GOVERNMENT ENGINEERING COLLEGE , BILASPUR (C.G)

UNIX & SHELL PROGRAMMING LAB 322563(22)

SUBMITTED BY :

NAME :ROSHAN KUMAR
BRANCH : CSE 5Th
ROLL NO. : 300702218018

List of Experiment to be performed

1.a) Write a shell script that accepts a file name, starting and ending line numbers as arguments and displays all the lines between the given line numbers.

b) Write a shell script that deletes all lines containing a specified word in one or more files supplied as arguments to it.

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

b) Write a shell script that receives any number of file names as arguments checks if every argument supplied is a file or a directory and reports accordingly. Whenever the argument is a file, the number of lines on it is also reported.

3. a) Write a shell script to list all of the directory files in a directory.

b) Write a shell script to find factorial of a given integer.

4.a) Write an awk script to count the number of lines in a file that do not contain vowels.

b) Write an awk script to find the number of characters, words and lines in a file.

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

b) Write a c program that makes a copy of a file using standard I/O and system calls.

6. a) Implement in C the following Unix commands using System calls o cat , ls, mv

b) Write a C program to emulate the Unix ls –l command.

7. Write a program that takes one or more file/directory names as command line input and reports the following information on the file.

a) File type.	b) Number of links.
c) Time of last access.	d) Read, Write and Execute permissions.

8. a) Write a C program to list for every file in a directory, its inode number and file name.

b) Write a C program that demonstrates redirection of standard output to a file. Ex: ls > f1.

9.a) Write a C program to create a child process and allow the parent to display ―parent‖ and the child to display ―child‖ on the screen.

b) Write a C program that illustrates how to execute two commands concurrently with a command pipe. Ex:- ls –l | sort

10. a) Write a C program to create a Zombie process.

b) Write a C program that illustrates how an orphan is created.

Experiment No. :1 (a)

AIM: Write a Shell script that accepts a filename, starting and ending line numbers as arguments and displays all the lines between the given line numbers.

Solution:

```
echo "enter the filename"
read fname
echo "enter the starting line number"
read s
echo "enter the ending line number"
read n
sed -n $s,$n\p $fname | cat > newline
cat newline
```

OutPUT:

```
[root@localhost ~]# vi 1s.sh
[root@localhost ~]# ./1s.sh
bash: ./1s.sh: Permission denied
[root@localhost ~]# chmod 777 1s.sh
[root@localhost ~]# ./1s.sh
enter the filename
sales.dat
enter the starting line number
2
enter the ending line number
4
1       computers       9161
1       textbooks       21312           2       clothing        3252
```

Experiment No. :1 (b)

AIM: Write a shell script that deletes all lines containing a specified word in one or more files supplied as arguments to it.

```
## for this program we have to create one or more files (optional),
## I am creating two files names are del ,dell.

[root@localhost ~]# vi del
                    unix is os
                    dos is also os
                    here using unix
                    unix is powerful os
[root@localhost ~]# vi dell
                    Windows nt is also os
                    there are some difference between unix and windows nt
                    but unix is great among all os

## after creation two files now we have to write sed script file name is del.sed using vi edi
tor.

[root@localhost ~]# vi del.sed
{
 /os/d
}
```

OUTPUT:

```
[root@localhost ~]# sed -f del.sed del dell
   here using unix
    there are some difference between unix and windows nt
```

Experiment No. :2 (a)

AIM: 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.

```
echo "enter the directory name"
read dir
if [ -d $dir ]
        then
        cd $dir
        ls > f
        exec < f
                while read line
                do
                if [ -f $line ]
                then
                        if [ -r $line -a -w $line -a -x $line ]
                        then
                                echo "$line has all permissions"
                        else
                                echo "files not having all permissions"
                        fi
                fi
                done
fi
```

OUTPUT:

```
student@ubuntu:~$sh prg3.sh
enter the directory name
dir1
ff has all permissions
files not having permissions
```

Experiment No. :2 (b)

AIM: Write a Shell script that receives any number of file names as arguments checks if every argument supplied is a file or a directory and reports accordingly. Whenever the argument is a file, the number of lines on it is also reported.

```
for x in $*
        do
                if [ -f $x ]
                then
                echo " $x is a file "
                echo " no of lines in the file are "
                wc -l $x
                elif [ -d $x ]
                then
                echo " $x is a directory "
                else
                        echo " enter valid filename or directory name "
                fi
        done
```

OUTPUT:

```
guest-glcbIs@ubuntu:~$sh lprg4.sh dir1 d1
dir1 is a directory
d1 is a file
no of lines in the file are 2
```

Experiment No. :3(a)

AIM: Write a Shell script to list all of the directory files in a directory.

```
# !/bin/bash
echo "enter directory name"
read dir
          if[ -d $dir]
          then
                    echo "list of files in the directory"
                    ls –l $dir|egrep ‗^d‘
          else
                    echo "enter proper directory name"
          fi
```

OUTPUT:

```
guest-glcbIs@ubuntu:~$sh lprg6.sh
enter directory name
dir1
list of files in the directory
drwxrwxr-x 4 guest-glcbls guest-glcbls 140 2012-07-06 14:40 dir1
```

Experiment No. : 3 (b)

AIM: Write a Shell script to find factorial of a given integer.

```
# !/bin/bash
echo "enter a number"
read num
fact=1
        while [ $num -ge 1 ]
        do
                fact=`expr $fact\* $num`
                num=‘expr $num – 1‘
        done
echo "factorial of $n is $fact"
```

OUTPUT:

```
guest-glcbIs@ubuntu:~$sh lprg7.sh
enter a number
4
Factorial of 4 is 24
```

Experiment No. :4 (a)

AIM: Write an awk script to count lines without vowels in a file.

1st create a file test1.

```
[singh@00-13-02-56-15-7c programs]$ vi test1
engineering
data
and
lab
workshop
programming
rdx
bpb
hp
```

```
[singh@00-13-02-56-15-7c programs]$ vi raj11.sh
```

```
#!/bin/bash
echo "Enter file name"
read file
awk '$0!~/[aeiou]/{ count++ }
END{print "The number of lines that does not contain vowels are: ",count}' $file
```

OUTPUT :

```
[singh@00-13-02-56-15-7c programs]$ sh raj11.sh
Enter file name
test1
The number of lines that does not contain vowels are: 3
```

Experiment No. : 4 (b)

AIM: Write a awk script to find the number of characters, words and lines in a file.

```
BEGIN{print "record.\t characters \t words"}
#BODY section
{
        len=length($0)
        total_len =len
        print(NR,":\t",len,":\t",NF,$0)
        words =NF
}
END{
        print("\n total")
        print("characters :\t" total len)
        print("lines :\t" NR)
}
```

OUTPUT:

```
Student@ubuntu:~$ awk –f cnt.awk ff1
Record  words
1:      5:      1hello
Total
Characters:5
Lines:1
```

Experiment No. : 5 (a)

AIM: Write a Shell script that accepts a list of file names as its arguments, counts and reports the occurrence of each word that is present in the first argument file on other argument files.

```
if [ $# -eq 0 ]
then
          echo "no arguments"
else
          tr " " " " < $1 > temp
          shift
          for i in $*
          do
          tr " " " " < $i > temp1
          y=`wc -l < temp`
          j=1
                    while [ $j -le $y ]
                    do
                    x=`head -n $j temp | tail -1`
                    c=`grep -c "$x" temp1`
                    echo $x $c
                    j=`expr $j 1`
                    done
          done
fi
```

OUTPUT:

```
$sh 9a.sh hegde.sh ravi.sh
Raghu 2
Hary 1
Vinay 9
```

Experiment No. : 5 (b)

AIM: Write a C Program that makes a copy of a file using standard I/O and system calls.

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
void typefile (char *filename)
{
        int fd, nread;
        char buf[1024];
        fd = open (filename, O_RDONLY);
        if (fd == -1) {
                perror (filename);
        return;
        }
        while ((nread = read (fd, buf, sizeof (buf))) > 0)
        write (1, buf, nread);
        close (fd);
        }
int main (int argc, char **argv)
{
        int argno;
        for (argno = 1; argno < argc; argno )
        typefile (argv[argno]);
        exit (0);
}
```

OUTPUT:

```
student@ubuntu:~$gcc –o prg10.out prg10.c
student@ubuntu:~$cat > ff
hello
hai
```

```
student@ubuntu:~$./prg10.out ff
hello
hai
```

Experiment No. :6 (a)

AIM: Implement in C the following Unix commands using system calls

A). cat B). ls C). mv

A) cat

```
#include<sys/types.h>
#include<sys/stat.h>
#include<stdio.h>
#include<fcntl.h>
main( int argc,char *argv[3] )
{
        int fd,i;
        char buf[2];
        fd=open(argv[1],O_RDONLY,0777);
        if(fd==-argc)
        {
                printf("file open error");
        }
        else
        {
                while((i=read(fd,buf,1))>0)
                {
                        printf("%c",buf[0]);
                }
        close(fd);
        }
}
```

OUTPUT:

```
student@ubuntu:~$gcc –o prgcat.out prgcat.c
student@ubuntu:~$cat > ff
hello
hai
```

```
student@ubuntu:~$./prgcat.out ff
hello
hai
```

B) ls

```
#include <sys/types.h>
#include <sys/dir.h>
#include <sys/param.h>
#include <stdio.h>
#define FALSE 0
#define TRUE 1
extern int alphasort();
char pathname[MAXPATHLEN];
main()  {
          int count,i;
          struct dirent **files;
          int file_select();
          if (getwd(pathname) == NULL )
          {
                    printf("Error getting pathn");
                    exit(0);
          }
          printf("Current Working Directory = %sn",pathname);
          count = scandir(pathname, &files, file_select, alphasort);
          if (count <= 0)
          {
                    printf("No files in this directoryn");
                    exit(0);
          }
                    printf("Number of files = %dn",count);
```

```
          for (i=1;i<count 1;  i)
          printf("%s \n",files[i-1]->d_name);
}
int file_select(struct direct  *entry)
{
          if ((strcmp(entry->d_name, ".") == 0) ||(strcmp(entry->d_name, "..") == 0))
          return (FALSE);
          else
          return (TRUE);
}
```

Output:

```
Student@ubuntu:~$ gcc list.c
Student@ubuntu:~$ ./a.out
Current working directory=/home/student/
Number of files=57
```

C) mv

```
#include<sys/types.h>
#include<sys/stat.h>
#include<stdio.h>
#include<fcntl.h>
main( int argc,char *argv[] )
{
          int i,fd1,fd2;
          char *file1,*file2,buf[2];
          file1=argv[1];
          file2=argv[2];
          printf("file1=%s file2=%s",file1,file2);
          fd1=open(file1,O_RDONLY,0777);
          fd2=creat(file2,0777);
          while(i=read(fd1,buf,1)>0)
                    write(fd2,buf,1);
                    remove(file1);
                    close(fd1);
                    close(fd2);
}
```

OUTPUT:

```
student@ubuntu:~$gcc –o mvp.out mvp.c
student@ubuntu:~$cat > ff
hello
hai
student@ubuntu:~$./mvp.out ff ff1
student@ubuntu:~$cat  ff
cat:ff:No such file or directory
student@ubuntu:~$cat ff1
hello
hai
```

Experiment No. :6 (b)

AIM: Write a C program to emulate the Unix ls-l command.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>
int main()
{
        int pid;        //process id
        pid = fork();   //create another process
        if ( pid < 0 )
        {                       //fail
                printf(–\nFork failed\n‖); exit
                (-1);
        }
        else if ( pid == 0 )
        {               //child
                execlp ( –/bin/ls‖, –ls‖, –-l‖, NULL );  //execute ls
        }
        else
        {                       //parent
                wait (NULL);            //wait for child
                printf(–\nchild complete\n‖);
                exit (0);
        }
}
```

OUTPUT:

```
guest-glcbIs@ubuntu:~$gcc –o lsc.out lsc.c
guest-glcbIs@ubuntu:~$./lsc.out
total 100
```

```
-rwxrwx—x 1 guest-glcbls guest-glcbls 140 2012-07-06 14:55 f1
drwxrwxr-x 4 guest-glcbls guest-glcbls 140 2012-07-06 14:40 dir1
child complete
```

Experiment No. :7

AIM: Write a Program that takes one or more file/directory names as command line input and reports the following information on the file.

A). File type B). Number of links.

C).Time of last access. D).Read,Write and Execute permissions.

```
clear
for i in $*
do
        if [ -d $i ]
        then
                echo –Given directory name is found as $i‖
        fi
        if [ -f $i ]
        then
                echo –Given name is a file as $i –
        fi
                echo –Type of file/directory $i‖
        file $i
                echo –Last access time is:‖ ls
                -l$i | cut-c 31-46
                echo  "no.of links"
        ln $i
        if [ -x $i –a -w $i-a –r $i ]
        then
                echo –$i contains all permission‖
        else
                echo –$i does not contain all permissions‖
        fi
done
```

OUTPUT:

```
student@ubuntu:~$sh prg12.sh ff1
given name is file ff1
Type of file/directory ff1
last access time
2012-07-07 10:1
No.of links
ff1 does not contain all permissions
```

Experiment No. : 8 (a)

AIM: .Write a C program to list for every file in a directory,its inode number and file name.

```
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
main(int argc, char *argv[])
{
        char d[50];
        if(argc==2)
        {       bzero(d,sizeof(d));
                strcat(d,"ls ");
                strcat(d,"-i ");
                strcat(d,argv[1]);
                system(d);
        }
        else
                printf("\nInvalid No. of inputs");
}
```

OUTPUT:

```
student@ubuntu:~$ mkdir dd
student@ubuntu:~$ cd dd
student@ubuntu:~/dd$ cat >f1
hello
^z
student@ubuntu:~/dd$ cd
student@ubuntu:~$gcc –o flist.out flist.c
student@ubuntu:~$./flist.out dd
hello
46490 f1
```

Experiment No. :8 (b)

AIM: Write a C Program that demonstrates redirection of standard output to a file .EX:ls>f1.

```
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
main(int argc, char *argv[])
{
        char d[50];
        if(argc==2)
        {
                bzero(d,sizeof(d));
                strcat(d,"ls ");
                strcat(d,"> ");
                strcat(d,argv[1]);
                system(d);
        }
        else
                printf("\nInvalid No. of inputs");
}
```

OUTPUT:

```
student@ubuntu:~$ gcc –o std.out std.c
student@ubuntu:~$ls
downloads   documents   listing.c    listing.out       std.c   std.out
student@ubuntu:~$ cat > f1
^z
student@ubuntu:~$./std.out f1
student@ubuntu:~$cat f1
downloads
documents
listing.c
listing.out
```

```
std.c
std.out
```

Experiment No. : 9 (a)

AIM: Write a C 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 <sys/wait.h> /* contains prototype for wait */
int main(void)
{
        int pid;
        int status;
        printf("Hello World!\n");
        pid = fork( );
        if(pid == -1) /* check for error in fork */
        {
                perror("bad fork");
                exit(1);
        }
        if (pid == 0)
                printf("I am the child process.\n");
        else
        {
                wait(&status); /* parent waits for child to finish */
                printf("I am the parent process.\n");
        }
}
```

Output:

```
student@ubutnu:$gcc –o child.out child.c
student@ubutnu: ./child.out
Hello World!
I am the child process.
I am the parent process
```

Experiment No. :9 (b)

AIM: Write a program that illustrates how to execute two commands concurrently with a command pipe.

```
#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <stdlib.h>

int main()

{

        int pfds[2];

        char buf[30];

        if(pipe(pfds)==-1)

        {

                perror("pipe failed");

                exit(1);

        }

        if(!fork())

        {

        close(1);

        dup(pfds[1];

        system (—ls –l‖);

        }

        else

        {

                printf("parent reading from pipe \n");

                while(read(pfds[0],buf,80))

                printf("%s \n" ,buf);

        }

}
```

OUTPUT:

```
[student@gcet ~]$ vi pipes2.c
[student@gcet ~]$ cc pipes2.c
[student@gcet ~]$ ./a.out
Parent reading from pipe
Total 24
-rwxrwxr-x l student student 5563Aug 3 10:39 a.out
-rw-rw-r—l
Student student 340 jul 27 10:45 pipe2.c
-rw-rw-r—l student student
Pipes2.c
-rw-rw-r—l student student 401 34127 10:27 pipe2.c
student
```

Experiment No. : 10 (a)

AIM: Write a C program to create a Zombie process.

```
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
int main ()
{
  int pid_t child_pid;
  child_pid = fork ();
  if  (child_pid  >  0)
  { sleep (60);
  }
  else
    { exit
   (0);
  }
  return 0;
```

OUTPUT:

```
guest-glcbIs@ubuntu:~$gcc zombie.c
guest-glcbIs@ubuntu:~$./a.out
```

Then command prompt will wait for some time(60 sec) and then again command prompt will appear later.

Experiment No. : 10 (b)

AIM: Write a C program that illustrates how an orphan is created.

```
#include <stdio.h>
main()
{
int pid ;
printf("I'am the original process with PID %d and PPID %d.\n",getpid(),getppid());
pid=fork();
if(pid!=0 )
{
printf("I'am the parent with PID %d and PPID %d.\n",getpid(),getppid());
printf("My child's PID is %d\n",pid) ;
}
else
{
sleep(4);
printf("I'm the child with PID %d and PPID %d.\n",getpid(), getppid()) ;
}
printf ("PID %d terminates.\n", getpid()) ;
}
```

OUTPUT:

```
guest-glcbIs@ubuntu:~$gcc –o prg18.out prg18.c
guest-glcbIs@ubuntu:~$./prg18.out
I am the original process with PID2242 and PPID1677.
I am the parent with PID2242 and PPID1677
My child‘s PID is 2243
PID2243 terminates.
$ I am the child with PID2243 and PPID1.
PID2243 termanates.
```