**1)Write a C program that behaves like a shell (command interpreter). It has its own prompt say “NewShell$”. Any normal shell command is executed from your shell by starting a child process to execute the system program corresponding to the command. It should additionally interpret the following command.**

**count c - print number of characters in file**

**count w - print number of words in file**

**count l - print number of lines in file**

→Program

#include<stdio.h>

#include<stdlib.h>

char \*buff,\*t1,\*t2,\*t3,ch;

FILE \*fp;

int pid;

void count(char \*t2,char \*t3)

{

int charcount=0,wordcount=0,linecount=0;

if((fp=fopen(t3,"r"))==NULL)

printf("File not found");

else

{

if(strcmp(t2,"l")==0)

{

while((ch=fgetc(fp))!=EOF)

{

if(ch=='\n')

linecount++;

}

printf("The total no. of lines :%d\n",linecount);

}

else if(strcmp(t2,"w")==0)

{

while((ch=fgetc(fp))!=EOF)

{

if(ch=='\n'|| ch==' ')

wordcount++;

}

printf("The total no. of words :%d\n",wordcount);

}

else if(strcmp(t2,"c")==0)

{

while((ch=fgetc(fp))!=EOF)

{

if(ch!=' '&&ch!='\n')

charcount++;

}

printf("The total no. of characters :%d\n",charcount);

}

else

printf("Command not found");

}

fclose(fp);

}

main()

{

while(1)

{

printf("myshell$");

fflush(stdin);

t1=(char \*)malloc(80);

t2=(char \*)malloc(80);

t3=(char \*)malloc(80);

buff=(char \*)malloc(80);

fgets(buff,80,stdin);

sscanf(buff,"%s %s %s",t1,t2,t3);

if(strcmp(t1,"pause")==0)

exit(0);

else if(strcmp(t1,"count")==0)

count(t2,t3);

else

{

pid=fork();

if(pid<0)

printf("Child process is not created\n");

else if(pid==0)

{

if(strcmp(t1,"exit")==0)

exit(0);

}

wait(NULL);

exit(0);

}

}

}

Output

./a.out

myshell$count c a.txt

The total no. of characters :36

myshell$count w a.txt

The total no. of words :10

myshell$count l a.txt

The total no. of lines :4

myshell$ls

a.c a.out a.txt b.c count.c list.c search.c typeline.c

myshell$pause

**2)Write a C program that behaves like a shell (command interpreter). It has its own prompt say “NewShell$”. Any normal shell command is executed from your shell by starting a child process to execute the system program corresponding to the command. It should additionally interpret the following command.**

**i. list f - print name of all files in directory**

**ii. list n - print number of all entries**

**iii. list i - print name and inode of all files**

→Program

#include<stdio.h>

#include<stdlib.h>

#include<dirent.h>

char \*buff,\*t1,\*t2,\*t3,ch;

int pid;

void list(char t2,char \*t3)

{

DIR \*dir;

struct dirent \*entry;

int cnt=0;

dir=opendir(t3);

if (dir==NULL)

{

printf("Directory %s not found",t3);

return;

}

switch(t2)

{

case 'f': while((entry=readdir(dir))!=NULL)

{

printf("%s\n",entry->d\_name);

}

break;

case 'n': while((entry=readdir(dir))!=NULL)

cnt++;

printf("Total No of Entries: %d\n",cnt);

break;

case 'i': while((entry=readdir(dir))!=NULL)

{

printf("\n%s\t %d",entry->d\_name,entry->d\_ino);

}

break;

default : printf("Invalid argument");

}

closedir(dir);

}

main()

{

while(1)

{

printf("myshell$");

fflush(stdin);

t1=(char \*)malloc(80);

t2=(char \*)malloc(80);

t3=(char \*)malloc(80);

buff=(char \*)malloc(80);

fgets(buff,80,stdin);

sscanf(buff,"%s %s %s",t1,t2,t3);

if(strcmp(t1,"pause")==0)

exit(0);

else if(strcmp(t1,"list")==0)

list(t2[0],t3);

else

{

pid=fork();

if(pid<0)

printf("Child process is not created\n");

else if(pid==0)

{

if(strcmp(t1,"exit")==0)

exit(0);

}

wait(NULL);

exit(0);

}

}

}

Output:

cc shelldir.c

./a.out

myshell$list f program

inc.c

t5.c

dimen.c

char.c

test1.c

t2.c

relation.c

testp.c

.

f1.c

testp.c~

next.c

bit.c

sum.c

enum.c

test3.c

t1.c

test.c

ifelse.c

t4.c

assign.c

count.c

typecon.c

getchar.c

hex.c

basic.c

simple.c

enum1.c

f.c

..

unary.c

increment.c

example.c

myshell$list n program

Total No of Entries: 33

myshell$list i program

inc.c 442551

t5.c 411989

dimen.c 444245

char.c 440378

test1.c 442717

t2.c 406774

relation.c 439929

testp.c 411985

. 486779

f1.c 410969

testp.c~ 404883

next.c 440150

bit.c 444011

sum.c 411050

enum.c 410850

test3.c 411113

t1.c 411045

test.c 439975

ifelse.c 440597

t4.c 411987

assign.c 440112

count.c 445877

typecon.c 442564

getchar.c 445693

hex.c 443464

basic.c 440063

simple.c 439974

enum1.c 450844

f.c 411042

.. 482158

unary.c 442527

increment.c 439848

example.c 440686myshell$

myshell$pause

**3)Write a C program that behaves like a shell (command interpreter). It has its own prompt say “NewShell$”. Any normal shell command is executed from your shell by starting a child process to execute the system program corresponding to the command. It should additionally interpret the following command.**

**i. typeline +10 - print first 10 lines of file**

**ii. typeline -20 - print last 20 lines of file**

**iii. typeline a - print all lines of file**

→Program

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<string.h>

char \*buff,\*t1,\*t2,\*t3,ch;

FILE \*fp;

int pid;

void typeline(char \*t2,char \*t3)

{

int i,n,count=0,num;

if((fp=fopen(t3,"r"))==NULL)

printf("File not found\n");

if(strcmp(t2,"a")==0)

{

while((ch=fgetc(fp))!=EOF)

printf("%c",ch);

fclose(fp);

return;

}

n=atoi(t2);

if(n>0)

{

i=0;

while((ch=fgetc(fp))!=EOF)

{

if(ch=='\n')

{

i++;

if(i==n)

{

break;

}

}

printf("%c",ch);

}

printf("\n");

}

else

{

count=0;

while((ch=fgetc(fp))!=EOF)

if(ch=='\n')

count++;

fseek(fp,1,SEEK\_SET);

i=0;

while((ch=fgetc(fp))!=EOF)

{

if(ch=='\n')

i++;

if(i==count+n)

break;

}

while((ch=fgetc(fp))!=EOF)

printf("%c",ch);

}

fclose(fp);

}

main()

{

while(1)

{

printf("myshell$");

fflush(stdin);

t1=(char \*)malloc(80);

t2=(char \*)malloc(80);

t3=(char \*)malloc(80);

buff=(char \*)malloc(80);

fgets(buff,80,stdin);

sscanf(buff,"%s %s %s",t1,t2,t3);

if(strcmp(t1,"pause")==0)

exit(0);

else if(strcmp(t1,"typeline")==0)

typeline(t2,t3);

else

{

pid=fork();

if(pid<0)

printf("Child process is not created\n");

else if(pid==0)

{

if(strcmp(t1,"exit")==0)

exit(0);

}

wait(NULL);

exit(0);

}

}

}

Output:

cc shelltypeline.c

./a.out

myshell$typeline a f1.txt

123456

78910

1112

788O643

xcgcgc

6nbcgd

vb

vgfbhghgd

myshell$typeline -1 f1.txt

vgfbhghgd

myshell$typeline +3 f1.txt

123456

78910

1112

myshell$typeline +2 f1.txt

123456

78910

myshell$typeline -2 f1.txt

vb

vgfbhghgd

myshell$pause

**4)Write a C program that behaves like a shell (command interpreter). It has its own prompt say “NewShell$”.Any normal shell command is executed from your shell by starting a child process to execute the system program corresponding to the command. It should additionally interpret the following command.**

**i. search f - search first occurrence of pattern in filename**

**ii. search c - count no. of occurrences of pattern in filename**

**iii. search a - search all occurrences of pattern in filename**

→Program

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

char \*buff,\*t1,\*t2,\*t3,\*t4,ch;

FILE \*fp;

int pid;

void search(char \*t2,char \*t3,char \*t4)

{

int i=1,count=0;

char \*p;

if((fp=fopen(t4,"r"))==NULL)

printf("File not found\n");

else

{

if(strcmp(t2,"f")==0)

{

while(fgets(buff,80,fp))

{

if((strstr(buff,t3))!=NULL)

{

printf("%d: %s\n",i,buff);

break;

}

}

i++;

}

else if(strcmp(t2,"c")==0)

{

while(fgets(buff,80,fp))

{

if((strstr(buff,t3))!=NULL)

{

count++;

}

}

printf("No of occurences of %s= %d\n",t3,count);

}

else if(strcmp(t2,"a")==0)

{

while(fgets(buff,80,fp))

{

if((strstr(buff,t3))!=NULL)

{

printf("%d: %s\n",i,buff);

}

i++;

}

}

else

printf("Command not found\n");

fclose(fp);

}

}

main()

{

while(1)

{

printf("myshell$");

fflush(stdin);

t1=(char \*)malloc(80);

t2=(char \*)malloc(80);

t3=(char \*)malloc(80);

t4=(char \*)malloc(80);

buff=(char \*)malloc(80);

fgets(buff,80,stdin);

sscanf(buff,"%s %s %s %s",t1,t2,t3,t4);

if(strcmp(t1,"pause")==0)

exit(0);

else if(strcmp(t1,"search")==0)

search(t2,t3,t4);

else

{

pid=fork();

if(pid<0)

printf("Child process is not created\n");

else if(pid==0)

{

if(strcmp(t1,"exit")==0)

exit(0);

system(buff);

}

else

{

wait(NULL);

exit(0);

}

}

}

}

cc search.c

./a.out

myshell$search f aa s.txt

1: hello aa welcome

myshell$search c dyp s.txt

No of occurences of dyp= 3

myshell$search a dyp s.txt

2: dyp bb

3: tybcs aa dyp

5: dyp gm

myshell$ls

#a.c# a.out a.txt b.c count.c list.c search.c s.txt typeline.c

myshell$pause

**5)Write a program to create a file with hole in it**

#include<stdio.h>

#include<fcntl.h>

#include<string.h>

int main()

{

int n=creat("file.txt","w");

char ch[16]="hello world how are";

char str[20]="od -c file.txt";

system("chmod 777 file.txt");

write(n,ch,16);

lseek(n,48,SEEK\_SET);

write(n,ch,16);

system(str);

return(0);

}

Output:

[mcs11@localhost Desktop]$ ./a.out

0000000 h e l l o w o r l d h o w

0000020 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0

\*

0000060 h e l l o w o r l d h o w

0000100

**6)Write a program to print the size of the file.**

→Program

#include <stdio.h>

#include<fcntl.h>

int main()

{

int fd1,position;

fd1=open("a.txt",O\_RDONLY);

position=lseek(fd1,0,2);

printf("Size of file is %d bytes\n",position);

return 0;

}

Output:

cc cs.c

[mcs11@localhost Desktop]$ ./a.out

Size of file is 26 bytes

**7)Write a program to read the current directory and display the name of the files, no of files in current directory.**

#include<stdlib.h>

#include<stdio.h>

#include<string.h>

main(int argc, char \*argv[])

{

char d[50];

int cnt=0;

if(argc==2)

{

bzero(d,sizeof(d));

strcat(d,"ls ");

strcat(d,"-l ");

strcat(d,argv[1]);

system(d);

}

else

printf("\nInvalid No. of inputs");

}

OutPut:

/\*[root@localhost unix]# cc slip10.c

[root@localhost unix]# ./a.out rohini

total 4

-rw-r--r-- 1 root root 23 Feb 26 06:05 f1.txt\*/

**Q8)Write a C program to implement the following unix/linux command**

i. ls –l > output.txt \*/

/\*Write a C program to implement the following unix/linux command

i. ls –l > output.txt \*/

#include<sys/types.h>

#include<unistd.h>

#include<sys/stat.h>

main()

{

int pid=fork();

int fd=creat("output.txt",S\_IRWXU);

if(pid!=0 && pid!=-1)

{

close(1);

dup(fd);

execlp("ls","ls","-l",NULL);

close(fd);

}

}

Output:

cc lsoutput.c

output.txt

a.out

cprogram.c

file.txt

hole.c

hole.c~

lsoutput.c

lsoutput.c~

mode.c

output.txt

twowaypipe.c

twowaypipe.c~

**9)Takes multiple files as Command Line Arguments and print their inode number.**

#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/stat.h>// contains constructs that facilitate getting information about files attributes.

/\* \* \* get\_inode-returns the inode of the file associated

with the given file descriptor, or -1 on failure

a system struct that is defined to store information about files\*/

int get\_inode (int fd)

{

struct stat buf;

int ret;

ret = fstat(fd, &buf);

if ( ret <0 ) {

perror ("fstat");

return -1;

}

return buf.st\_ino;

}

int main (int argc, char \*argv[])

{

int fd, inode;

if ( argc <2) {

fprintf ( stderr, "usage: %s <file>\n", argv[0]) ;

return 1;

}

fd = open ( argv[1],O\_RDONLY);

if ( fd <0 ) {

perror ("open");

return 1;

}

inode = get\_inode (fd);

printf("%d\n",inode);

return 0;

}

output:

cc cs.c

[mcs11@localhost Desktop]$ ./a.out a.txt

203315983

**10)Write a program to print the mode of file where file name accepted through CommandLine.**

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <stdio.h>

#include <unistd.h>

#include <string.h>

int main(int argc, char \*argv[])

{

struct stat f\_stat;

if(argc!=2)

{

printf("pass invalid arguments\n");

return 1;

}

if(stat(argv[1],&f\_stat)==-1)

{

printf("Cannot read file %s:\n",argv[1]);

}

else

{

printf("Inode:%lu\n",f\_stat.st\_ino);

printf("File mode:%o\n",f\_stat.st\_mode);

}

}

Output:

[mcs11@localhost Desktop]$ ./a.out a.txt

Inode:203315983

File mode:100644