

OS Week 2 LAB ASSIGNMEN

Question no:1

Aim: What is Shell? Explain about types of Shells.

Description:

SHELL is a program which provides the interface between the user and an operating system. When the user logs in OS starts a shell for user. **Kernel** controls all essential computer operations, and provides the restriction to hardware access, coordinates all executing utilities, and manages Resources between process. Using kernel only user can access utilities provided by operating system.

Types of Shell:

- **C Shell**

1. C shell (csh) is a Unix shell that provides a command-line user interface (UI) to interact with an operating system (OS). Created by Bill Joy at the University of California at Berkeley in the late 1970s, C shell is one of the oldest Unix shells used today. It was developed as an alternative to Unix's original shell, the Bourne shell. These two Unix shells and the Korn shell are the three most used shells.
2. Denoted as **csh**
3. Syntax:
 - Command full-path name is /bin/csh,
 - Non-root user default prompt is hostname %,
 - Root user default prompt is hostname #.

- **The Bourne Shell**

1. The Bourne shell, often referred to as the "sh" shell, is one of the earliest and most basic Unix shells. It was developed by Stephen Bourne at AT&T Bell Labs in the late 1970s and was the default shell for Unix systems for many years. The Bourne shell is a command-line interface used for interacting with Unix-like operating systems, such as Linux and various versions of Unix.
2. Denoted as **sh**
3. Syntax:
 - command full-path name is /bin/sh and /sbin/sh,
 - Non-root user default prompt is \$,
 - Root user default prompt is #.

- **The Korn Shell**

1. The Korn shell, often referred to as "ksh," is another Unix shell that was developed by David Korn at AT&T Bell Laboratories in the early 1980s. It is an extension and enhancement of the original Bourne shell (sh) and was designed to provide a more powerful and user-friendly command-line interface with additional features and

capabilities. The Korn shell was intended to bridge the gap between the simplicity of the Bourne shell and the complexity of the C shell (csh).

2. It is denoted as **ksh**
3. Syntax:
 - Command full-path name is /bin/ksh,4
 - Non-root user default prompt is \$,
 - Root user default prompt is #.

- **GNU Bourne-Again Shell**

1. The GNU Bourne-Again Shell, commonly referred to as "Bash," is a Unix shell and command language that was created as a free and open-source replacement for the original Bourne Shell (sh). Bash was developed by Brian Fox for the Free Software Foundation (FSF) as part of the GNU Project. It has become one of the most widely used and versatile shells in the Unix and Unix-like operating systems world.
2. Denoted as **bash**
3. Command full-path name is /bin/bash,
Default prompt for a non-root user is bash-g.gg\$
(g.gg indicates the shell version number like bash-3.50\$),
Root user default prompt is bash-g.gg#.

- **T Shell**

1. The "T shell" or "tcsh" is a Unix shell that is an enhanced version of the C shell (csh). It was developed by Ken Greer at the University of California, Berkeley, and released in 1986. The name "tcsh" stands for "TENEX C Shell," named after the TENEX operating system, which was an early time-sharing operating system. It was originally developed for the Plan 9 operating system, but has since been ported to other systems, including Linux, FreeBSD, and macOS.
2. Denoted as **tsh**
3. Command full-path name is /bin/tcsh,
Default prompt for a non-root user is abhishekaslk(user):~>
Root user default prompt is root@abhishekaslk(user):~#.

Question no:2

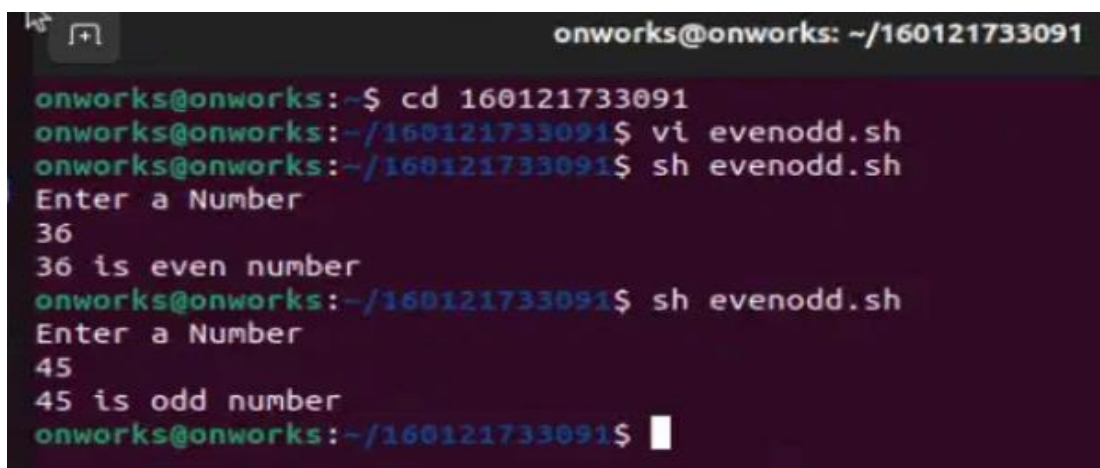
Aim: Write a Shell program to

- i. To check given number is even or odd
- ii. To check given number is prime or not
- iii. To check given number is palindrome number or not
- iv. To check given number is Armstrong number or nit

Code:

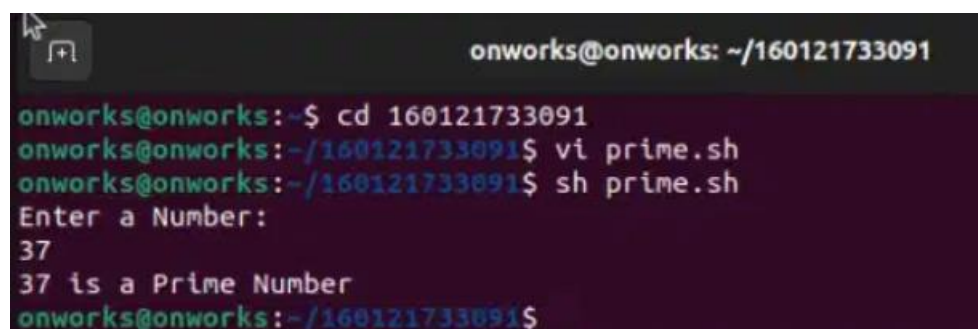
1. Write a Shell program to
 - i. To check given number is even or odd.

```
echo "Enter a Number"
read n
if [ $((($n % 2)) -eq 0 ]
then
echo "$n is even number"
else
echo "$n is odd number"
fi
```



```
onworks@onworks: ~/160121733091
onworks@onworks:~$ cd 160121733091
onworks@onworks:~/160121733091$ vi evenodd.sh
onworks@onworks:~/160121733091$ sh evenodd.sh
Enter a Number
36
36 is even number
onworks@onworks:~/160121733091$ sh evenodd.sh
Enter a Number
45
45 is odd number
onworks@onworks:~/160121733091$
```

- ii. To check given number is prime or not



```
onworks@onworks: ~/160121733091
onworks@onworks:~$ cd 160121733091
onworks@onworks:~/160121733091$ vi prime.sh
onworks@onworks:~/160121733091$ sh prime.sh
Enter a Number:
37
37 is a Prime Number
onworks@onworks:~/160121733091$
```

```

echo "Enter a number: "
read n
i=2
if [ $n -lt 2 ]
then
    echo "$n is not a prime number"
else
    while [ $i -lt $n ]
    do
        if [ $(( $n % $i )) -eq 0 ]
        then
            echo "$n is not a prime number"
            exit
        fi
        i=$(( $i + 1 ))
    done
    echo "$n is a prime number"
fi

```

- i) To check given number is palindrome number or not

```

onworks@onworks: ~/160121733091
onworks@onworks:~$ cd 160121733091
onworks@onworks:~/160121733091$ vi palindrome.sh
onworks@onworks:~/160121733091$ sh palindrome.sh
Enter a number:
16011061
16011061 is a Palindrome
onworks@onworks:~/160121733091$

```

```

echo "Enter a number: "
read n
temp=$n
rev=0
rem=0
while [ $n -gt 0 ]
do
    rem=$(( $n % 10 ))
    rev=$(( ($rev * 10) + $rem ))
    n=$(( $n / 10 ))
done
if [ $temp -eq $rev ]
then
    echo "$temp is palindrome"
else
    echo "$temp is not Palindrome"
fi

```

- ii) To check given number is Armstrong number or not

```
onworks@onworks: ~/160121733091
onworks@onworks:~$ cd 160121733091
onworks@onworks:~/160121733091$ vi prime.sh
onworks@onworks:~/160121733091$ sh prime.sh
Enter a Number:
370
370 is an armstrong number
```

```
echo "Enter a number: "
read n
temp=$n
sum=0
x=0
r=0
while [ $temp -gt 0 ]
do
    r=$((temp % 10))
    x=$((($r * $r * $r))
    sum=$((sum + $x))
    temp=$((temp / 10))
done
if [ $sum -eq $n ]
then
    echo "$n is armstrong"
else
    echo "$n is not armstrong"
fi
```

Question no:3

Aim: Write about the following

- i) Shell
- ii) Kernel
- iii) Terminal

Description:

i) Shell:

A shell is a command-line interface (CLI) program that acts as an intermediary between a user and the operating system (OS). It provides a way for users to interact with the computer's underlying OS by accepting and interpreting commands entered by the user and then executing those commands. Shells can be thought of as command interpreters or command processors.

There are various shell programs available for Unix-like and Linux operating systems, each with its own set of features and capabilities. Some of the most common shells include:

- **Bash (Bourne-Again Shell):** Bash is one of the most widely used shells on Unix and Linux systems. It is known for its powerful scripting capabilities, command line editing, and extensive features.
- **Zsh (Z Shell):** Zsh is another highly customizable and feature-rich shell that offers advanced command-line editing and a wide range of plugins and extensions.
- **Fish (Friendly Interactive Shell):** Fish is designed to be user-friendly and provides auto-suggestions, syntax highlighting, and a simpler, more intuitive command syntax.

Shells are responsible for interpreting and executing commands, managing processes, handling I/O redirection, and more. They offer a flexible and powerful way for users to interact with their operating systems and automate tasks through shell scripting.

ii) Kernel:

The kernel is the core component of an operating system. It is responsible for managing and controlling the hardware resources of a computer and acts as an intermediary between the hardware and software layers of a system. The primary functions of a kernel include:

- **Process Management:** The kernel manages processes, which are individual programs or tasks running on the computer. It schedules processes, allocates CPU time, and handles process creation and termination.
- **Memory Management:** It manages the computer's physical and virtual memory, ensuring that processes have access to the memory they need and protecting memory from unauthorized access.
- **File System Management:** The kernel provides access to the file system, allowing processes to read, write, and manipulate files and directories.
- **Device Management:** It controls interactions with hardware devices such as disks, network interfaces, and input/output devices.

- **Security:** The kernel enforces security policies, controls access to system resources, and protects against unauthorized access and malicious activities.
- **Interprocess Communication (IPC):** It provides mechanisms for processes to communicate and share data with each other.

Different operating systems have different kernels, with some of the most well-known kernels including the Linux kernel (used in Linux distributions), the Windows NT kernel (used in modern Windows operating systems), and the macOS kernel (used in Apple's macOS).

iii) Terminal:

A terminal, often referred to as a "command-line terminal" or simply "console," is a text-based interface that allows users to interact with a computer or operating system by entering commands and receiving text-based output. Terminals provide a way to access the command-line interface (CLI) of an operating system.

Key characteristics of a terminal include:

- **Text-Based:** Unlike graphical user interfaces (GUIs), terminals are entirely text-based. Users type commands and view text-based output.
- **Shell Interaction:** Terminals provide an environment where users can run shell programs (such as Bash, Zsh, or Fish) and execute various commands.
- **Scripting:** Terminals are used for running scripts, which are sequences of commands saved in a file for automation and repetitive tasks.
- **Remote Access:** Terminals can be used to remotely access and manage other computers and servers over a network using protocols like SSH (Secure Shell).
- **Customization:** Users can customize the appearance and behavior of their terminal windows to suit their preferences.

Terminals are a powerful tool for system administration, software development, and other tasks that require precise control over a computer's operations. They provide direct access to the

underlying operating system, making it possible to perform a wide range of tasks efficiently and flexibly.

Question no:4

Aim: Write about the following

- i) Base 10 number to binary
- ii) Base 10 to octal
- iii) Base 10 to Hexa decimal
- iv) Base 10 to Base 5

Code:

- i) Base 10 number to binary

```
onworks@onworks: ~/160121733091
onworks@onworks:~$ cd 160121733091
onworks@onworks:~/160121733091$ vi dectobin.sh
onworks@onworks:~/160121733091$ sh dectobin.sh
Enter a number:
15
1111
```

```
echo "Enter a number: "
read n
val=0
power=1
while [ $n -ne 0 ]
do
    r=$(( $n % 2 ))
    val=$(( ($r * $power) + $val ))
    power=$(( $power * 10 ))
    n=$(( $n / 2 ))
done
echo "$val"
```


ii) Base 10 to octal

```
onworks@onworks: ~/160121733091
onworks@onworks:~$ cd 160121733091
onworks@onworks:~/160121733091$ vi dectooct.sh
onworks@onworks:~/160121733091$ sh dectooct.sh
Enter a number:
15
17
onworks@onworks:~/160121733091$
```

```
echo "Enter a number: "
read n
val=0
power=1
while [ $n -ne 0 ]
do
    r=$(( $n % 8 ))
    val=$(( ($r * $power) + $val ))
    power=$(( $power * 10 ))
    n=$(( $n / 8 ))
done
echo "$val"
```

iii) Base 10 to Hexa decimal

```
onworks@onworks: ~/160121733091
onworks@onworks:~$ cd 160121733091
onworks@onworks:~/160121733091$ vi dectohex.sh
onworks@onworks:~/160121733091$ sh dectohex.sh
Enter a number:
15
F
```

```

echo "Enter a number: "
read n
val=0
hex=(0 1 2 3 4 5 6 7 8 9 A B C D E F)
while [ $n -ne 0 ]
do
    r=$(( $n % 16 ))
    val=${hex[r]}$val
    n=$(( $n / 16 ))
done
echo "$val"

```

iv) Base 10 to Base 5

```

onworks@onworks: ~/160121733091
onworks@onworks:~$ cd 160121733091
onworks@onworks:~/160121733091$ vi dectob5.sh
onworks@onworks:~/160121733091$ sh dectob5.sh
Enter a number:
15
30

```

```

echo "Enter a number: "
read n
val=0
power=1
while [ $n -ne 0 ]
do
    r=$(( $n % 5 ))
    val=$(( ($r * $power) + $val ))
    power=$(( $power * 10 ))
    n=$(( $n / 5 ))
done
echo "$val"

```

Question no:5

Aim: Draw the architecture diagram of Linux/Unix Operating System.

Description:**Hardware Layer:**

At the lowest level is the hardware layer, which includes the physical computer components such as CPU, memory, storage devices, and peripheral devices.

Kernel:

Above the hardware layer is the kernel, which is the core of the operating system. The kernel interacts directly with the hardware and provides essential services such as process management, memory management, device management, and system calls for user applications.

System Libraries:

Above the kernel, there are system libraries (e.g., GNU C Library or glibc). These libraries provide a set of functions and routines that applications can use to perform various tasks, making it easier for developers to write software that interacts with the operating system.

Shell and Command-Line Interface (CLI):

The shell is the user interface to the operating system. Users interact with the shell through a command-line interface (CLI) by entering commands and receiving text-based output. Popular shells include Bash, Zsh, and Tcsh.

System Utilities:

System utilities are command-line tools and applications that perform various tasks, such as file manipulation, text processing, networking, and system administration. These utilities often come pre-installed with the operating system.

Graphical User Interface (GUI):

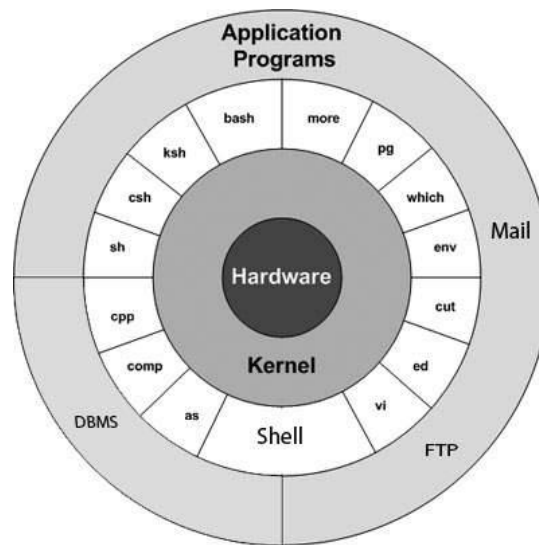
While Linux/Unix systems are traditionally associated with the CLI, many desktop environments and window managers provide a GUI for users who prefer a graphical interface. Common desktop environments include GNOME, KDE, Xfce, and others.

Application Layer:

At the top layer are user applications, including web browsers, email clients, office suites, and software developed for specific purposes. These applications run on top of the operating system and utilize its services.

User Space and Kernel Space:

The operating system is often divided into two main spaces: user space and kernel space. User space contains user applications and libraries, while kernel space contains the kernel and device drivers. Communication between user space and kernel space is managed by the kernel through system calls.



Linux Architecture

