INTRODUCTION

- **What is UNIX**? UNIX is an operating system. Operating system makes the computer work. provides essential services to run – like using the CPU, allocating memory, accessing devices etc.

UNIX is a ***multiprogramming*** system. It permits multiple programs to run simultaneously. This can happen in two ways –

(a) **Multiuser -** Multiple users can run separate jobs that share the system's CPU and resources.
(b) **Multitasking** - A single user can also run multiple jobs.

- **UNIX architecture** The UNIX operating system is made up of KERNEL and SHELL.

The kernel

The Kernel interacts with the machine's hardware. The Kernel is the core of the operating system – a collection of routines mostly written in C. These routines communicate with the hardware directly. It is the part of the UNIX system that is loaded into the memory when the system is booted. User programs (the applications) that need to communicate with the hardware (like the hard disk or the terminal) use the services of the kernel. These programs access the kernel through a set of functions called ***system calls.*** The Kernel also manages the system's memory, schedules processes, performs other tasks.

The shell

The shell acts as an interface between the user and the kernel. When a command is entered through the keyboard. the shell examines the input and finally communicates with the kernel to see that the command is executed.

Users can use different shells on the same machine.

Filename Completion - By typing part of the name of a command, filename or directory and pressing the [**Tab**] key, the shell will complete the rest of the name automatically. If the shell finds more than one name beginning with those letters you have typed, it will beep, prompting you to type a few more letters before pressing the tab key again.

History - The shell keeps a list of the commands you have typed in. If you need to repeat a command, use the cursor keys to scroll up and down the list or type history for a list of previous commands.

As an illustration of the way that the shell and the kernel work together, suppose a user types **rm myfile** (which has the effect of removing the file **myfile**). The shell searches the file store for the file containing the program **rm**, and then requests the kernel, through system calls, to execute the program **rm** on myfile. When the process **rm myfile** has finished running, the shell then returns the UNIX prompt % to the user, indicating that it is waiting for further commands.

Using Vi Editor

The vi editor is available on almost all Unix systems. vi can be used from any type of terminal because it does not depend on arrow keys and function keys--it uses the standard alphabetic keys for commands.

vi (pronounced "vee-eye") is short for "vi"sual editor. It displays a window into the file being edited that shows 24 lines of text. vi is a text editor.

This help note explains the basics of vi:

- opening and closing a file
- moving around in a file
- elementary editing
- vi has many other commands and options not described here.

Modes of operation

vi has three modes of operation:

- command mode
- insert mode
- ex command mode

In command mode, the letters of the keyboard perform editing functions (like moving the cursor, deleting text, etc.). To enter command mode, press the escape <Esc> key.

In insert mode, insertion of new text, editing of existing text etc. operations can be performed.

The ex Command mode permits us to give commands at the command line. The bottom line of the vi is called the command line. All commands entered in the ex command mode are displayed in the command line.

STARTING vi

vi filename	edit a file named "filename"
vi newfile	create a new file named "newfile"

ENTERING TEXT

i	insert text left of cursor
a	append text right of cursor

CLOSING AND SAVING A FILE

ZZ	save file and then quit
:wq!	save file and then quit
:w	save file
:q!	discard changes and quit file

Shell Scripts

Write a shell script that will add three integer numbers, which are supplied by the user and then find their average.

```
# !/bin/bash
#Addition and average of three integer.

echo "Enter the first integer : "
read fno

echo "Enter the second integer : "
read sno

echo "Enter the third integer : "
read tno

sum=`expr $fno + $sno + $tno`
echo "The summation is: $sum"

avg=`expr $sum / 3`
echo "The average is : $avg"
```

Output

```
Enter the first integer : 10

Enter the second integer : 20

Enter the third integer : 30

The summation is : 60

The average is : 20
```

Write a shell script to check whether a given number is positive or negative or zero.

```
#Script to see whether given
number is positive or negative
# !/bin/bash

 echo -n "Enter a number:"
 read num
 if test $num -eq 0
 then
   echo "$num is zero"
 else
   if test $num -gt 0
   then
     echo "$num is positive"
   else
     echo "$num is negative"
   fi
 fi
```

Output

```
Enter a number : 5
5 is positive

Enter a number : -4
-4 is negative

Enter a number : 0
0 is zero
```

Write a shell script to check whether a given number is even or odd.

```
# !/bin/bash
#Check whether a given number is even
or odd.

clear
echo -n "Enter a number : " ; read num
rem=`expr $num % 2`
if [ $rem -eq 0 ] ; then
   echo "The number is even"
else
   echo "The number is odd"
```

Output

```
Enter a number : 4
The number is even

sEnter a number : 7
The number is odd
```

The marks obtained by a student in 3 different subjects are input through the keyboard. The student gets a division as per the following rules:

Average above or equal to 60	**-**	**First division**
Average between 50 and 59	**-**	**Second division**
Average between 40 and 49	**-**	**Third division**
Average less than 40	**-**	**Fail**

Write a shell script to calculate the division obtained by the student.

```
# !/bin/bash
clear
echo "Enter marks in First subject "
read m1

echo "Enter marks in Second subject "
read m2

echo "Enter marks in Third subject "
read m3

total=`expr $m1 + $m2 + $m3`
per=`expr $total / 3`

if [ $per -ge 60 ] ; then
      echo FIRST DIVISION
fi

if [ $per -ge 50 ] && [ $per -lt 60 ]
then
      echo SECOND DIVISION
fi

if [ $per -ge 40 ] && [ $per -lt 50 ]
then
      echo THIRD DIVISION
fi

if [ $per -lt 40 ] ; then
      echo FAIL
fi
```

Output

```
Enter marks in First subject      50

Enter marks in Second subject     60

Enter marks in Third subject      70

FIRST DIVISION
```

Write a shell script to find the factorial of a given number

```
# !/bin/bash
# Find the factorial of a given number

clear
n=0
num=0
fact=1

echo -n "Enter number to find factorial: "
read n
```

Output

```
Enter number to find factorial: 5

Factorial for 5 is 120
```

```
num=$n

while [ $n -ge  1 ]
do
  fact=`expr $fact \* $n`
  n=`expr $n - 1`
done

echo "Factorial for $num is $fact"
```

Write a shell script to find the first n Fibonacci numbers

```
# !/bin/bash
#Fibonacci numbers

clear
echo –n ”Enter the number of
terms : ” ; read n

a=0
b=1

echo “The first $n Fibonacci
numbers are : “
for ((i=1; i<=n; i++))
do
   c=`expr $a + $b`
   b=$a
   a=$c
   echo –n $c” “
done
```

Output

Enter the number of terms : 6

The first 6 Fibonacci numbers are :

1 1 2 3 5 8

Write a shell script to compute ‘m’ to the power of a positive integer ‘n’, i.e. m^n

```
# !/bin/bash

clear
echo –n ”Enter the value of m ” ;
read m
echo –n ”Enter the value of n ” ;
read n

sum=1
i=1

while [ $i –le $n ]
do
  sum=`expr $sum \* $m`
  i=`expr $i + 1`
done

echo –n “The result is : $sum”
```

Output

Enter the value of m 2

Enter the value of n 3

The result is : 8

Write a shell script to find the maximum and minimum number from n given numbers

```
# !/bin/bash

clear
echo –n "Enter how many numbers : "
read n
echo –n "Enter integer value : "
read num

max=$num
min=$num

for ((i=1; i<n; i++))
do
   echo –n "Enter integer value : "
   read newnum
   if [ $newnum –gt $max ] ; then
       max=$newnum
   fi

   if [ $newnum –lt $min ] ; then
       min=$newnum
   fi
done

echo –n "The maximum number is : $max"
echo
echo –n "The minimum number is : $min"
```

Output

```
Enter how many numbers : 5

Enter integer value : 4
Enter integer value : 10
Enter integer value : 1
Enter integer value : 7
Enter integer value : 5


The maximum number is : 10

The minimum number is : 1
```

Write a shell script to print given number's sum of all digits (eg. If number is 123, then it's sum of all digits will be 1+2+3=6)

```
# !/bin/bash

clear
echo –n "Enter a number "
read n

sum=0
sd=0

while [ $n -gt 0 ]
do
    sd=`expr $n % 10`
    sum=`expr $sum + $sd`
    n=`expr $n / 10`
done

echo "Sum of digit for number $n is $sum"
```

Output

```
Enter a number    123

Sum of digit for number 123 is 6
```

Write a shell script to print the given number in reverse order (eg. If the given number is 123, then show as 321)

```
# !/bin/bash

clear
echo –n ”Enter a number ”
read n

rev=0
sd=0

while [ $n -gt 0 ]
do
    sd=`expr $n % 10`
    rev=`expr $rev \* 10  + $sd`
    n=`expr $n / 10`
done

    echo  "Reverse of $n is $rev"
```

Output

```
Enter a number    123

Reverse of 123 is 321
```

Write a shell script to print the Pascal's triangle upto a given number of row.

```
# !/bin/bash

clear
echo –n “Enter number of rows : “
read max
m=0
while [ $m –lt $max ]
do
   x=0; y=1
   while [ $x –le $m ]
   do
     if [ $m –eq 0 ] || [ $x –eq 0 ]
     then
         echo –n $y” “
     else
         a=`expr $m - $x + 1`
         y=`expr $y \* $a / $x`
         echo –n $y” “
     fi
   x=`expr $x + 1`
   done
   echo
   m=`expr $m + 1`
done
```

Output

```
Enter number of rows :
7


1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
```

Write a shell script to print the value of $1^2 + 2^2 + 3^2 + \ldots \ldots + n^2$

```
# !/bin/bash

clear
echo –n “Enter number of terms: “
read n
sum=0
for (( i=1 ; i<=n ; i++))
```

Output

```
Enter number of terms :
4

The value is : 30
```

```
do
   a=`expr 2 \* $i - 1`
   b=`expr 2 \* $i - 1`
   sum=`expr $sum + $a \* $b`
done
echo "The value is : $sum"
```

Write a shell script to sort given 5 number in ascending order using array.

```
# !/bin/bash

declare nos[5]=(4 -1 2 66 10)
# or you can write: nos=(4 -1 2 66 10)

# Prints the array before sorting

echo "Original Numbers in array:"
for (( i=0; i<=4; i++ ))
do
  echo ${nos[$i]}
done

# Now do the Sorting of numbers

for (( i=0; i<=4; i++ ))
do
   for (( j=$i; j<=4; j++ ))
   do
      if [ ${nos[$i]} -gt ${nos[$j]}  ]; then
           t=${nos[$i]}
           nos[$i]=${nos[$j]}
           nos[$j]=$t
      fi
   done
done

# Print the sorted array

echo -e "\nSorted Numbers in Ascending
Order:"
for (( i=0; i <= 4; i++ ))
do
  echo ${nos[$i]}
done
```

Output

Original Numbers in array
4 -1 2 66 10

Sorted numbers in ascending order :

-1 2 4 10 66

Write a shell script to print the maximum and minimum numbers in an array of 10 elements.

```
# !/bin/bash

clear
declare arr[10]

# Input the numbers in the array
```

```
for (( i=0; i<=9; i++ ))
do
  echo -e "Enter the $i th number "
  read arr[$i]
done

# Now display the contents of the array

echo -n "The array is : "
for (( i=0; i<=9; i++ ))
do
  echo -n ${arr[$i]}" "
done

# Display the maximum number

max=${arr[0]}
for ((i=1; i<=9; i++))
do
   if [ $max -lt ${arr[$i]} ] ; then
         max=${arr[$i]}
   fi
done
echo
echo -n "The maximum number is : $max"

# Display the minimum number

min=${arr[0]}
for ((i=1; i<=10; i++))
do
   if [ $min -gt ${arr[$i]} ] ; then
         min=${arr[$i]}
   fi
done
echo
echo –n "The minimum number is : $min"
```

Write a shell script to display the prime numbers from 1 to n (n is a given number)

```
# !/bin/bash

clear
echo -n "Enter the number upto which prime no.
will be counted : "
read n
echo
echo "The prime numbers from 1 to $n are : "
if [ $n -eq 1 ] ; then
    echo $n
else
    for (( j=1; j<=n; j++ ))
    do
        i=2
        q=1
        while [ $i -lt $j ]
        do
```

Output

```
Enter the number upto which prime no. will be counted :

25

The prime numbers from 1 to 25 are :

1
2
3
```

```
            q=`expr $j % $i`
            if [ $q -eq 0 ] ; then
                 break
            else
                 i=`expr $i + 1`
            fi
        done
        if [ $q -ne 0 ] ; then
             echo $j
        fi
    done
fi
```

```
5
7
11
13
17
19
23
```

Write a shell script that reads an integer and test whether it is divisible by 11 without dividing it by 11.

```
clear
echo " Enter the number you want to test "
read num
i=0
while [ $num -ne 0 ]
do
        x[$i]=` expr $numr % 10 `
        num=` expr $num / 10 `
        i=` expr $i + 1 `
done
j=0
s1=0
while [ $j -lt $i ]
do
        s1=` expr $s1 + ${x[$j]} `
        j=` expr $j + 2 `
done
s2=0
k=1
while [ $k -lt $i ]
do
        s2=` expr $s2 + ${x[$k]} `
        k=` expr $k + 2 `
done
if [ $s1 -eq $s2 ] then
        echo " The number is divisible by 11 "
fi
if [ $s1 -ne $s2 ] then
        echo " The number is not divisible by 11 "
fi
```

OUTPUT:

```
Enter the number you want to test        121
The number is divisible by 11
Enter the number you want to test        4561
the number is not divisible by 11
```

Further are the tasks for the students to see the outputs by themselves.

Write a shell script to find whether a given year is leap year or not Years divisible by 4 are leap years, with the exception of centurial years that are not divisible by 400. Therefore, the years 1700, 1800, 1900 and 2100 are not leap years, but 1600, 2000, and 2400 are leap years. Now, summarize the rules:

1. A year that is divisible by 4 is a leap year.
2. Exception to rule : a century year that is divisible by 400 is a leap year. (Century year is the year that is divisible by 100)

```
# !/bin/bash
clear
echo –n "Enter a valid year : " ; read y
y4=`expr $y % 4`
if [ $y4 –ne 0 ] ; then
   echo "$y is not a leap year"
else
   y100=`expr $y % 100`
   if [ $y100 –ne 0 ] ; then
        echo "$y is a leap year"
   else
        y400=`expr $y % 400`
        if [ $y400 –ne 0 ] ; then
              echo "$y is not a leap year"
        else
              echo "$y is a leap year"
        fi
   fi
fi
```

Write a shell script to print a string in reverse order

```
# !/bin/bash

clear
echo "Enter a string : " ; read str
len=${#str}
for ((i=$len; i>=0; i--))
do
  echo –n ${name:#i:1}
done
```

Write a shell script to check whether a given number is a palindrome or not

```
# !/bin/bash

clear
echo "Enter the number : " ; read num
n=$num
while [ $num –ne 0 ]
do
   let dig=num%10
```

```
    let rev=rev\*10+dig
    let num=num/10
done
if [ $n –eq $rev ]
  then
     echo “The number $n is palindrome”
  else
     echo “The number $n is not a palindrome”
fi
```

Write a shell script to check whether a given string is a palindrome or not

```
# !/bin/bash

clear
echo –n “Enter the string : “ ; read str
len=${#str}
palin=”true”
let mid=len/2
let j=len-1
for (( i=0 ; i<$mid; i++ ))
do
   left=${str:$i:1}
   right=${str:$j:1}
   let j=j-1
   if [ $left != $right ] ; then
        palin=”false”
   fi
done
if [ $palin = “true” ] ; then
       echo “$str is a Palindrome”
else
       echo “$str is not a Palindrome”
fi
```

Write a shell script that will add two integers from command line

```
# !/bin/bash

clear
if [ $# -ne 2 ]
then
   echo “Enter 2 numbers “
   exit
```

```
fi
   echo "Sum of $1 and $2 is `expr $1 + $2`"
```

Write a shell script that will accept three numbers from command line and display the biggest among them

```
# !/bin/bash

clear
if [ $# -ne 3 ]
    then
        echo "Enter 3 numbers "
        exit
fi
   n1=$1
   n2=$2
   n3=$3
   if [ $n1 -gt $n2 ] && [ $n1 -gt $n3 ]
      then
         echo "$n1 is Bigest number"
   elif [ $n2 -gt $n1 ] && [ $n2 -gt $n3 ]
      then
         echo "$n2 is Bigest number"
   elif [ $n3 -gt $n1 ] && [ $n3 -gt $n2 ]
      then
         echo "$n3 is Bigest number"
   elif [ $1 -eq $2 ] && [ $1 -eq $3 ] && [ $2 -eq $3 ]
      then
         echo "All the three numbers are equal"
   else
         echo "I can not figure out which number is biger"
   fi
```

References –

1. UNIX Shell Programming. By Yashavant Kanetkar
2. Your UNIX. By Sumitabha Das