1. While true

Do

Echo “implementation of 10 unix command”

Echo “1. For listing”

Echo “2. Display”

Echo “3. Calender”

Echo “4. Date”

Echo “5. Tty”

Echo “6. User name”

Echo “7. Who”

Echo “8. Time taken by process”

Echo “9. Make a directory”

Echo “10. Find file”

Echo “exit”

Echo “enter your choice”

Read choice

Case $choice in

1)echo “list of the file using commands”

Ls -l;;

2)echo “echo is used as display command” ;;

3)cal;;

4)date;;

5)tty;;

6)uname;;

7)who;;

8)time sleep 2;;

9)echo “enter diectory name”

Read name

Ls -l;;

10)echo “enter a file”

Read file

Find $file;;

\*)exit ;;

Esac

Done

~

**2) Write a shell script that displays list of all the files in the current directory to which the user has read, write and execute permissions.**

# script to print all the files with rwx permissions

Echo “The list of files with read, write and execute permission”

For file in \*

Do

If [ -f $file ]

Then

If [ -r $file -a -w $file -a -x $file ]

Then

Ls -l $file

Fi

Fi

Done

**3) Write a shell script that accepts a list of file names as its arguments, count and reports the occurrence of each word that is present in the first argument file on other argument files.**

# ‘script to count the occurance of words present in first file compared to

# that of other files’

Clear

Echo “Enter the number of files:”

Read n

Echo “Enter the file names:”

Read file

Set $file

For I in `cat $1`

Do

Echo “word= $i”

Echo “-----------”

For j in $@

Do

Echo “$j = `grep -o -I “$i” $j | wc -l`”

Done

Echo “-----------”

Done

**4) Write a shell script that accepts one or more file name as arguments and converts all of them to uppercase, provided they exist in the current directory.**

Echo “Enter the file name:”

Read file

For x in $file

Do

If [ ! -f $x ]

Then

Echo “file not found!”

Continue

Fi

Tr ‘[a-z]’ ‘[A-Z]’ < $x

Done

**5) Write grep commands to the following:**

a. To select the lines from a file that has exactly 2 characters.

b. To select the lines from a file that has more than 2 blank spaces.

# file that has exectly 2 char, line that has more than two spaces

While true

Do

Echo “enter the filename”

Read file

Echo “1. To select the lines from a file that has exactly 2 char”

Echo “2. To select the lines from a file that has more than 2 spaces”

Echo “3.exit”

Echo “Enter your choice”

Read ch

Case $ch in

1. Echo “lines that have only 2 char”

Grep -n ‘\<..\>’ $file;;

1. Echo “lines that has more than 2 spaces are”

Grep ‘[[:space:]]\{2,\}’ $file;;

\*) exit;;

Esac

Done

**6) Write a shell script which accepts two file names as arguments. Compare the contents. If they are same, then delete the second file.**

# script to accept two file names as arguments. Compare the contents. If same then delete the second file

Echo “Enter the first filename:”

Read file1

Echo “Enter the second filename:”

Read file2

If [ -f $file1 ]

Then

If cmp $file1 $file2

Then

Echo “The contents are same. Deleting the second file”

Rm $file2

Else

Echo “The contents are different”

Fi

Else

Echo “File/s not found”

Fi

**7) Write a shell script**

**a. to count number of lines in a file that do not contain vowels.**

**b. to count number of characters, words, lines in a given file.**

# script to count the number of lines without vowels in a file

Echo “Enter the filename:”

Read file

Count=0

Awk ‘$0!~/[aeiou]/{ count++ }END{printf “The number of lines without vowels are: %d\n”,count}’ $file

# script to count the number of characters, words, lines in a given file

Echo “Enter the filename:”

Read file

Chars=`cat $file | wc -c`

Words=`cat $file | wc -w`

Lines=`grep -c “.” $file`

Echo “The file has $chars characters, $words words and $lines lines.”

**8) Write a shell script to list all the files in a given directory.**

# script to list all the files of a given directory

Echo “Enter the directory name:”

Read dir

For file in “${dir}/”\*

Do

Echo $file

Done

**9) Write a shell script to display list of users currently logged in.**

Echo “List of users currently logged in are:”

Who

**10) Write a shell script to read three text files in the current directory and merge them into a single file and returns a file descriptor for the new file.**

# script to read three text files in the current directory and merge them into a single file and return the file descriptor

If [ $# -eq 3 ]

Then

Cat $1 $2 $3 > merged.txt

Echo “Files merged and named merged.txt”

Cat merged.txt

Else

Echo “Supply exactly three files”

Fi

**PART B**

1. **Write a program to copy a file into another using system calls.**

#include <stdio.h>

#include <unistd.h>

#include <fcntl.h>

Void main()

{

Char buf;

Int fd\_one, fd\_two;

Fd\_one = open(“first\_file”, O\_RDONLY);

If (fd\_one == -1)

{

Printf(“Error opening first\_file\n”);

Close(fd\_one);

Return;

}

Fd\_two = open(“second\_file”,

O\_WRONLY | O\_CREAT,

S\_IRUSR | S\_IWUSR | S\_IRGRP | S\_IROTH);

While(read(fd\_one, &buf, 1))

{

Write(fd\_two, &buf, 1);

}

Printf(“Successful copy”);

Close(fd\_one);

Close(fd\_two);

}

[Compling C program]

Cc filecopy.c

[display the content of first file]

Cat first\_file

Hello world

Bye

[executing c program]

./a.exe

Successful copy

[displaying output file]

Cat second\_file

Hello world

Bye

1. **Write a c program using system call to create, open , write, close, stat,fstat, lseek**

#include <stdio.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/stat.h>

Char buf1[]=”LAB “;

Char buf2[]=”OS Linux”;

Int main( void)

{

Int fd;

If ((fd=creat(“file.gol”, 0666)) < 0) {

Perror(“Creation error”);

      Exit (1);

}

If (write(fd, buf1, sizeof(buf1)) < 0)

            Perror(“Writing error”);

            Exit(2);

}

If (lseek(fd, 4096, SEEK\_SET) < 0)

            Perror(“Positioning error”);

            Exit(3);

}

If (write(fd, buf2, sizeof(buf2)) < 0)

            Perror(“Writing error”);

            Exit(2);

}

}

1. **Write a program to create a child process and allow the parent to display parent and the child to display child on the screen**

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

Int main()

{

Pid\_t p;

Printf(“before fork\n”);

P=fork();

If(p==0)

{

Printf(“I am child having id %d\n”,getpid());

Printf(“My parent’s id is %d\n”,getppid());

}

Else{

Printf(“My child’s id is %d\n”,p);

Printf(“I am parent having id %d\n”,getpid());

}

Printf(“Common\n”);

}

~

Cc newchild.c

./a.exe

Before fork

My child’s id is 1933

I am parent having id 1932

Common

I am child having id 1933

My parent’s id is 1

Common

1. **Write a program to create a zombie process**

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

Int main() {

Pid\_t child\_pid = fork(); // Fork returns process ID

If (child\_pid > 0) {

// Parent process

Sleep(5); // Parent sleeps for 50 seconds

} else {

// Child process

Exit(0); // Child exits

}

Return 0;

}

Cc newzobie.c

./a.exe

~

1. **Write a program to implement inter process communication using pipes**

#include<stdio.h>

#include<unistd.h>

Int main() {

Int pipefds[2];

Int returnstatus;

Char writemessages[2][20]={“Hi”, “Hello”};

Char readmessage[20];

Returnstatus = pipe(pipefds);

If (returnstatus == -1) {

Printf(“Unable to create pipe\n”);

Return 1;

}

Printf(“Writing to pipe – Message 1 is %s\n”, writemessages[0]);

Write(pipefds[1], writemessages[0], sizeof(writemessages[0]));

Read(pipefds[0], readmessage, sizeof(readmessage));

Printf(“Reading from pipe – Message 1 is %s\n”, readmessage);

Printf(“Writing to pipe – Message 2 is %s\n”, writemessages[0]);

Write(pipefds[1], writemessages[1], sizeof(writemessages[0]));

Read(pipefds[0], readmessage, sizeof(readmessage));

Printf(“Reading from pipe – Message 2 is %s\n”, readmessage);

Return 0;

}

Cc inprocss.c

./a.exe

Writing to pipe – Message 1 is Hi

Reading from pipe – Message 1 is Hi

Writing to pipe – Message 2 is Hi

Reading from pipe – Message 2 is Hello

**6. Write a program to simulate CPU Scheduling algorithm**

a) Round Robin b) SJF

#include<stdio.h>

Int main()

{

  Int count,j,n,time,remain,flag=0,time\_quantum;

  Int wait\_time=0,turnaround\_time=0,at[10],bt[10],rt[10];

  Printf(“Enter Total Process:\t “);

  Scanf(“%d”,&n);

  Remain=n;

  For(count=0;count<n;count++)

  {

    Printf(“Enter Arrival Time and Burst Time for Process Process Number %d :”,count+1);

    Scanf(“%d”,&at[count]);

    Scanf(“%d”,&bt[count]);

    Rt[count]=bt[count];

  }

  Printf(“Enter Time Quantum:\t”);

  Scanf(“%d”,&time\_quantum);

  Printf(“\n\nProcess\t|Turnaround Time|Waiting Time\n\n”);

  For(time=0,count=0;remain!=0;)

  {

    If(rt[count]<=time\_quantum && rt[count]>0)

    {

      Time+=rt[count];

      Rt[count]=0;

      Flag=1;

    }

    Else if(rt[count]>0)

    {

      Rt[count]-=time\_quantum;

      Time+=time\_quantum;

    }

If(rt[count]==0 && flag==1)

    {

      Remain--;

      Printf(“P[%d]\t|\t%d\t|\t%d\n”,count+1,time-at[count],time-at[count]-bt[count]);

      Wait\_time+=time-at[count]-bt[count];

      Turnaround\_time+=time-at[count];

      Flag=0;

    }

    If(count==n-1)

      Count=0;

    Else if(at[count+1]<=time)

      Count++;

    Else

      Count=0;

  }

  Printf(“\nAverage Waiting Time= %f\n”,wait\_time\*1.0/n);

Printf(“Avg Turnaround Time = %f”,turnaround\_time\*1.0/n);

  Return 0;

}

Cc robin.c

./a.exe

Enter Total Process: 3

Enter Arrival Time and Burst Time for Process Process Number 1 :0 1

Enter Arrival Time and Burst Time for Process Process Number 2 :0 3

Enter Arrival Time and Burst Time for Process Process Number 3 :0 4

Enter Time Quantum: 2

Process |Turnaround Time|Waiting Time

P[1] | 1 | 0

P[2] | 6 | 3

P[3] | 8 | 4

Average Waiting Time= 2.333333

Avg Turnaround Time = 5.000000