11. 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) count c <filename> - print number of characters in file

ii) count w <filename> - print number of words in file

iii) count l <filename> - print number of lines in file

#include<stdio.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<unistd.h>

#include<dirent.h>

#include<fcntl.h>

void count(char c, char \*fn)

{

int lc=0,wc=0,cc=0,handle;

char ch;

if((handle=open(fn,O\_RDONLY))==-1)

{

printf("File %s not found\n",fn);

return;

}

while(read(handle,&ch,1)!=0)

{

if(ch==' ')

wc++;

else if(ch=='\n')

{

lc++;

wc++;

}

else

cc++;

}

close(handle);

switch(c)

{

case 'c':

printf("Total No.of Characters = %d\n",cc);

break;

case 'w':

printf("Total No.of Words = %d\n",wc);

break;

case 'l':

printf("Total No.of Lines = %d\n",lc);

break;

}

}

main()

{

char command[80],t1[20],t2[20],t3[20],t4[20];

int n;

system("clear");

while(1)

{

printf("NewShell$");

/\* fflush(stdin) is used to flush or clear the output buffer of the stream.

When it is used after the scanf(), it flushes the input buffer also.

It returns zero if successful, otherwise returns EOF\*/

fflush(stdin);

fgets(command,80,stdin);

/\*sscanf () function is used to read formatted input from a string\*/

n = sscanf(command,"%s %s %s %s",t1,t2,t3,t4);

switch(n)

{

case 1:

if(fork()==0)

{

/\*execlp replaces the calling process image with a new process image.

This has the effect of running a new program with the process ID of the calling process. \*/

execlp(t1,t1,NULL);

perror(t1);

}

break;

case 2:

if(!fork())

{

execlp(t1,t1,t2,NULL);

perror(t1);

}

break;

case 3:

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

count(t2[0],t3);

else

{

if(!fork())

{

execlp(t1,t1,t2,t3,NULL);

perror(t1);

}

}

break;

case 4:

if(!fork())

{

execlp(t1,t1,t2,t3,t4,NULL);

perror(t1);

}

}

}

}