**INTRODUCTION:**

**Unix** has two main sets internal implementation and the interface that is seen and used by users. i.e. **Kernel** and **Shell**. Kernel takes care of hardware interface and shell communicates with Kernel.

**Shell** works as interpreter of user commands and translates them into action. The shell forms the outer part of the operating system and forms the interface between the user and the kernel. For each user logged in, there is shell in action. When a command is given by the user, it is examined by the shell and communicated to the kernel for execution. The functions used to communicate with the kernel are called system calls and are built into the kernel. The system calls are in all the flavors of UNIX are the same. Hence, software developed on one UNIX system can be easily run on another UNIX system.

**Shell Script** is a [script](http://www.answers.com/topic/scripting-programming-language) written for the [shell](http://www.answers.com/topic/command-shell), or [command line interpreter](http://www.answers.com/topic/command-line-interface), of an [operating system](http://www.answers.com/topic/operating-system). The shell is often considered a simple [domain-specific programming language](http://www.answers.com/topic/domain-specific-programming-language). Typical operations performed by shell scripts include file manipulation, program execution, and printing text.

Many shell script interpreters double as [command line interface](http://www.answers.com/topic/command-line-interface), such as the various [Unix shells](http://www.answers.com/topic/unix-shell), [Windows Power Shell](http://www.answers.com/topic/windows-powershell) or the [MS-DOS](http://www.answers.com/topic/ms-dos) COMMAND.COM. Others, such as [AppleScript](http://www.answers.com/topic/applescript) or the graphical [Windows Script Host](http://www.answers.com/topic/windows-script-host) (WScript.exe), add scripting capability to computing environments without requiring a command line interface. Other examples of [programming languages](http://www.answers.com/topic/programming-language) primarily intended for shell scripting include [DCL](http://www.answers.com/topic/digital-command-language) and [JCL](http://www.answers.com/topic/jcl). Shell scripts are used for repeatable processes, unattended operation, anything of a repetitive nature, etc. In short, they can be very valuable for just about anything.

There are many possible Unix shells that users have access to. Some of these are:   
sh - Bourne shell (the original shell)   
ksh - Korn shell   
bash - Bourne-again shell   
csh - C shell   
tcsh - variant of the C shell and other features   
zsh - the 'z' shell   
rsh - Restricted shell   
ksh93 - '93 version of ksh

Here we use Bourne shell to execute our shell script programs....

**Program Statement:-**  01 .09 . 2011

**Print the first n Fibonacci numbers.**

**Program code:-**

echo -e "Enter the no. of terms(n): \c"

read n

a=0

b=1

initial=1

echo -e "\n\nThe 1st $n Fibonacci numbers are:-\c"

echo -e "0 1 \c"

n=`expr $n - 2`

while [ $initial -le $n ]

do

result=`expr $a + $b`

echo -e "$result \c"

a=$b

b=$result

initial=`expr $initial + 1`

done

**Output:**

Enter the no. of terms(n): 10

The 1st 10 Fibonacci numbers are:-0 1 1 2 3 5 8 13 21 34

**Discussions:**

At first we enter some no of terms to execute the program and using the following commands we get the Fibonacci series. Here **clear** is used to clear the output screen, **echo** command is used to echo argument , **read** command is used to input the value, use the **expr** command to perform arithmetic or string manipulation and making comparisons, The dollar (**$**) symbol is used in UNIX as an indicator for a variable. There are reserved variables, The bash **while** loop is a control flow statement that allows code or commands to be executed repeatedly based on a given condition.……**Program Statement:-** 08 .09 .2011

**Write a program to generate a multiplication table.**

**Program code:-**

echo -e "\n \c"

for((i=1;i<=10;i++))

do

echo -e " $i\c"

done

echo -e "\n----\c"

for((i=1;i<=10;i++))

do

echo -e "-----\c"

done

for((i=1;i<=10;i++))

do

if [ $i -ne 10 ]

then

echo -e "\n$i |\c"

else

echo -e "\n$i|\c"

fi

for((j=1;j<=10;j++))

do

result=`expr $i \\* $j`

if [ $result -lt 10 ]

then

echo -e " $result\c"

else

echo -e " $result\c"

fi

done

done

echo -e "\n

**Output:**

1 2 3 4 5 6 7 8 9 10

--------------------------------------------------------

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

2 | 2 4 6 8 10 12 14 16 18 20

3 | 3 6 9 12 15 18 21 24 27 30

4 | 4 8 12 16 20 24 28 32 36 40

5 | 5 10 15 20 25 30 35 40 45 50

6 | 6 12 18 24 30 36 42 48 54 60

7 | 7 14 21 28 35 42 49 56 63 70

8 | 8 16 24 32 40 48 56 64 72 80

9 | 9 18 27 36 45 54 63 72 81 90

10| 10 20 30 40 50 60 70 80 90 100

**Discussions:**

We write a logical approach of the multiplication table using the following commands and we get the output of all multiplication. **For** loop can be used where list is basically the list of the values u want to traverse on, The dollar (**$**) symbol is used in UNIX as an indicator for a variable. There are reserved variables, use the **expr** command to perform arithmetic or string manipulation and making comparisons, By **if** command we can perform conditional testing on strings, numbers and files……..

**Program Statement:-** 15 . 09 . 2011

**Write a program to print first n prime numbers.**

**Program code:-**

echo -e "Enter the number of terms: \c"

read no

n=0

a=2

echo -e "\nThe prime numbers are..........\n"

while [ $n -lt $no ]

do

flag=1

for((i=2;i<$a;i++))

do

if [ `expr $a % $i` -eq 0 ]

then

flag=0

break

fi

done

if [ $flag -eq 1 ]

then

echo -e " $a\c"

n=`expr $n + 1`

fi

a=`expr $a + 1`

done

echo -e "\n"

**Output:**

Enter the number of terms: 10

The prime numbers are..........

2 3 5 7 11 13 17 19 23 29

**Discussions**:

At first we enter some terms we find out the prime no.s between then using the following commands we get the prime no.s in the input range. The bash **while** loop is a control flow statement that allows code or commands to be executed repeatedly based on a given condition, By **if** command we can perform conditional testing on strings, numbers and files. **For** loop can be used where list is basically the list of the values u want to traverse on, **flag** is an intermediate variable to store the medium value.……

**Program Statement**:- 22 . 09 . 2011

**Write a shell script to calculate HCF and LCM of two numbers.**

**Program code:**-

echo -e "Enter the 1st number:- \c"

read a

echo -e "Enter the 2nd number:- \c"

read b

n=$a

d=$b

result=`expr $a % $b`

while [ $result -ne 0 ]

do

a=$b

b=$result

result=`expr $a % $b`

done

hcf=$b

echo -e "\nThe H.C.F. is $hcf"

lcm=`expr ( $a \\* $d ) / $hcf`

echo –e "\nThe L.C.M. is $lcm"

**Output:**

**Run 1:**

Enter the 1st number:- 4

Enter the 2nd number:- 6

The H.C.F. is 2

The L.C.M. is 12

**Run 2:**

Enter the 1st number:- 24

Enter the 2nd number:- 36

The H.C.F. is 12

The L.C.M. is 72

**Discussions:**

We enter two number from keyboard, using the commands in the logical approach of the program we get the LCM and HCF of the two no.s. The dollar (**$**) symbol is used in UNIX as an indicator for a variable there are reserved variables, The bash **while** loop is a control flow statement that allows code or commands to be executed repeatedly based on a given condition.……

**Program Statement:**- 03 . 11 . 2011

**Write a shell script to check whether a given year is leap or not. The year is given as command line argument.**

**Program code:-**

if [ $# -lt 1 ]

then

echo -e "Input year does not given."

exit

fi

yy=$1

flag=0

if [ `expr $yy % 4` -eq 0 ]

then

if [ `expr $yy % 100` -eq 0 ]

then

if [ `expr $yy % 400` -eq 0 ]

then

flag=1

fi

else

flag=1

fi

fi

if [ $flag -ne 1 ]

then

echo -e "\n NON-LEAP YEAR."

else

echo -e "\n LEAP YEAR. "

fi

**Output:**

**Run 1:**

[sandip@localhost ~]$ sh leapyear.sh 2000

LEAP YEAR.

**Run 2:**

[sandip@localhost ~]$ sh leapyear.sh 2011

NON-LEAP YEAR.

**Discussions**:

We input a year by the command line argument and using the following commands we checked that the year is LEAP YEAR or not. By **if** command we can perform conditional testing on strings, numbers and files, flag is an intermediate variable to store the medium value.……

**Program Statement:-** 10 . 11 . 2011

**Write a program to check for Palindrome string.**

**Program code:-**

echo -e "Enter the string:- \c"

read str

i=1

rev=""

while [ 1 ]

do

c=`echo $str|cut -c $i`

if [ -z "$c" ]

then

break

else

rev=$c$rev

i=`expr $i + 1`

fi

done

if [ $str = $rev ]

then

echo -e "PALINDROME"

else

echo -e "NOT PALINDROME"

fi

**Output**:

**Run 1:**

Enter the string:- madam

PALINDROME

**Run 2:**

Enter the string:- School

NOT PALINDROME

**Discussions**:

We 1st enter a string from keyboard and using the following commands we checked that the string is PALINDROME or not. The bash **while** loop is a control flow statement that allows code or commands to be executed repeatedly based on a given condition, By **if** command we can perform conditional testing on strings, numbers and files. **cut** is a [Unix](http://en.wikipedia.org/wiki/Unix) [command line](http://en.wikipedia.org/wiki/Command_line) utility which is used to extract sections from each line of input usually from a [file](http://en.wikipedia.org/wiki/Computer_file)..……

**Program Statement:-** 17 . 11 . 2011

**Write a program to find the Maximum and Minimum numbers in a given series of numbers.**

**Program code:-**

echo -e "Enter the number of terms: \c"

read n

echo -e "Enter the elements:- "

read a

max=$a

min=$a

for((i=1;i<n;i++))

do

read a

if [ $max -lt $a ]

then

max=$a

fi

if [ $min -gt $a ]

then

min=$a

fi

done

echo -e "The maximum element is $max"

echo -e "The minimum element is $min"

**Output:**

Enter the number of terms: 10

Enter the elements:-

2

4

6

8

10

34

24

67

61

45

The maximum element is 67

The minimum element is 2

**Discussions**:

At first we enter a range of no.s and then input some no.s. after that using the following commands we find the minimum and maximum of the no.s in a given range. Here **echo** command is used to echo argument , **read** command is used to input the value, The dollar (**$**) symbol is used in UNIX as an indicator for a variable there are reserved variables, **For** loop can be used where list is basically the list of the values u want to traverse on.……

**Program Statement:**- 24 . 11 . 2011

**Write a shell script to implement a file copy program. User give the source and destination filename as input.**

**Program code:-**

echo -e "Enter the source filename: \c"

read f1

echo -e "Enter the destination filename: \c"

read f2

cp $f1 $f2

echo -e "copy complete."

**Output:**

**Run 1:**

Enter the source filename: main

Enter the destination filename: Duplicate

copy complete.

**Run 2:**

Enter the source filename: factorial

Enter the destination filename: factt

copy complete.

**Discussions**:

We enter the source filename where the data being stored and then input the destination filename from which the data is stored after the operation. The operation is performed by the following commands. Here, **cp** command is used for copying files….. **Program Statement:-** 01 . 12 . 2011

**Find out the angstrom numbers between 1 to 500.**

**Program code:**-

echo -e "\n\t The angstrom numbers between 1 to 500 are....: "

for((no=1;no<=500;no++))

do

no1=$no

i=0

sum=0

while [ $no1 -ne 0 ]

do

no1=`expr $no1 / 10`

i=`expr $i + 1`

done

no1=$no

while [ $no1 -ne 0 ]

do

ld=`expr $no1 % 10`

j=$i

m=1

while [ $j -ne 0 ]

do

m=`expr $m \\* $ld`

j=`expr $j - 1`

done

sum=`expr $sum + $m`

no1=`expr $no1 / 10`

done

if [ $no -eq $sum ]

then

echo -e " $no"

fi

done

**Output**:

The angstrom numbers between 1 to 500 are....:

1

2

3

4

5

6

7

8

9

153

370

371

407

**Discussions:**

In the given program we write the program and the range of the no. is given from which we can calculate the Armstrong no. using the following commands. The Armstrong no.s is outputted from a given range. The bash **while** loop is a control flow statement that allows code or commands to be executed repeatedly based on a given condition, By **if** command we can perform conditional testing on strings, numbers and files. **For** loop can be used where list is basically the list of the values u want to traverse on.……

**Program Statement:-** 08 . 12 . 2011

**A file contains examination records in the following format:**

**Name subject 1 - - - - - - - - - - - - - - - - - - subject 5**

| | |

| | |

| | |

**Write a shell script to arrange the records according to the total marks of all subjects in descending order.**

**Program code:-**

echo -e "Enter the filename: \c"

read file

exec<$file

while read line

do

set -- `echo $line`

sum=`expr $2 + $3 + $4 + $5 + $6`

echo "$sum $line">>file1

done

sort -r file1>file2

exec<file2

while read line

do

set -- `echo $line`

echo "$2 $3 $4 $5 $6 $7">>file3

done

cp file3 $file

rm file1 file2 file3

**Output:**

Enter the file name: record.txt

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | MATH MARKS | PHYS MARKS | COMP MARKS |
| S.KUNDU | 82 | 56 | 91 |
| B .KAR | 56 | 45 | 87 |
| A .CHATTERJEE | 76 | 78 | 76 |
| D.ROY | 36 | 34 | 98 |
| S DAS | 67 | 65 | 65 |
| N GHOSH | 45 | 32 | 84 |
| P SARKAR | 55 | 43 | 71 |
| L SHARMA | 93 | 89 | 95 |

**Discussions:**

At first we enter the filename where the student’s record is stored in any manner using the following commands we can rearrange the file records in descending order. Here **read** command is used to input the value, The  **exec** command replaces the current shell process with the specified command, **rm** is used to remove a file. [**cp**](http://www.math.utah.edu/lab/unix/unix-commands.html#cp) command is used for copying files……

**Program Statement:-**  15 . 12 . 2011

**Write a program to find the file names with their size whose size is in the given range a-b.**

**Program code:-**

echo -e "Enter the value of a: \c"

read a

echo -e "Enter the value of b: \c"

read b

ls -l>file1

exec<file1

read line

echo -e ""

while read line

do

set -- `echo $line`

if [ $5 -ge $a -a $5 -le $b ]

then

echo -e "$5\t$9"

fi

done

rm file1

**Output:**

Enter a value of a:2

Enter a value of b:4

Apseries.txt, gpseries.txt, fibo.txt, prime.txt

**Discussions:**

Here we input a range, from this range we can show only that file which size is on the given range. This operation is performed by using the following commands. The bash **while** loop is a control flow statement that allows code or commands to be executed repeatedly based on a given condition, **rm** is used to remove a file….

**Program Statement:-** 22 . 12 . 2011

**Write a program to display the contents of a file in reverse order i.e. the first line becomes the last line, the second line becomes the second last line and so on with last line becoming the first line.**

**Program code:-**

echo -en "\n\n enter any text file name"

read file

rfl=""

t=`tty`

exec < $file

while read line

do

rfl=$line"\n"$rfl

done

exec < $t

echo -e "\n\n $rfl"

**Output:**

Enter any text file name apseries.txt

done

i=`expr $i + $d`

echo -e "$i"

do

while [ $i -le $n ]

i=$a

echo -e "The list is........"

read a

echo -e "Enter the 1st number->c"

read d

echo -e "Enter the common difference->c"

read n

echo -e "Enter the no of terms->c"

clear

**Discussions:**

We first input the text file name from the keyboard where the data is stored and using the following commands we can show then in the reverse order. Here **echo** command is used to echo argument , The dollar (**$**) symbol is used in UNIX as an indicator for a variable there are reserved variables, The bash **while** loop is a control flow statement that allows code or commands to be executed repeatedly based on a given condition, The  **exec** command replaces the current shell process with the specified command……

**Program Statement:-** 05 . 01 . 2012

**Write a program to compute the factorial of a number using recursive function.**

**Program code:-**

clear

factor()

{

if [ $n -ne 0 ]

then

f=`expr $f \echo is used to input from the keyboard read is used to input from the keyboard\* $n`

n=`expr $n - 1`

factor

fi

}

echo -en "enter a number"

read n

f=1

factor

echo "factorial is=$f"

**Output:**

Run 1:

Enter a number 4

Factorial is=24

Run 2:

Enter a number 6

Factorial is=720

**Discussions:**

At first we enter a number and using the following commands it can calculate the factorial of that number. Here use the **expr** command to perform arithmetic or string manipulation and making comparisons, By **if** command we can perform conditional testing on strings, numbers and files, and fact is a function to calculate the factorial of a number..……

**Program Statement:-** 12 . 01 . 2012

**Write a program to convert a number from any base to any base where then input base and the output base of the number are input from the keyboard.**

**Program code:-**

echo -en "\n enter the input base"

read i

echo -en "\n enter the output base"

read a

echo "obase=$a" > temp

echo "ibase=$i" >> temp

echo en "\n enter the number"

read n

n=`echo $n|tr "[ a - f ]" "[ A - F ]"`

echo $n >> temp

bc < temp > temp2

echo -e "\n the converted no. is:"

cat temp2

rm temp

rm temp2

**Output:**

**Run 1:**

enter the input base2

enter the output base10

enter the number

10100011

the converted no. is:

163

**Run 2:**

enter the input base8

enter the output base16

enter the number

764

the converted no. is:

1F4

**Discussions:**

Numbers are from different base in the computer arithmetic so that to perform the following operation and using the following commands we input two base at which the number is presently in and at which the number is to be converted. Here **clear** is used to clear the output screen, **echo** command is used to echo argument , **read** command is used to input the value, The dollar (**$**) symbol is used in UNIX as an indicator for a variable there are reserved variables, [**cat**](http://www.math.utah.edu/lab/unix/unix-commands.html#cat) command is used for creating and displaying short files, **rm** is used to remove a file .……

**Program Statement:-** 19 . 01 . 2012

**Write a program that adds “<<” at the beginning and “>>” at the end of each line of a file. The name of the file is accepted as a command line argument.**

**Program code:-**

t=`tty`

exec < $1

while read line

do

echo "<<" $line ">>" >> file1

done

mv file1 $1

cat $1

exec < $t

**Output**:

[sandip@localhost ~]$ sh ffile.sh apseries.sh

<< clear >>

<< echo -e "Enter the no of terms->c" >>

<< read n >>

<< echo -e "Enter the common difference->c" >>

<< read d >>

<< echo -e "Enter the 1st number->c" >>

<< read a >>

<< echo -e "The list is........" >>

<< i=$a >>

<< while [ $i -le $n ] >>

<< do >>

<< echo -e "$i" >>

<< i=`expr $i + $d` >>

<< done >>

**Discussions:**

The bash **while** loop is a control flow statement that allows code or commands to be executed repeatedly based on a given condition, The **exec** command replaces the current shell process with the specified command, move and rename files by using the **mv** command under UNIX. [**cat**](http://www.math.utah.edu/lab/unix/unix-commands.html#cat)  command is used for creating and displaying short files.……