3. a) Write a C program that makes a copy of a file using standard I/O, and system calls

Program:

#include<stdio.h>

#include<unistd.h>

#include<fcntl.h>

void typefile(char \*ramjim)

{

int fd,nread;

char buf[1024];

fd=open(ramjim,O\_RDONLY);

if(fd==1)

{

perror(ramjim);

}

while((nread=read(fd,buf,sizeof(buf)))>0)

write(1,buf,nread);

close(fd);

}

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

{

int argno;

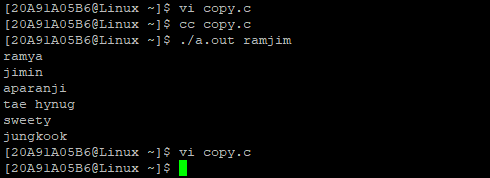
for(argno=1;argno<argc;argno++)

typefile(argv[argno]);

return 0;

}

Output:



3b) Write a C program to emulate the UNIX ls –l command

Program:

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<stdlib.h>

void main()

{

int pid;

pid=fork();

if(pid<0)

{

printf("fork failed");

exit(0);

}

else if(pid==0)

{

execlp("/bin/ls","ls","-1",NULL);

}

else

{

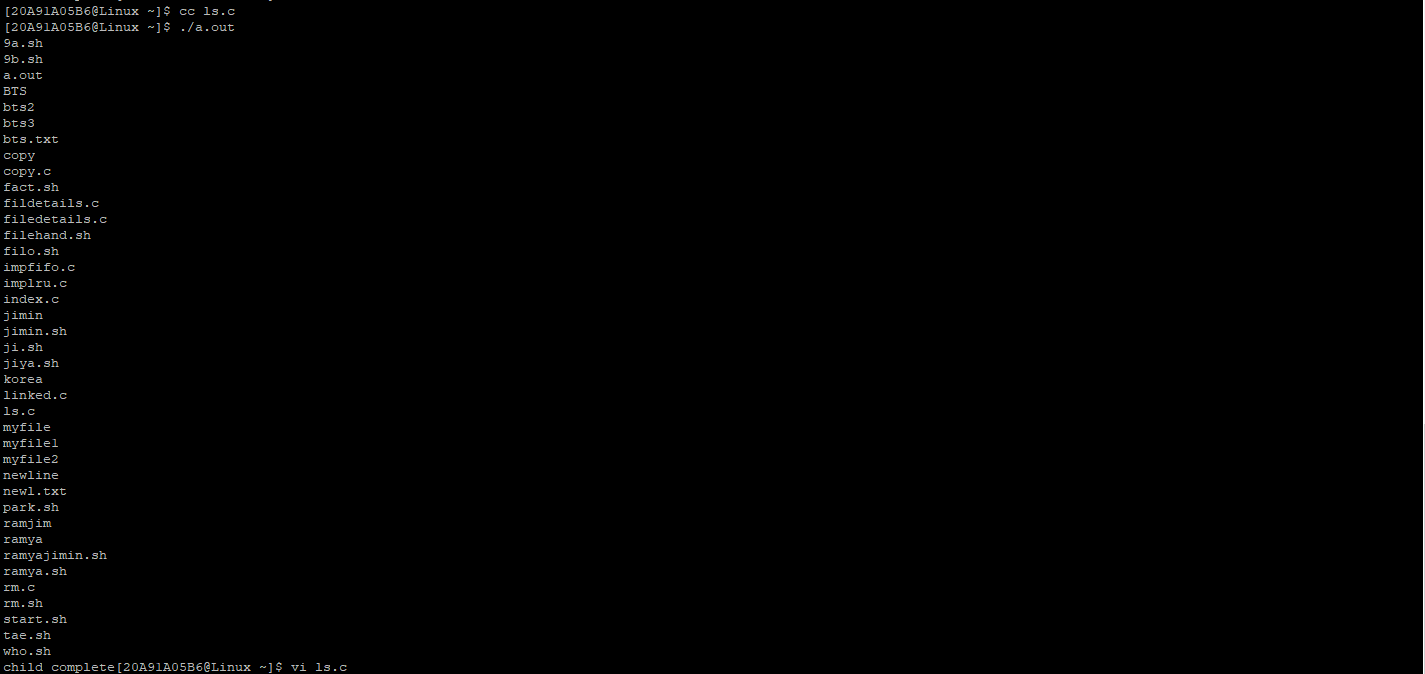
wait(NULL);

printf("child complete");

}

}

Output:



3c) Write a C program that illustrates how to execute two commands concurrently with a command pipe. Ex: - ls –l | sort.

Program:

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<stdlib.h>

void main()

{

int x[2];

char buf[30];

if(pipe(x)==-1)

{

perror("pipe failed");

exit(1);

}

if(!fork())

{

dup(x[1]);

system("ls -l");

}

else

printf("parent readingfrom pipe");

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

printf("%s",buf);

}

Output:

