program to display student grades.

```

input_file="prgm14.txt"
if [[ ! -f "$input_file" ]]; then
echo "Input file not found!"
exit 1
fi
declare -A prgm14
while read line; do
name=$(echo "$line" | cut -d ',' -f 1)
grade=$(echo "$line" | cut -d ',' -f 2)
prgm14["$name"]=$grade
done < "$input_file"
for name in "${!prgm14[@]}"; do
echo "$name: ${prgm14[$name]}"
done

```

prgm14.txt

```

Alice,85
Bob,90
Charlie,70

```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm14.txt
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm14.sh
Charlie: 70
Alice: 85
Bob: 90
```

15. Write a Shell program to find the smallest and largest numbers from a set of numbers.

```
numbers=(5 3 8 1 9 4 7 2)
smallest=${numbers[0]}
largest=${numbers[0]}
for number in "${numbers[@]"; do
if (( number < smallest )); then
    smallest=$number
fi
if (( number > largest )); then
    largest=$number
fi
done
echo "Smallest number : $smallest"
echo "Largest number : $largest"
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm15.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm15.sh
Smallest number : 1
Largest number : 9
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

16. Write a Shell program to find the smallest digit from a number.

```
echo "Enter a number:"
read number
smallest=${number:0:1}
for (( i=1; i<${#number}; i++ )); do
digit=${number:i:1}
if (( digit < smallest )); then
    smallest=$digit
fi
done
```

```
fi
done
echo "Smallest digit: $smallest"
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm16.sh
Enter a number:
786895
Smallest digit: 5
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

17. Write a Shell program to find the sum of all numbers between 50 and 100, which are divisible by 3 and not divisible by 5.

```
sum=0
for (( i=50; i<=100; i++ )); do
if (( i % 3 == 0 )) && (( i % 5 != 0 )); then
    sum=$(( sum + i ))
fi
done
echo "Sum of numbers between 50 and 100, which are divisible by 3 and not divisible by 5: $sum"
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm17.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm17.sh
Sum of numbers between 50 and 100, which are divisible by 3 and not divisible by 5: 1050
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

18. Write a Shell program to find the second highest number from a set of numbers.

```
numbers=(5 3 8 1 9 4 7 2)
highest=${numbers[0]}
second_highest=${numbers[0]}
for number in "${numbers[@]"; do
if(( number > highest )); then
    second_highest=$highest
    highest=$number
elif (( number != highest )) && (( number > second_highest )); then
    second_highest=$number
fi
done
echo "Second highest number: $second_highest"
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm18.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm18.sh
Second highest number: 8
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

19. Write a Shell program to find the sum of digits of a number using a function.

```
function sum_of_digits {
local number=$1
local sum=0
while (( number > 0 )); do
    sum=$(( sum + number % 10 ))
    number=$(( number / 10 ))
done
echo "$sum"
}
echo "Enter a number: "
read number
result=$(sum_of_digits $number)
echo "Sum of digits of $number: $result"
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm19.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm19.sh
Enter a number:
653
Sum of digits of 653: 14
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

20. Write a Shell program to print the reverse of a number using a function.

```
function reverse_number {
local number=$1
local reverse=0
while (( number > 0 )); do
    reverse=$(( reverse * 10 + number % 10 ))
    number=$(( number / 10 ))
done
echo "$reverse"
}
echo "Enter a number:"
read number
```



```
result=$(reverse_number $number)
echo "Reverse of $number: $result"
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm20.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm20.sh
Enter a number:
7654
Reverse of 7654: 4567
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

21. Write a Shell program to find the factorial of a number using a for loop.

```
echo -n "Enter a number: "
read number
factorial=1
for(( i=1; i<=number; i++ ))
do
    factorial=$((factorial * $i))
done
echo "The factorial of $number is $factorial"
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm21.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm21.sh
Enter a number: 5
The factorial of 5 is 120
```

22. Write a Shell program to generate Fibonacci series.

```
echo "How many numbers do you want from the Fibonacci series ?"
read total
x=0
y=1
i=2
echo "Fibonacci Series up to $total terms :: "
echo "$x"
echo "$y"
while [ $i -lt $total ]
do
    i=$((i + 1))
    z=$((x + y))
    echo "$z"
```

```
x=$y
y=$z
done
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm22.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm22.sh
How many numbers do you want of Fibonacci series ?
5
Fibonacci Series up to 5 terms ::
0
1
1
2
3
```

23. Write a shell script, which receives two filenames as arguments. It checks whether the two files contents are the same or not. If they are the same then the second file is deleted.

```
if [ $# -ne 2 ]; then
    echo "Usage: $0 file1 file2"
    exit 1
fi

if cmp -s "$1" "$2"; then
    echo "The contents of $1 and $2 are the same. Deleting $2..."
    rm "$2"
else
    echo "The contents of $1 and $2 are different."
fi
```

```
sjcet@HP-Z238:~$ cd Shanupj/sem2/cnw
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm23.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm23.sh
Usage: prgm23.sh file1 file2
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

24. Write a Menu driven Shell script that Lists current directory, Prints Working Directory, displays Date and displays Users logged in

```
while true
```

```
do
  clear
  echo "Menu:"
  echo "1. List current directory"
  echo "2. Print working directory"
  echo "3. Display date"
  echo "4. Display users logged in"
  echo "5. Exit"

  read -p "Enter your choice: " choice

  case $choice in
    1)
      ls -l
      read -p "Press enter to continue"
      ;;
    2)
      pwd
      read -p "Press enter to continue"
      ;;
    3)
      date
      read -p "Press enter to continue"
      ;;
    4)
      who
      read -p "Press enter to continue"
      ;;
    5)
      exit 0
      ;;
    *)
      echo "Invalid choice. Press enter to try again"
      read
      ;;
  esac
done
```

```
Menu:
1. List current directory
2. Print working directory
3. Display date
4. Display users logged in
5. Exit
Enter your choice: 2
/home/sjcet/Shanupj/sem2/cnw
Press enter to continue
```

```
Menu:
1. List current directory
2. Print working directory
3. Display date
4. Display users logged in
5. Exit
Enter your choice: 1
total 104
-rw-rw-r-- 1 sjcet sjcet 12 Apr 18 14:07 file1.txt
-rw-rw-r-- 1 sjcet sjcet 12 Apr 18 14:08 file2.txt
-rw-rw-r-- 1 sjcet sjcet 195 Mar 30 15:51 prgm10.sh
-rw-rw-r-- 1 sjcet sjcet 264 Mar 30 15:59 prgm11.sh
-rw-rw-r-- 1 sjcet sjcet 209 Apr 11 14:28 prgm12.sh
-rw-rw-r-- 1 sjcet sjcet 255 Apr 11 14:48 prgm13.sh
-rw-rw-r-- 1 sjcet sjcet 332 Apr 11 15:06 prgm14.sh
-rw-rw-r-- 1 sjcet sjcet 27 Apr 11 14:58 prgm14.txt
-rw-rw-r-- 1 sjcet sjcet 292 Apr 11 15:16 prgm15.sh
-rw-rw-r-- 1 sjcet sjcet 209 Apr 11 15:25 prgm16.sh
-rw-rw-r-- 1 sjcet sjcet 211 Apr 11 15:53 prgm17.sh
-rw-rw-r-- 1 sjcet sjcet 336 Apr 11 16:06 prgm18.sh
-rw-rw-r-- 1 sjcet sjcet 272 Apr 11 16:14 prgm19.sh
-rw-rw-r-- 1 sjcet sjcet 288 Apr 11 16:20 prgm20.sh
-rw-rw-r-- 1 sjcet sjcet 167 Apr 3 15:57 prgm21.sh
-rw-rw-r-- 1 sjcet sjcet 235 Apr 3 16:01 prgm22.sh
-rw-rw-r-- 1 sjcet sjcet 235 Apr 18 14:04 prgm23.sh
-rw-rw-r-- 1 sjcet sjcet 780 Apr 18 14:10 prgm24.sh
-rw-rw-r-- 1 sjcet sjcet 132 Mar 30 14:31 prgm2.sh
-rw-rw-r-- 1 sjcet sjcet 146 Mar 30 14:52 prgm3.sh
-rw-rw-r-- 1 sjcet sjcet 199 Mar 30 14:58 prgm4.sh
-rw-rw-r-- 1 sjcet sjcet 151 Mar 30 15:07 prgm5.sh
-rw-rw-r-- 1 sjcet sjcet 238 Mar 30 15:18 prgm6.sh
-rw-rw-r-- 1 sjcet sjcet 289 Mar 30 15:23 prgm7.sh
-rw-rw-r-- 1 sjcet sjcet 626 Apr 3 15:12 prgm8.sh
-rw-rw-r-- 1 sjcet sjcet 262 Mar 30 15:48 prgm9.sh
Press enter to continue
```

```
Menu:
1. List current directory
2. Print working directory
3. Display date
4. Display users logged in
5. Exit
Enter your choice: 3
Tuesday 18 April 2023 02:12:34 PM IST
Press enter to continue
```

```
Menu:
1. List current directory
2. Print working directory
3. Display date
4. Display users logged in
5. Exit
Enter your choice: 4
sjcet  tty2      2023-04-18 13:44 (tty2)
Press enter to continue
```

25. Shell script to check executable rights for all files in the current directory, if a file does not have the execute permission then make it executable.

```
for file in *; do
if [[ ! -x "$file" ]]; then
    chmod +x "$file"

echo "Made $file
executable"

fi
done
```



```
Made 10.sh executable
Made 11.sh executable
Made 12.sh executable
Made 13.sh executable
Made 14.sh executable
Made 15.sh executable
Made 16.sh executable
Made 17.sh executable
Made 18.sh executable
Made 19.sh executable
Made 20.sh executable
Made 21.sh executable
Made 22.sh executable
Made 23.sh executable
Made 24.sh executable
Made 25.sh executable
Made 26.sh executable
Made 27.sh executable
Made 28.sh executable
Made 29.sh executable
Made 2.sh executable
Made 30.sh executable
Made 31.sh executable
Made 32.sh executable
Made 33.sh executable
Made 34.sh executable
Made 3.sh executable
```

26. Write a Shell program to generate all combinations of 1, 2, and 3 using a loop.

```
for i in 1 2 3
do
  for j in 1 2 3
  do
    for k in 1 2 3
    do
      echo "$i $j $k"
    done
  done
done
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm26.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm26.sh
1 1 1
1 1 2
1 1 3
1 2 1
1 2 2
1 2 3
1 3 1
1 3 2
1 3 3
2 1 1
2 1 2
2 1 3
2 2 1
2 2 2
2 2 3
2 3 1
2 3 2
2 3 3
3 1 1
3 1 2
3 1 3
3 2 1
3 2 2
3 2 3
3 3 1
3 3 2
3 3 3
```

27. Write a Shell program to create the number series.

```
1
2 3
4 5 6
7 8 9 10
```

```
num=1
row=1
```

```
while [ $row -le 4 ]; do
  for (( i=1; i<=$row; i++ )); do
    echo -n "$num "
    num=$((num+1))
  done
```

```
echo ""
row=$((row+1))
done
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm27.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm27.sh
1
2 3
4 5 6
7 8 9 10
```

28. Write a Shell program to create Pascal's triangle.

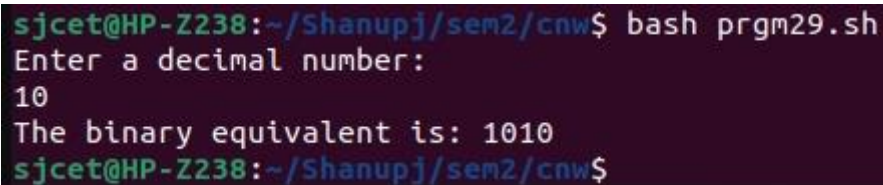
```
function binom {
    if [ $2 -eq 0 ] || [ $2 -eq $1 ]; then
        echo 1
    else
        echo $(( (binom $(( $1-1 )) $(( $2-1 ))) + (binom $(( $1-1 )) $2) ))
    fi
}
echo "Enter the number of rows in Pascal's triangle: "
read rows

for (( i=0; i<$rows; i++ )); do
    for (( j=0; j<=$i; j++ )); do
        val=$(binom $i $j)
        echo -n "$val "
    done
    echo ""
done
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm28.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm28.sh
Enter the number of rows in Pascal's triangle:
5
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```


29. Write a Decimal to Binary Conversion Shell Script

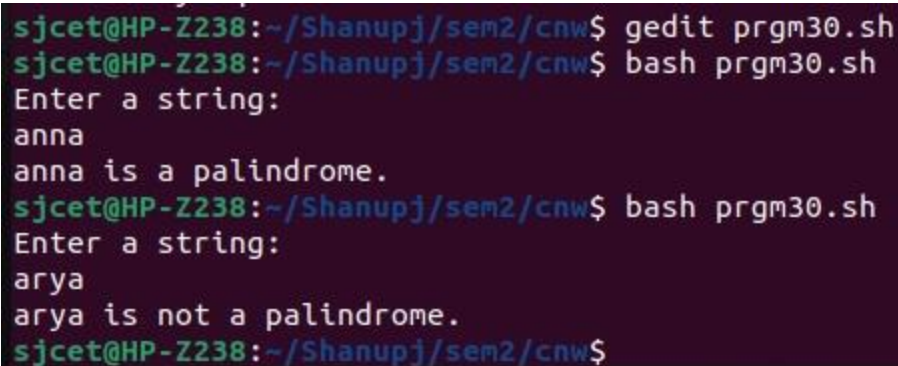
```
echo "Enter a decimal number: "  
read decimal  
binary=""  
while [ $decimal -gt 0 ]; do  
    remainder=$((decimal % 2))  
    binary="$remainder$binary"  
    decimal=$((decimal / 2))  
done  
echo "The binary equivalent is: $binary"
```



```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm29.sh  
Enter a decimal number:  
10  
The binary equivalent is: 1010  
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

30. Write a Shell Script to Check Whether a String is Palindrome or not

```
echo "Enter a string: "  
read string  
reverse=$(echo $string | rev)  
if [ "$string" == "$reverse" ]; then  
    echo "$string is a palindrome."  
else  
    echo "$string is not a palindrome."  
fi
```



```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm30.sh  
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm30.sh  
Enter a string:  
anna  
anna is a palindrome.  
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm30.sh  
Enter a string:  
arya  
arya is not a palindrome.  
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

31. Write a shell script to find out the unique words in a file and also count the occurrence of each of these words.

```
echo "Enter the file name: "
read file

if [ ! -f "$file" ]; then
    echo "File not found."
    exit 1
fi

contents=$(tr '[:upper:]' '[:lower:]' < $file | sed 's/^[^a-z0-9]/ /g')
words=( $contents )

declare -A count
for word in "${words[@]"; do
    if [ -n "$word" ]; then
        ((count[$word]++))
    fi
done

echo "Unique words in $file:"
for word in "${!count[@]"; do
    echo "$word: ${count[$word]}"
done
```

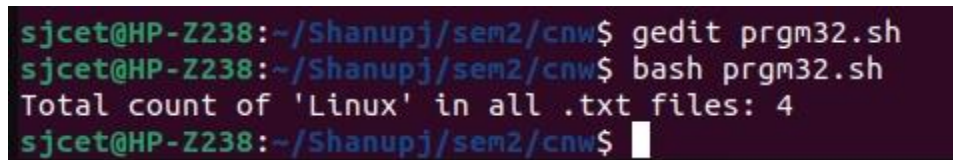


```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm31.sh
Enter the file name:
file1.txt
Unique words in file1.txt:
linux: 2
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

32. Write a shell script to get the total count of the word “Linux” in all the “.txt” files and also across files present in subdirectories.

```
search_dir="."
files=$(find "$search_dir" -type f -name "*.txt")
count=0
for file in $files; do
```

```
occurrences=$(grep -o "Linux" "$file" | wc -l)
count=$((count + occurrences))
done
echo "Total count of 'Linux' in all .txt files: $count"
```



```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm32.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm32.sh
Total count of 'Linux' in all .txt files: 4
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

- 33. Write a shell script to validate password strength. Here are a few assumptions for the password string. Length – minimum of 8 characters. Contain both the alphabet and number. Include both the small and capital case letters.**

```
read -p "Enter your password: " password

if [[ ${#password} -lt 8 ]]; then
    echo "Password length must be at least 8 characters."
    exit 1
fi

if ! [[ "$password" =~ [A-Za-z]+[0-9]+ ]]; then
    echo "Password must contain both alphabet and number."
    exit 1
fi

if ! [[ "$password" =~ [a-z]+ ]] || ! [[ "$password" =~ [A-Z]+ ]]; then
    echo "Password must include both small and capital case letters."
    exit 1
fi

echo "Password is valid."
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm33.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm33.sh
Enter your password: 123
Password length must be at least 8 characters.
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm33.sh
Enter your password: saranyamohan
Password must contain both alphabet and number.
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm33.sh
Enter your password: saranya0308@
Password must include both small and capital case letters.
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm33.sh
Enter your password: Saranya0308@
Password is valid.
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

34. Write a shell script to print the count of files and subdirectories in the specified directory.

```
if [ $# -eq 0 ]; then
    echo "Usage: $0 directory"
    exit 1
fi

directory=$1

if [ ! -d $directory ]; then
    echo "Error: $directory is not a directory"
    exit 1
fi

num_files=$(find $directory -maxdepth 1 -type f | wc -l)
num_dirs=$(find $directory -maxdepth 1 -type d | wc -l)

echo "Number of files in $directory: $num_files"
echo "Number of directories in $directory: $((num_dirs - 1))"
```

```
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ gedit prgm34.sh
sjcet@HP-Z238:~/Shanupj/sem2/cnw$ bash prgm34.sh
Usage: prgm34.sh directory
sjcet@HP-Z238:~/Shanupj/sem2/cnw$
```

35. Write a shell script to reverse the list of strings and reverse each string further in the list.

```
#!/bin/bash
# Define a list of strings
my_list=("string1" "string2" "string3"
"string4") # Reverse the order of the
list my_list=$(echo "${my_list[@]}" |
tr ' ' '\n' | tac | tr '\n' ' ')
```

```
# Reverse each string in the list
for i in "${!my_list[@]}" do
my_list[$i]=`echo
${my_list[$i]} | rev` done
```

```
# Print the reversed list of strings
echo "${my_list[@]}"
```

```
4gnirts 3gnirts 2gnirts 1gnirts
```