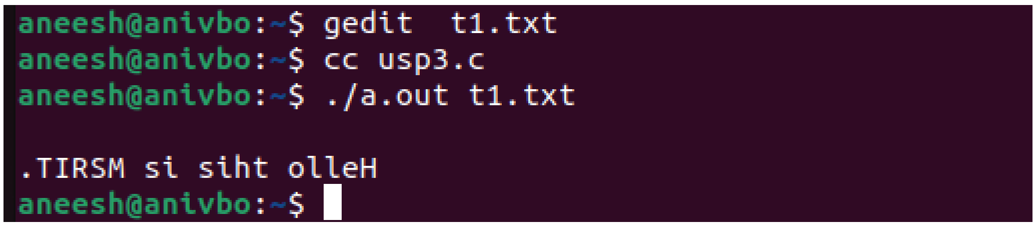
1.a) Write a C program to display the file content in reverse order using lseek system call.

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
int main(int argc, char *argv[]) {
    if (argc < 2) {
        fprintf(stderr, "Usage: %s <filename>\n", argv[0]);
        return 1;
    }
    int fd = open(argv[1], O_RDONLY);
    if (fd == -1) {
        perror("open");
        return 1;
    }
    int file_size = lseek(fd, 0, SEEK_END);
    if (file_size == -1) {
        perror("lseek");
        return 1;
    }
    for (int i = 1; i <= file_size; i++) {
        lseek(fd, -i, SEEK_END);
        char c;
        if (read(fd, &c, 1) != 1) {
            perror("read");
            return 1;
        }
        putchar(c);
    }
    printf("\n");
    close(fd);
    return 0;
}
```

Commands to Execute:

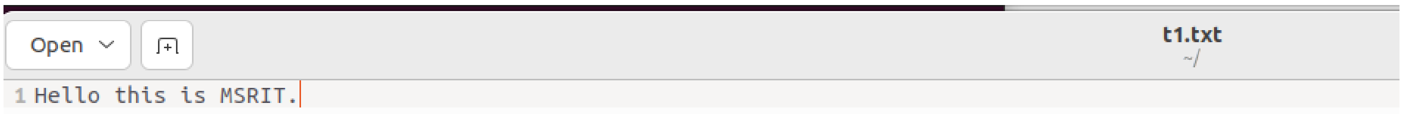
```
vi program_name.c (type program here)
vi file_name.txt (type some content here)
cc program_name.c
./a.out  <file_name>
```

Output:



```
aneesh@anivbo:~$ gedit  t1.txt
aneesh@anivbo:~$ cc usp3.c
aneesh@anivbo:~$ ./a.out t1.txt

.TIRSM si siht olleH
aneesh@anivbo:~$
```



```
Hello this is MSRIT.
```

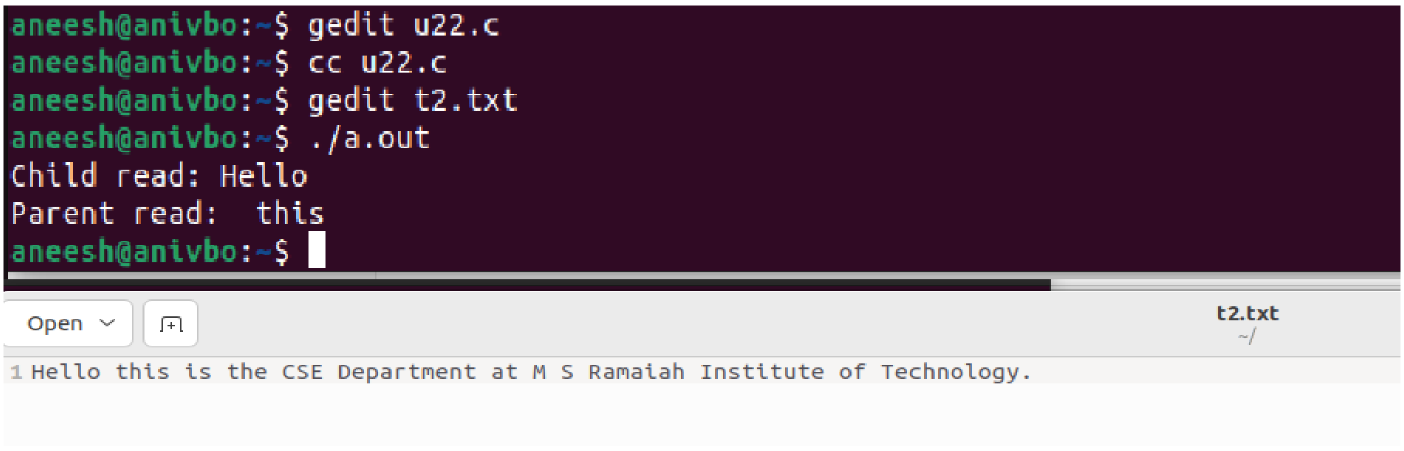
1.b) Write a C program to create a child process and show how parent and child processes will share the text file and justify that both parent and child shares the same file offset.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/wait.h>
int main() {
    int fd = open("t2.txt", O_RDWR);
    if (fd == -1) {
        perror("open");
        return 1;
    }
    pid_t pid = fork();
    if (pid == -1) {
        perror("fork");
        return 1;
    } else if (pid == 0) {
        char buffer[10];
        read(fd, buffer, 5);
        buffer[5] = '\0';
        printf("Child read: %s\n", buffer);
    } else {
        wait(NULL);
        char buffer[10];
        read(fd, buffer, 5);
        buffer[5] = '\0';
        printf("Parent read: %s\n", buffer);
    }
    close(fd);
    return 0;
}
```

Commands to Execute:

```
vi program_name.c (type program here)
vi file_name.txt (type some content here)
cc program_name.c
./a.out
```

Output:



```
aneesh@anivbo:~$ gedit u22.c
aneesh@anivbo:~$ cc u22.c
aneesh@anivbo:~$ gedit t2.txt
aneesh@anivbo:~$ ./a.out
Child read: Hello
Parent read:  this
aneesh@anivbo:~$
```

```
Hello this is the CSE Department at M S Ramaiah Institute of Technology.
```

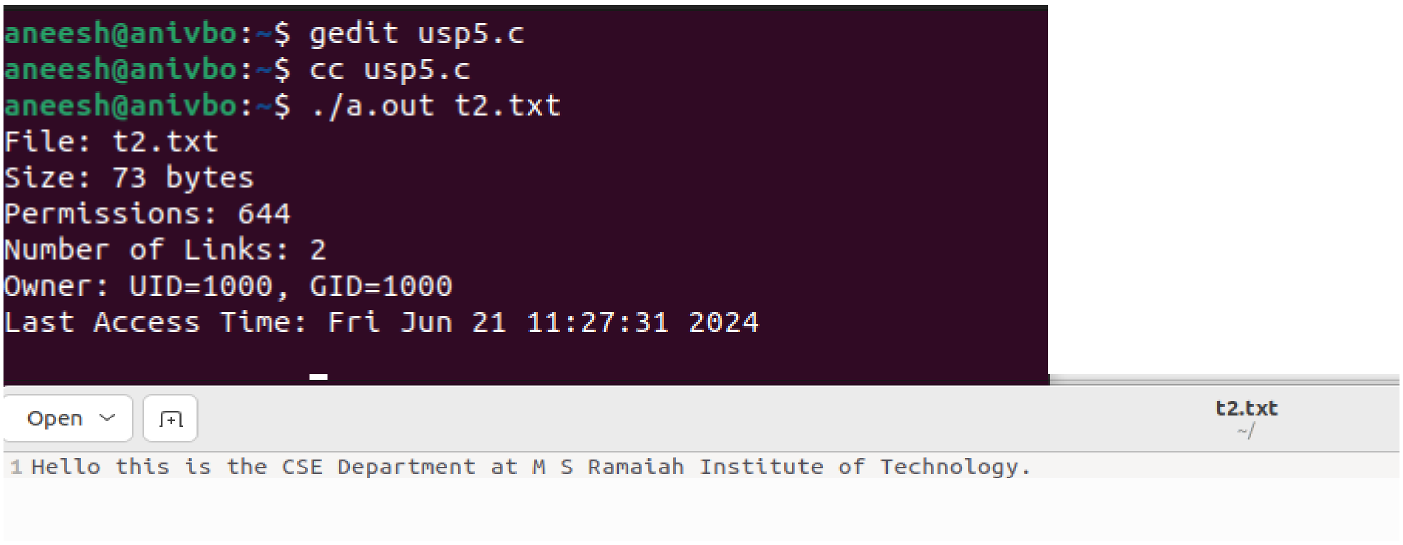
2.a) Write a C program to display various details of a file using stat structure (Atleast 5 fields).

```
#include <stdio.h>
#include <sys/stat.h>
int main(int argc, char *argv[]) {
    if (argc < 2) {
        fprintf(stderr, "Usage: %s <filename>\n", argv[0]);
        return 1;
    }
    struct stat file_stat;
    if (stat(argv[1], &file_stat) == -1) {
        perror("stat");
        return 1;
    }
    printf("File: %s\n", argv[1]);
    printf("Size: %lld bytes\n", (long long) file_stat.st_size);
    printf("Permissions: %o\n", file_stat.st_mode & 0777);
    printf("Number of Links: %ld\n", (long) file_stat.st_nlink);
    printf("Owner: UID=%ld, GID=%ld\n", (long) file_stat.st_uid, (long) file_stat.st_gid);
    printf("Last Access Time: %ld\n", (long) file_stat.st_atime);
    return 0;
}
```

Commands to Execute:

```
vi program_name.c (type program here)
vi file_name.txt (type some content here)
cc program_name.c
./a.out <file_name>
```

Output:



```
aneesh@anivbo:~$ gedit usp5.c
aneesh@anivbo:~$ cc usp5.c
aneesh@anivbo:~$ ./a.out t2.txt
File: t2.txt
Size: 73 bytes
Permissions: 644
Number of Links: 2
Owner: UID=1000, GID=1000
Last Access Time: Fri Jun 21 11:27:31 2024
```

```
Hello this is the CSE Department at M S Ramaiah Institute of Technology.
```

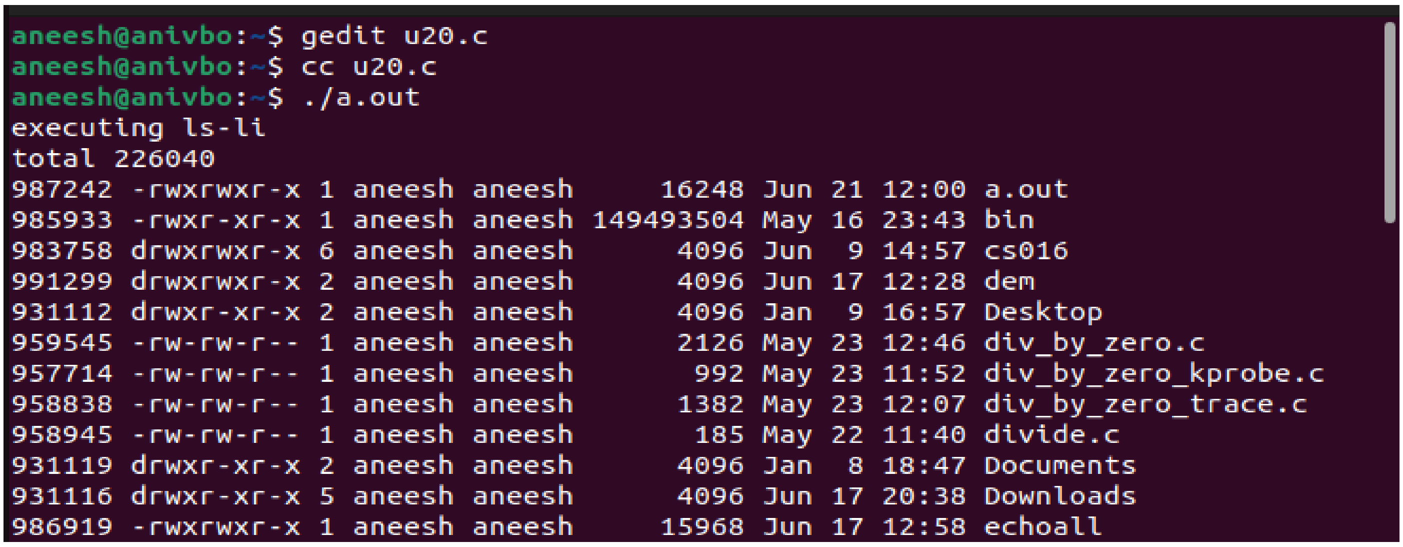
2.b) Write a C program to simulate system function.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/wait.h>
#include<sys/types.h>
int my_sys(const char *cm){
        if(cm==NULL){
                return -1;}
        pid_t pid=fork();
        if(pid==-1){
                printf("error\n");return -1;}
        else if(pid==0){
                execl("/bin/sh","sh","-c",cm,(char *)NULL);
                printf("execerror\n");
                exit(EXIT_FAILURE);
        }
        else{
                int st;
                if(waitpid(pid,&st,0)==-1){
                        return -1;}
                if(WIFEXITED(st)){
                        return WEXITSTATUS(st);
                }
                else{
                        return -1;}
                }
}
int main(){
printf("executing ls-li\n");
int res=my_sys("ls -li");
if(res==-1){
printf("error\n");
}
else{
printf("exited with status %d\n",res);
}
return 0;
}
```

Commands to Execute:

vi program_name.c (type program here)
cc program_name.c
./a.out

Output:



```
aneesh@anivbo:~$ gedit u20.c
aneesh@anivbo:~$ cc u20.c
aneesh@anivbo:~$ ./a.out
executing ls-li
total 226040
987242 -rwxrwxr-x 1 aneesh aneesh     16248 Jun 21 12:00 a.out
985933 -rwxr-xr-x 1 aneesh aneesh 149493504 May 16 23:43 bin
983758 drwxrwxr-x 6 aneesh aneesh      4096 Jun  9 14:57 cs016
991299 drwxrwxr-x 2 aneesh aneesh      4096 Jun 17 12:28 dem
931112 drwxr-xr-x 2 aneesh aneesh      4096 Jan  9 16:57 Desktop
959545 -rw-rw-r-- 1 aneesh aneesh      2126 May 23 12:46 div_by_zero.c
957714 -rw-rw-r-- 1 aneesh aneesh       992 May 23 11:52 div_by_zero_kprobe.c
958838 -rw-rw-r-- 1 aneesh aneesh      1382 May 23 12:07 div_by_zero_trace.c
958945 -rw-rw-r-- 1 aneesh aneesh       185 May 22 11:40 divide.c
931119 drwxr-xr-x 2 aneesh aneesh      4096 Jan  8 18:47 Documents
931116 drwxr-xr-x 5 aneesh aneesh      4096 Jun 17 20:38 Downloads
986919 -rwxrwxr-x 1 aneesh aneesh     15968 Jun 17 12:58 echoall
```

3.a) Write a C program to remove empty files from the given directory.

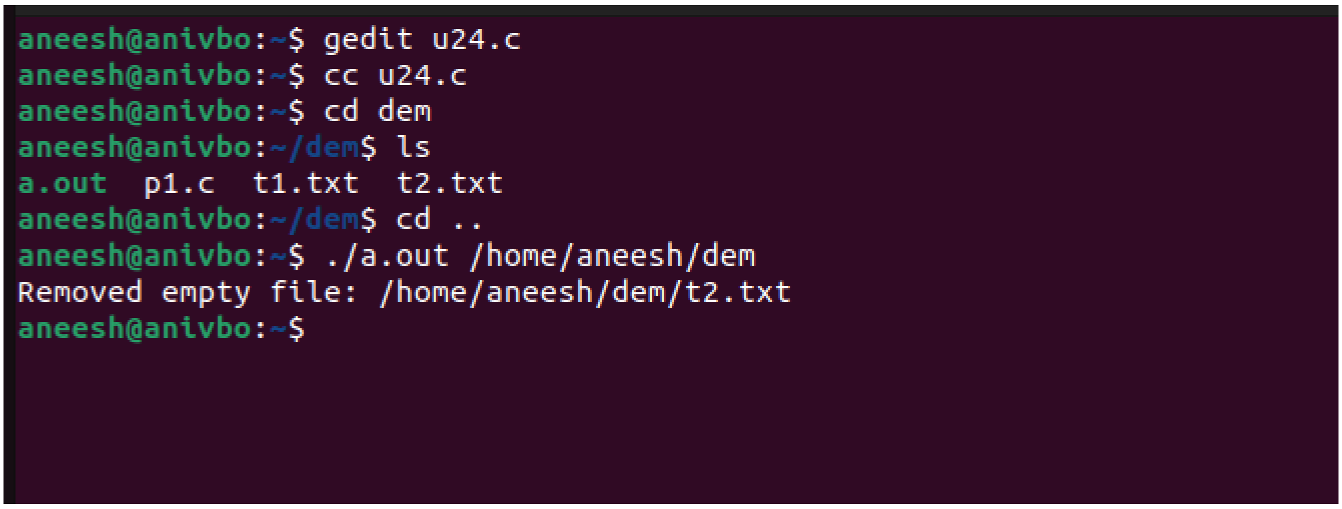
```
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
void remove_empty_files(const char *directory) {
    struct dirent *entry;
    DIR *dp = opendir(directory);
    if (dp == NULL) {
        perror("opendir");
        return;
    }
    while ((entry = readdir(dp)) != NULL) {
        if (strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") == 0) {
            continue;
        }
        char path[1024];
        snprintf(path, sizeof(path), "%s/%s", directory, entry->d_name);
        int fd = open(path, O_RDONLY);
        if (fd == -1) {
            perror("open");
            continue;
        }
        off_t size = lseek(fd, 0, SEEK_END);
        if (size == -1) {
            perror("lseek");
            close(fd);
            continue;
        }
        if (size == 0) {
            if (unlink(path) == 0) {
                printf("Removed empty file: %s\n", path);
            } else {
                perror("unlink");
            }
        }
    }
    closedir(dp);
}
int main(int argc,char *argv[]) {
    const char *directory = argv[1];
    remove_empty_files(directory);
    return 0;
}
```

Commands to Execute:

```
vi program_name.c
```

cc program_name.c
mkdir demo
cd demo
Inside demo directory, create an empty file
vi t1.txt
cd .. (come out of the directory)
./a.out <path_to_your_directory> (Ex. ./a.out /home/Aneesh/demo)

Output:



```
aneesh@anivbo:~$ gedit u24.c
aneesh@anivbo:~$ cc u24.c
aneesh@anivbo:~$ cd dem
aneesh@anivbo:~/dem$ ls
a.out  p1.c  t1.txt  t2.txt
aneesh@anivbo:~/dem$ cd ..
aneesh@anivbo:~$ ./a.out /home/aneesh/dem
Removed empty file: /home/aneesh/dem/t2.txt
aneesh@anivbo:~$
```

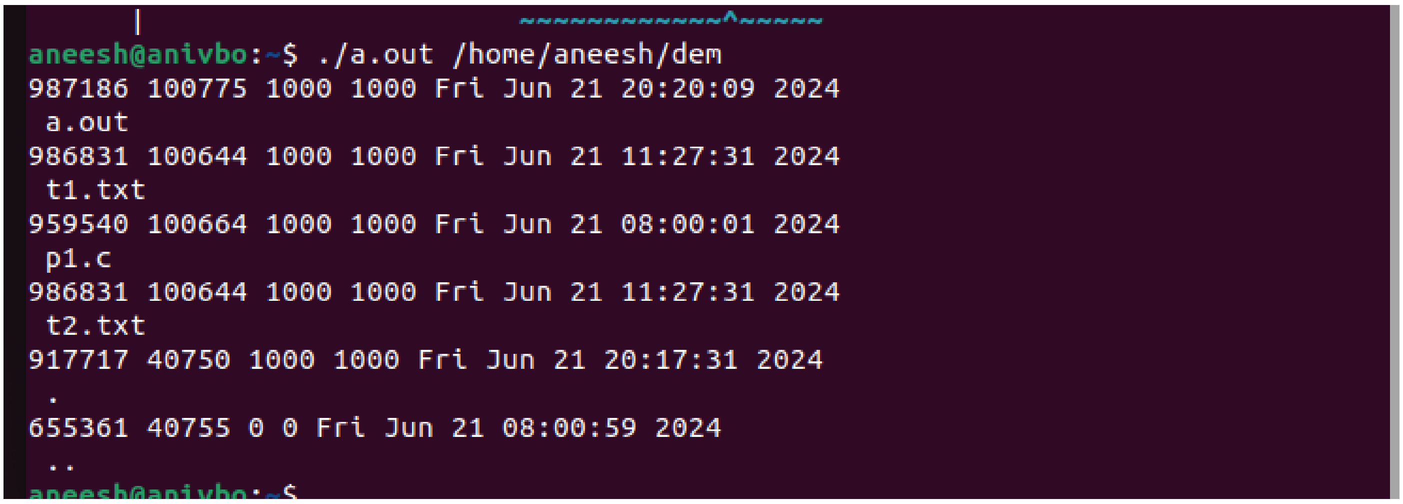
3.b) Write a C program to implement ls –li command which list the files in a specified directory. Your program should Print 5 attributes of files.

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <dirent.h>
#include <time.h>
#include<sys/stat.h>
int main(int argc,char *argv[]){
        struct dirent *d;
        struct stat m;
        DIR *dp=(argc>1) ? argv[1] : ".";
        dp = opendir(dp);
        if(dp){
                while(d = readdir(dp)){
                        stat(d->d_name,&m);
                        printf("%ld %o %d %d %s %s\n", m.st_ino, m.st_mode, m.st_uid, m.st_gid,
ctime(&m.st_atime),d->d_name);
  }
 }
}
```

Commands to Execute:

vi program_name.c (type program here)
cc program_name.c
./a.out <path of the directory> [Ex. /home/cs6a16/Downloads]

Output:



```
      |                         ~~~~~~~~~~~~^~~~~~
aneesh@anivbo:~$ ./a.out /home/aneesh/dem
987186 100775 1000 1000 Fri Jun 21 20:20:09 2024
 a.out
986831 100644 1000 1000 Fri Jun 21 11:27:31 2024
 t1.txt
959540 100664 1000 1000 Fri Jun 21 08:00:01 2024
 p1.c
986831 100644 1000 1000 Fri Jun 21 11:27:31 2024
 t2.txt
917717 40750 1000 1000 Fri Jun 21 20:17:31 2024
 .
655361 40755 0 0 Fri Jun 21 08:00:59 2024
 ..
aneesh@anivbo:~$
```

4.a) Write a C program to demonstrate the creation of soft links and the various properties of hard links.

```
#include<stdio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<unistd.h>
#include<sys/stat.h>
#include<sys/types.h>
int main(int argc,char *argv[]){
        if(argc==3){
                if((link(argv[1],argv[2]))==0){
                        printf("Hard link created\n");
                }
                else{
                        printf("Hard link error\n");
                }
        }
        else if(argc==4){
                if((symlink(argv[2],argv[3]))==0){
                        printf("Soft link created\n");
                }
                else{
                        printf("Soft link error\n");
                }
        }
        return 0;
}
```

Commands to Execute:

vi program_name.c (type program here)

cc program_name.c

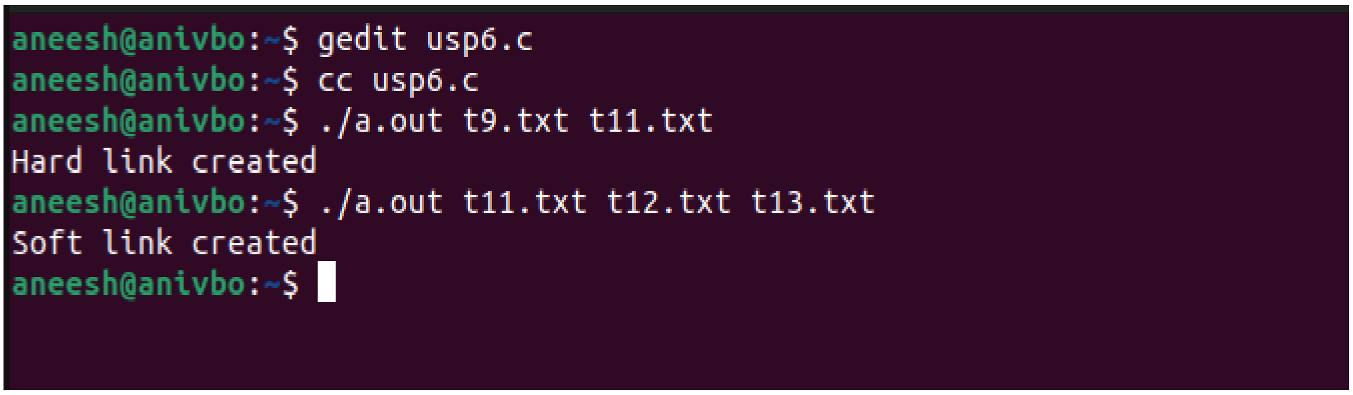
For Hard Link

./a.out <existing filename> <filename which isn't created>   (for ex if t1.txt is present , then ./a.out t1.txt t2.txt (t2.txt shouldn't be created using vi just pass it as name))

For Soft Link

./a.out <existing filename> <filename1 which isn’t created> <filename2 which isn’t created> (for ex if t1.txt is present , then ./a.out t1.txt t2.txt t3.txt (t2.txt,t3.txt shouldn’t be created using vi just pass them as names))

Output:



```
aneesh@anivbo:~$ gedit usp6.c
aneesh@anivbo:~$ cc usp6.c
aneesh@anivbo:~$ ./a.out t9.txt t11.txt
Hard link created
aneesh@anivbo:~$ ./a.out t11.txt t12.txt t13.txt
Soft link created
aneesh@anivbo:~$
```

4.b) Write a C program to

- **i. To create a child process.**
- **ii. The child should execute an interpreter file by passing a few arguments**
- **iii. Create an interpreter file that has the path of echoall.c file and pass one argument**
- **iv. Create echoall.c file which prints the arguments received from both child process and interpreter file.**

echoall.c

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc,char *argv[]){
        int i;
        for(i=0;i<argc;i++){
                printf("argv[%d]= %s\n",i,argv[i]);
        }
return 0;
}
```

inter.c

```
#include<stdio.h>
#include<sys/stat.h>
#include<sys/types.h>
#include<unistd.h>
#include<fcntl.h>
int main(){
        pid_t pid=fork();
        if(pid<0){
                printf("error\n");
        }
        else if(pid==0){
                if(execl("textinterpreter","test","myarg1","myarg2","myarg4",(char *)0)<0)
                        printf("error\n");
                        if(waitpid(pid,NULL,0)<0)
                                printf("error\n");
            }
return 0;
}
```

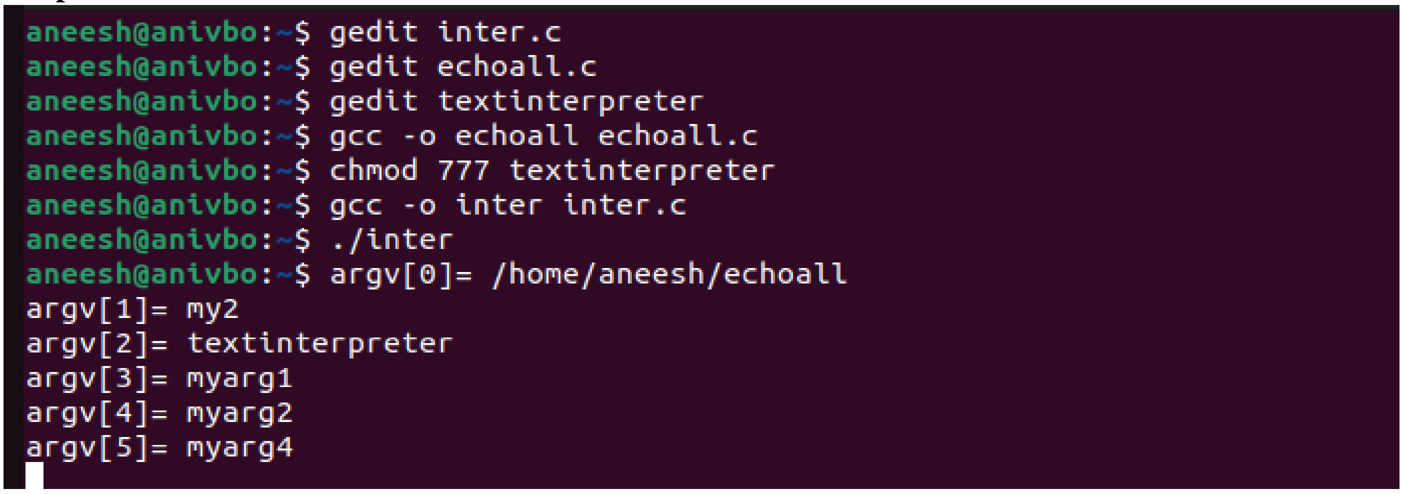
textinterpreter file

```
#! /home/<your name/computer name>/echoall my2
[Ex. #! /home/aneesh/echoall my2]
```

Commands to Execute:

```
gcc -o echoall echoall.c
chmod 777 textinterpreter
gcc -o inter inter.c
./inter
```

Output:



```
aneesh@anivbo:~$ gedit inter.c
aneesh@anivbo:~$ gedit echoall.c
aneesh@anivbo:~$ gedit textinterpreter
aneesh@anivbo:~$ gcc -o echoall echoall.c
aneesh@anivbo:~$ chmod 777 textinterpreter
aneesh@anivbo:~$ gcc -o inter inter.c
aneesh@anivbo:~$ ./inter
aneesh@anivbo:~$ argv[0]= /home/aneesh/echoall
argv[1]= my2
argv[2]= textinterpreter
argv[3]= myarg1
argv[4]= myarg2
argv[5]= myarg4
```

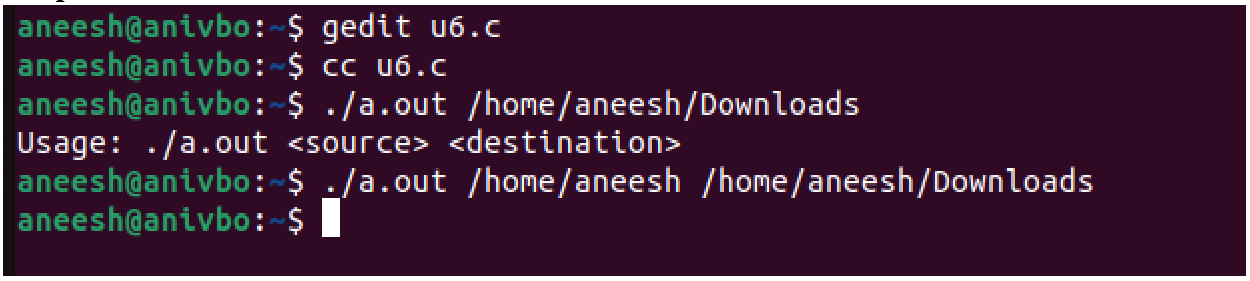
5.a) Write a program to copy access and modification time of a file to another file using utime function.

```
#include <stdio.h>
#include <utime.h>
#include <sys/stat.h>
int main(int argc, char *argv[]) {
    if (argc < 3) {
        fprintf(stderr, "Usage: %s <source> <destination>\n", argv[0]);
        return 1;
    }
    struct stat file_stat;
    if (stat(argv[1], &file_stat) == -1) {
        perror("stat");
        return 1;
    }
    struct utimbuf new_times;
    new_times.actime = file_stat.st_atime;
    new_times.modtime = file_stat.st_mtime;
    if (utime(argv[2], &new_times) == -1) {
        perror("utime");
        return 1;
    }
    return 0;
}
```

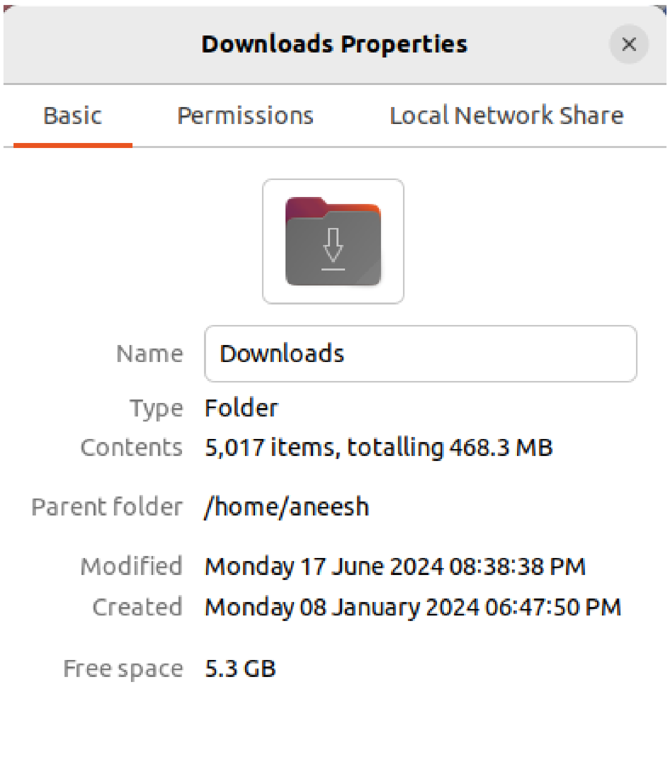
Commands to Execute:

vi program_name.c (type program here)
cc program_name.c
./a.out <source_path> <destination_path>
[Ex. ./a.out /home/aneesh /home/aneesh/Downloads]

Output:



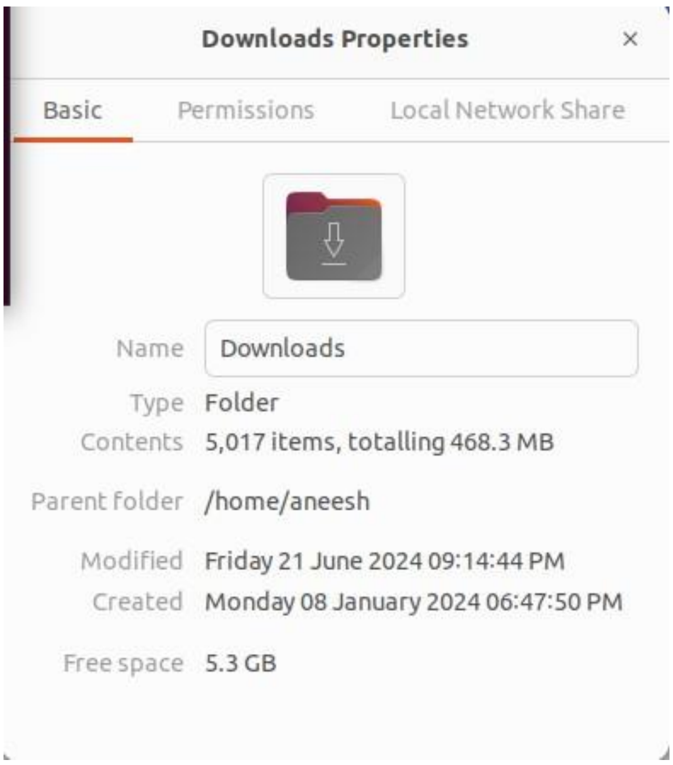
```
aneesh@anivbo:~$ gedit u6.c
aneesh@anivbo:~$ cc u6.c
aneesh@anivbo:~$ ./a.out /home/aneesh/Downloads
Usage: ./a.out <source> <destination>
aneesh@anivbo:~$ ./a.out /home/aneesh /home/aneesh/Downloads
aneesh@anivbo:~$ 
```



Downloads Properties

Name	Downloads
Type	Folder
Contents	5,017 items, totalling 468.3 MB
Parent folder	/home/aneesh
Modified	Monday 17 June 2024 08:38:38 PM
Created	Monday 08 January 2024 06:47:50 PM
Free space	5.3 GB

Time Before



Downloads Properties

Name	Downloads
Type	Folder
Contents	5,017 items, totalling 468.3 MB
Parent folder	/home/aneesh
Modified	Friday 21 June 2024 09:14:44 PM
Created	Monday 08 January 2024 06:47:50 PM
Free space	5.3 GB

Time After

5.b) Write a C program using sigaction system call which calls a signal handler on SIGINT signal and then reset the default action of the SIGINT signal.

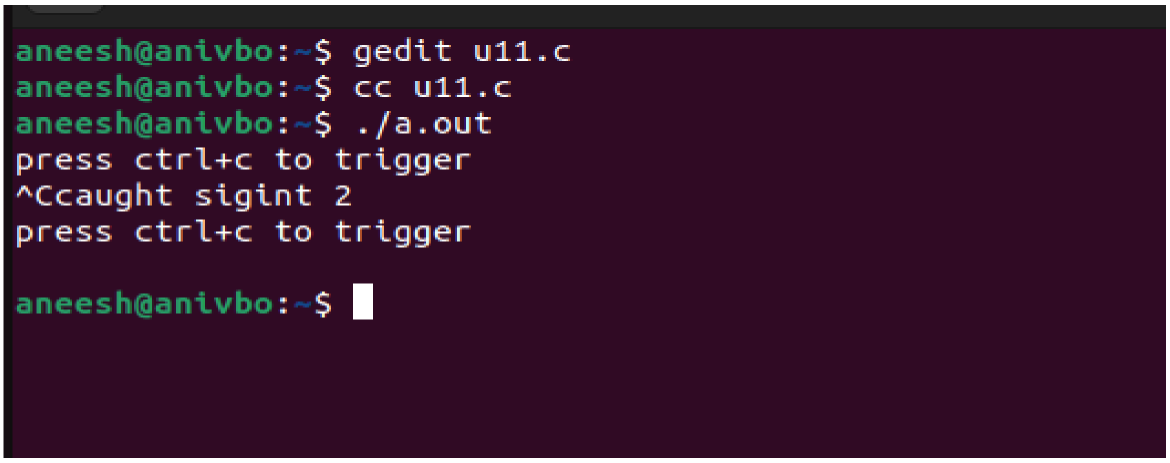
```
#include<stdio.h>
#include<stdlib.h>
#include<signal.h>
#include<unistd.h>
void s_h(int sn){
        printf("\ncaught sigint %d\n",sn);
        struct sigaction sa;
        sa.sa_handler=SIG_DFL;
        sa.sa_flags=0;
        sigemptyset(&sa.sa_mask);
        if(sigaction(SIGINT,&sa,NULL)==-1){
                printf("error\n");
                exit(EXIT_FAILURE);
        }
}
int main(){
        struct sigaction sa;
        sa.sa_handler=s_h;
        sa.sa_flags=0;
        sigemptyset(&sa.sa_mask);
        if(sigaction(SIGINT,&sa,NULL)==-1){
                printf("error\n");
                exit(EXIT_FAILURE);
        }
        while(1){
                printf("press ctrl+c to trigger\n");
                pause();
        }
return 0;
}
```

Commands to Execute:

```
vi program_name.c (type program here)
cc program_name.c
./a.out
```

Then press CTRL+C twice to show output

Output:



```
aneesh@anivbo:~$ gedit u11.c
aneesh@anivbo:~$ cc u11.c
aneesh@anivbo:~$ ./a.out
press ctrl+c to trigger
^Ccaught sigint 2
press ctrl+c to trigger

aneesh@anivbo:~$ 
```

6.a) Write a program to read n characters from a file and append them back to the same file using dup2 function.

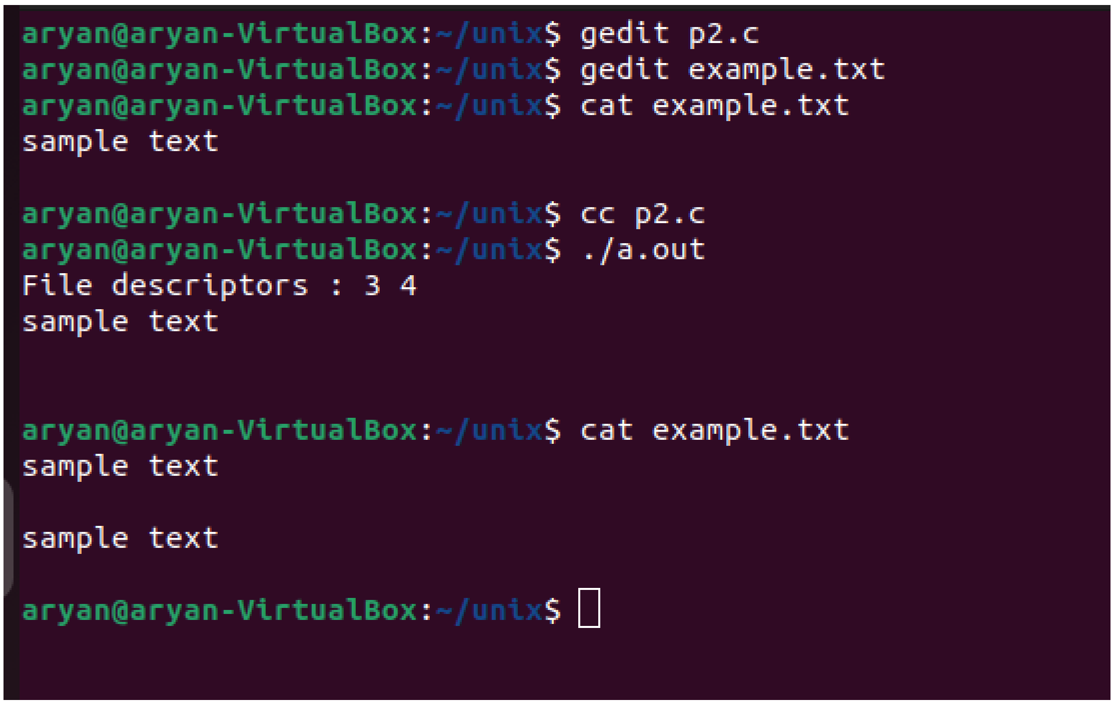
```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int main() {
   int fd1, fd2;
```

```
    char buf[50];
    fd1 = open("example.txt", O_RDWR, 0);
    fd2 = open("sample.txt", O_CREAT | O_RDWR, 0777);
    // Duplicate fd1 to fd2 using dup2
    fd2 = dup2(fd1, fd2);
    if (fd2 < 0) {
        printf("dup2 error\n");
        close(fd1);
        return 1;
    }
    printf("File descriptors: %d %d \n", fd1, fd2);
    if (read(fd1, buf, 20) < 0) {
        printf("read error\n");
        close(fd1);
        close(fd2);
        return 1;
    }
    if (lseek(fd2, 0, SEEK_END) < 0) {
        printf("lseek error\n");
        close(fd1);
        close(fd2);
        return 1;
    }
    if (write(fd2, buf, 20) < 0) {
        printf("write error\n");
        close(fd1);
        close(fd2);
        return 1;
    }
    printf("%s\n", buf);
    close(fd1);
    close(fd2);
    return 0;
}
```

Commands to Execute:

```
vi program_name.c (type program here)
vi file_name.txt
cat file_name.txt
cc program_name.c
./a.out
cat file_name.txt
```

Output:



```
aryan@aryan-VirtualBox:~/unix$ gedit p2.c
aryan@aryan-VirtualBox:~/unix$ gedit example.txt
aryan@aryan-VirtualBox:~/unix$ cat example.txt
sample text

aryan@aryan-VirtualBox:~/unix$ cc p2.c
aryan@aryan-VirtualBox:~/unix$ ./a.out
File descriptors : 3 4
sample text


aryan@aryan-VirtualBox:~/unix$ cat example.txt
sample text

sample text

aryan@aryan-VirtualBox:~/unix$
```

6.b) Consider the last 100 bytes as a region. Write a C program to check whether the region is locked or not. If the region is locked, print pid of the process which has locked. If the region is not locked, lock the region with an exclusive lock, read the last 50 bytes and unlock the region.

```
#include<stdio.h>
#include<stdlib.h>
#include<errno.h>
#include<fcntl.h>
#include<unistd.h>
int main(int argc,char *argv[]){
        int fd; char buf[255]; struct flock fv;
        if(argc<2){
                printf("usage %s <filename>\n",argv[0]);
                exit(0);
        }
        if((fd=open(argv[1],O_RDWR))==-1){
                printf("error\n");
                exit(1);
        }
        fv.l_type=F_WRLCK; fv.l_whence=SEEK_END;
        fv.l_start=SEEK_END-100; fv.l_len=100;
        printf("press enter to set lock\n");
        getchar();
        printf("trying to get lock\n");
        if((fcntl(fd,F_SETLK,&fv))==-1){
                fcntl(fd,F_GETLK,&fv);
                printf("file is locked by process pid: %d \n",fv.l_pid);
                return -1;
        }
        printf("locked\n");
        if((lseek(fd,SEEK_END-50,SEEK_END))==-1){
                printf("lseek\n");
                exit(1);
        }
        if((read(fd,buf,100))==-1){
                printf("read\n");
                exit(1);
        }
        printf("data from file:\n");
        puts(buf);
        printf("press enter to unlock\n");
        getchar();
        fv.l_type=F_UNLCK; fv.l_whence=SEEK_SET;
        fv.l_start=0; fv.l_len=0;
        if((fcntl(fd,F_UNLCK,&fv))==-1){
                printf("error\n");
                exit(0);
        }
        printf("unlocked\n");
        close(fd);
        return 0;
}
```

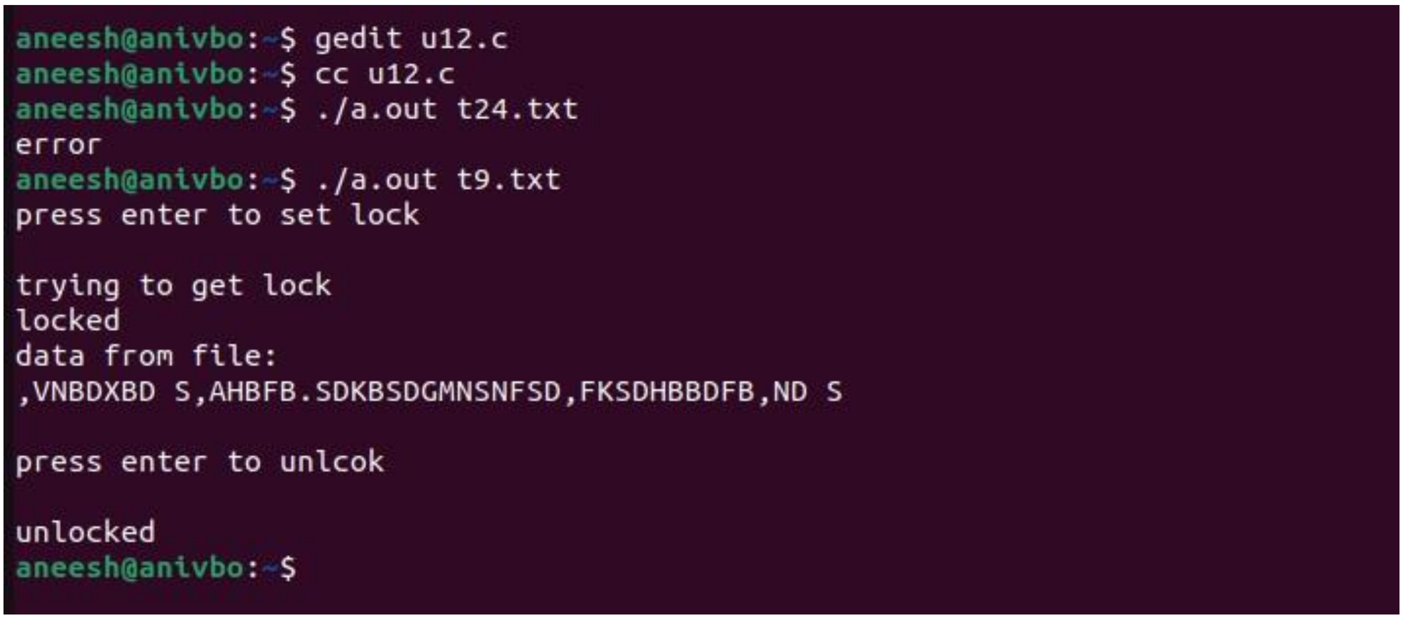
Commands to Execute:
Open Two Terminals
In one,

```
vi program_name.c(type your program here)
cc program_name.c
./a.out <file_name>(file should exist)
```

In Second,

```
cc program_name.c
./a.out <file_name>(file should exist)
```

Output:
First Terminal,



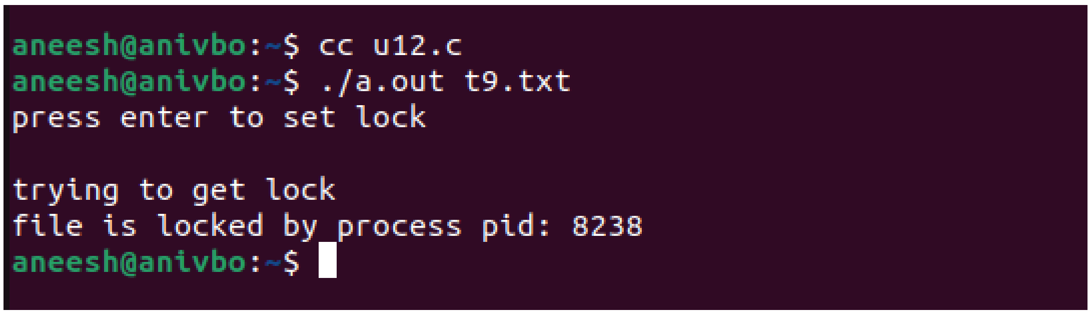
```
aneesh@anivbo:~$ gedit u12.c
aneesh@anivbo:~$ cc u12.c
aneesh@anivbo:~$ ./a.out t24.txt
error
aneesh@anivbo:~$ ./a.out t9.txt
press enter to set lock

trying to get lock
locked
data from file:
,VNBDXBD S,AHBFB.SDKBSDGMNSNFSD,FKSDHBBDFB,ND S

press enter to unlcok

unlocked
aneesh@anivbo:~$
```

Second Terminal,



```
aneesh@anivbo:~$ cc u12.c
aneesh@anivbo:~$ ./a.out t9.txt
press enter to set lock

trying to get lock
file is locked by process pid: 8238
aneesh@anivbo:~$
```

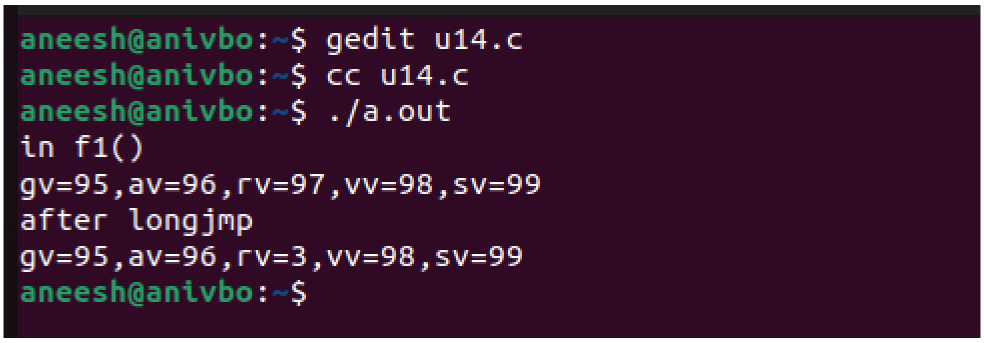
7.a) Write a C program to illustrate the effect of setjmp and longjmp functions on register and volatile variables.

```
#include<setjmp.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<fcntl.h>
static void f1(int,int,int,int);
static void f2(void);
static int gv;
static jmp_buf jb;
int main(void){
        int av=2;register int rv=3; volatile int vv=4;static int sv=5;gv=1;
        if(setjmp(jb)!=0){
                printf("after longjmp\n");
                printf("gv=%d,av=%d,rv=%d,vv=%d,sv=%d \n",gv,av,rv,vv,sv);
                exit(0);
        }
        gv=95;av=96;rv=97;vv=98;sv=99;
        f1(av,rv,vv,sv);
        exit(0);
}
static void f1(int i,int j,int k,int l){
        printf("in f1() \n");
        printf("gv=%d,av=%d,rv=%d,vv=%d,sv=%d \n",gv,i,j,k,l);
        f2();
}
static void f2(void){
        longjmp(jb,1);
}
```

Commands to Execute:

```
vi program_name.c (type program here)
cc program_name.c
./a.out
```

Output:



```
aneesh@anivbo:~$ gedit u14.c
aneesh@anivbo:~$ cc u14.c
aneesh@anivbo:~$ ./a.out
in f1()
gv=95,av=96,rv=97,vv=98,sv=99
after longjmp
gv=95,av=96,rv=3,vv=98,sv=99
aneesh@anivbo:~$
```

7.b) C program to simulate copy command by accepting the filenames from command line. Report all errors.

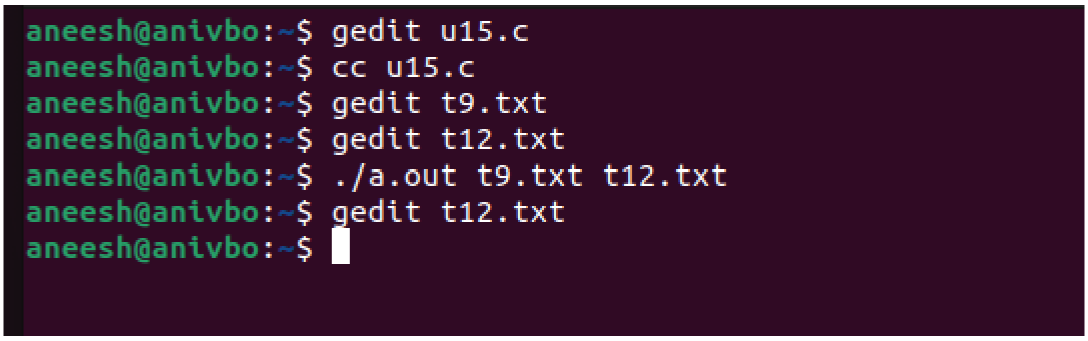
```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#define BUFFER_SIZE 1024
int main(int argc, char *argv[]) {
    if (argc < 3) {
        fprintf(stderr, "Usage: %s <source> <destination>\n", argv[0]);
        return 1;
    }
    int src_fd = open(argv[1], O_RDONLY);
    if (src_fd == -1) {
        perror("open source");
        return 1;
    }
    int dst_fd = open(argv[2], O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);
    if (dst_fd == -1) {
        perror("open destination");
        return 1;
    }
    char buffer[BUFFER_SIZE];
    ssize_t bytes_read;
    while ((bytes_read = read(src_fd, buffer, BUFFER_SIZE)) > 0) {
        if (write(dst_fd, buffer, bytes_read) != bytes_read) {
            perror("write");
            return 1;
        }
    }
    if (bytes_read == -1) {
        perror("read");
    }
    close(src_fd);
    close(dst_fd);
    return 0;
}
```

Commands to Execute:

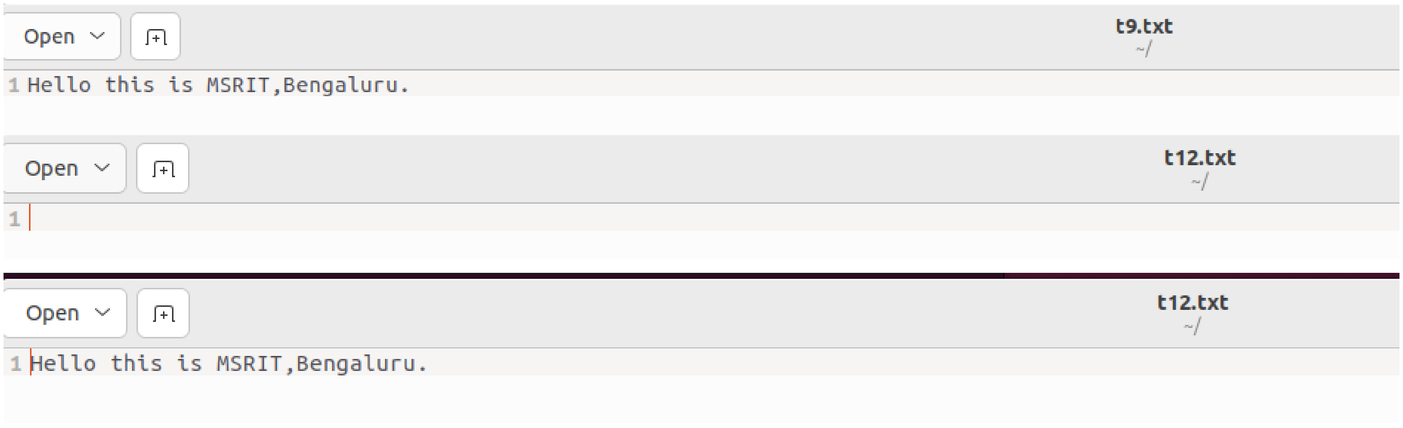
```
vi program_name.c (type program here)
cc program_name.c
```

./a.out <sourcefile_name> <destinationfile_name> (Write some content in sourcefile and keep destinationfile empty)

Output:



```
aneesh@anivbo:~$ gedit u15.c
aneesh@anivbo:~$ cc u15.c
aneesh@anivbo:~$ gedit t9.txt
aneesh@anivbo:~$ gedit t12.txt
aneesh@anivbo:~$ ./a.out t9.txt t12.txt
aneesh@anivbo:~$ gedit t12.txt
aneesh@anivbo:~$ 
```



```
Hello this is MSRIT,Bengaluru.
```

```

```

```
Hello this is MSRIT,Bengaluru.
```

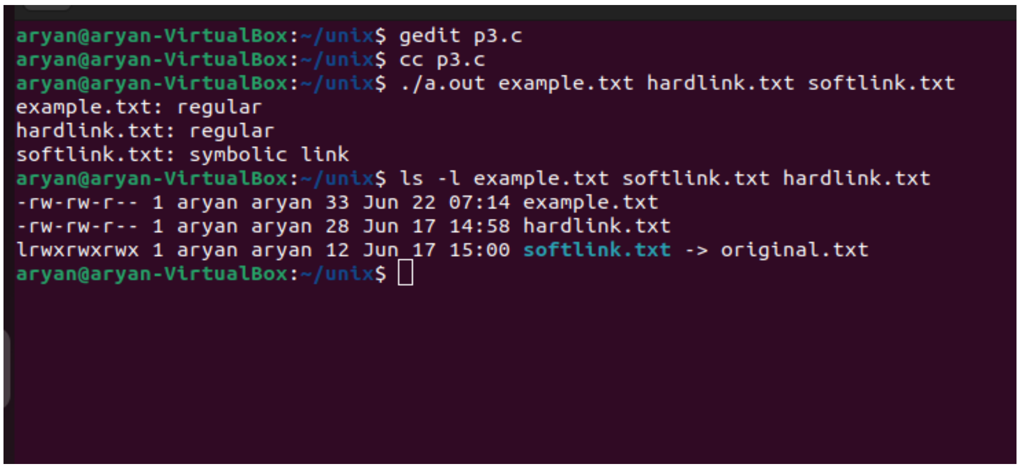
8.a) Write a C program that takes file name as an argument and prints the type of the given file.

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdlib.h>
void err_ret(const char *msg) {
    perror(msg);
}
int main(int argc, char *argv[]) {
    int i;
    struct stat buf;
    char *ptr;
    for (i = 1; i < argc; i++) {
        printf("%s: ", argv[i]);
        if (lstat(argv[i], &buf) < 0) {
            err_ret("lstat error");
            continue;
        }
        if (S_ISREG(buf.st_mode))
            ptr = "regular";
        else if (S_ISDIR(buf.st_mode))
            ptr = "directory";
        else if (S_ISCHR(buf.st_mode))
            ptr = "character special";
        else if (S_ISBLK(buf.st_mode))
            ptr = "block special";
        else if (S_ISFIFO(buf.st_mode))
            ptr = "fifo";
        else if (S_ISLNK(buf.st_mode))
            ptr = "symbolic link";
        else if (S_ISSOCK(buf.st_mode))
            ptr = "socket";
        else
            ptr = "** unknown mode **";
        printf("%s\n", ptr);
    }
    exit(0);
}
```

Commands to Execute:

```
vi program_name.c (type program here)
cc program_name.c
./a.out  <file_name1>  <file_name2>  <file_name3>
```

Output:



```
aryan@aryan-VirtualBox:~/unix$ gedit p3.c
aryan@aryan-VirtualBox:~/unix$ cc p3.c
aryan@aryan-VirtualBox:~/unix$ ./a.out example.txt hardlink.txt softlink.txt
example.txt: regular
hardlink.txt: regular
softlink.txt: symbolic link
aryan@aryan-VirtualBox:~/unix$ ls -l example.txt softlink.txt hardlink.txt
-rw-rw-r-- 1 aryan aryan 33 Jun 22 07:14 example.txt
-rw-rw-r-- 1 aryan aryan 28 Jun 17 14:58 hardlink.txt
lrwxrwxrwx 1 aryan aryan 12 Jun 17 15:00 softlink.txt -> original.txt
aryan@aryan-VirtualBox:~/unix$
```

8.b) Write a C program to perform the following operations

- **i. To create a child process**
- **ii. The child process should execute a program (using exec( )) to show the use of the access function**
- **iii. The Parent process should wait for the child process to exit**
- **iv. Also print the necessary process IDs**

Main program,

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <errno.h>
int main() {
    pid_t pid;
    int status;
    pid_t parent_pid = getpid();
    pid = fork();
    if (pid == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    } else if (pid == 0) {
        pid_t child_pid = getpid();
        printf("Child process (PID: %d) executing...\n", child_pid);
        execl("./p1", "p1", "example.txt", (char *)NULL);
        perror("execl");
        exit(EXIT_FAILURE);
    } else {
        printf("Parent process (PID: %d) executing...\n", parent_pid);
        waitpid(pid, &status, 0);
        printf("Parent process: Child process (PID: %d) has exited.\n", pid);
    }
    return 0;
}
```

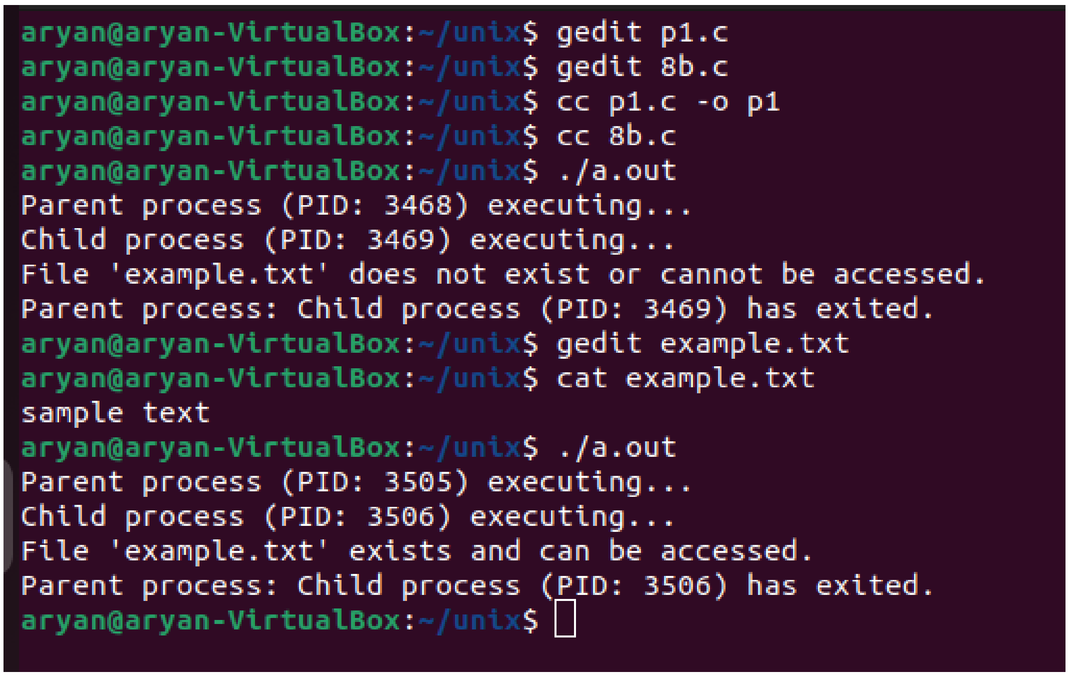
<u>**p1.c**</u>

```
#include   <stdio.h>
#include  <stdlib.h>
#include <unistd.h>
int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "Usage: %s <filename>\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    char *filename = argv[1];
    if (access(filename, F_OK) == 0) {
        printf("File '%s' exists and can be accessed.\n", filename);
    } else {
        printf("File '%s' does not exist or cannot be accessed.\n", filename);
    }
    return 0;
}
```

Commands to Execute:

```
vi program_name.c (type program here)
vi p1.c
cc -o p1 p1.c
cc program_name.c
./a.out
```

Output:



```
aryan@aryan-VirtualBox:~/unix$ gedit p1.c
aryan@aryan-VirtualBox:~/unix$ gedit 8b.c
aryan@aryan-VirtualBox:~/unix$ cc p1.c -o p1
aryan@aryan-VirtualBox:~/unix$ cc 8b.c
aryan@aryan-VirtualBox:~/unix$ ./a.out
Parent process (PID: 3468) executing...
Child process (PID: 3469) executing...
File 'example.txt' does not exist or cannot be accessed.
Parent process: Child process (PID: 3469) has exited.
aryan@aryan-VirtualBox:~/unix$ gedit example.txt
aryan@aryan-VirtualBox:~/unix$ cat example.txt
sample text
aryan@aryan-VirtualBox:~/unix$ ./a.out
Parent process (PID: 3505) executing...
Child process (PID: 3506) executing...
File 'example.txt' exists and can be accessed.
Parent process: Child process (PID: 3506) has exited.
aryan@aryan-VirtualBox:~/unix$
```

9.a) Write a C programs to demonstrate usage of umask and chmod functions.

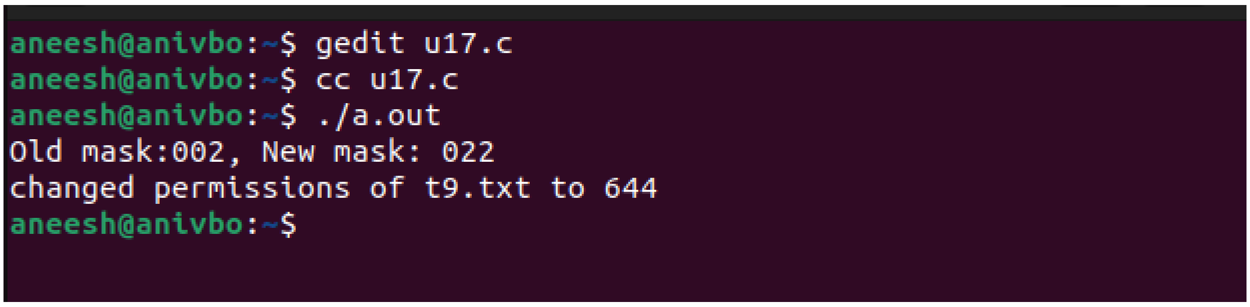
```c
#include <stdio.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
int main() {
    mode_t new_umask = 0022;
    mode_t old_umask;
    const char *file_path = "t1.txt";
    mode_t new_mode = 0644;
    old_umask = umask(new_umask);
    printf("Old umask was: %03o, new umask is: %03o\n", old_umask, new_umask);
    int fd = open(file_path, O_CREAT | O_WRONLY, 0777);
    if (fd == -1) {
        perror("open");
        return 1;
    }
    close(fd);
    if (chmod(file_path, new_mode) == -1) {
        perror("chmod");
        return 1;
    }
    printf("Changed permissions of %s to %03o\n", file_path, new_mode);
    return 0;
}
```

Commands to Execute:

```
vi program_name.c
cc program_name.c
./a.out
```

(remember to create a text file for this code before executing)

Output:



```
aneesh@anivbo:~$ gedit u17.c
aneesh@anivbo:~$ cc u17.c
aneesh@anivbo:~$ ./a.out
Old mask:002, New mask: 022
changed permissions of t9.txt to 644
aneesh@anivbo:~$
```

9.b) Write a C program

i. **To read first 20 characters from a file**
ii. **seek to 10th byte from the beginning and display 20 characters from there**
iii. **seek 10 bytes ahead from the current file offset and display 20 characters**
iv. **Display the file size**

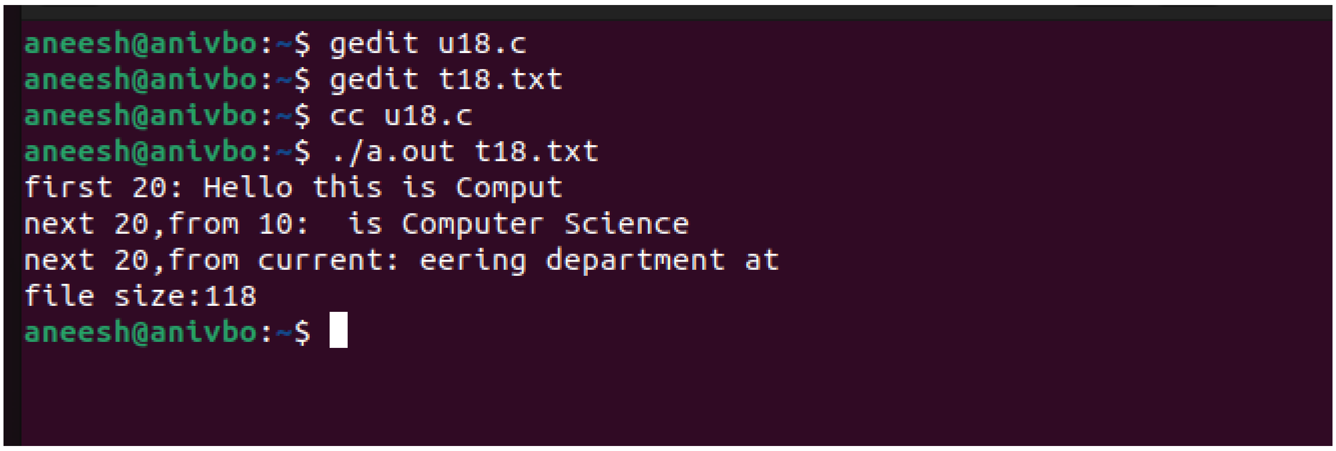
```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
int main(int argc, char *argv[]) {
    if (argc < 2) {
        fprintf(stderr, "Usage: %s <filename>\n", argv[0]);
        return 1;
    }
    int fd = open(argv[1], O_RDONLY);
    if (fd == -1) {
        perror("open");
        return 1;
    }
    char buffer[21];
    if (read(fd, buffer, 20) != 20) {
        perror("read");
        close(fd);
        return 1;
    }
    buffer[20] = '\0';
    printf("First 20 characters: %s\n", buffer);
    lseek(fd, 10, SEEK_SET);
    if (read(fd, buffer, 20) != 20) {
        perror("read");
        close(fd);
        return 1;
    }
    buffer[20] = '\0';
    printf("20 characters from 10th byte: %s\n", buffer);
    lseek(fd, 10, SEEK_CUR);
    if (read(fd, buffer, 20) != 20) {
        perror("read");
        close(fd);
        return 1;
    }
    buffer[20] = '\0';
    printf("20 characters from current offset: %s\n", buffer);
    off_t file_size = lseek(fd, 0, SEEK_END);
    if (file_size == -1) {
        perror("lseek");
        close(fd);
        return 1;
    }
    printf("File size: %lld bytes\n", (long long) file_size);
    close(fd);
    return 0;
}
```

Commands to Execute:

```
vi program_name.c(type your program here)
cc program_name.c
./a.out  <file_name>
```

(make sure to write arnd 100bytes content in file)

Output:



```
aneesh@anivbo:~$ gedit u18.c
aneesh@anivbo:~$ gedit t18.txt
aneesh@anivbo:~$ cc u18.c
aneesh@anivbo:~$ ./a.out t18.txt
first 20: Hello this is Comput
next 20,from 10:  is Computer Science
next 20,from current: eering department at
file size:118
aneesh@anivbo:~$ 
```

10.a) Write a C program such that it initializes itself as a Daemon Process.

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/stat.h>
#include<sys/types.h>
#include<syslog.h>
#include<unistd.h>
#include<fcntl.h>
void create_daemon(){
        pid_t pid=fork();
        if(pid<0){
                exit(EXIT_FAILURE);
        }
        if(pid>0){
                exit(EXIT_SUCCESS);
        }
        if(setsid()<0){
                exit(EXIT_FAILURE);
```

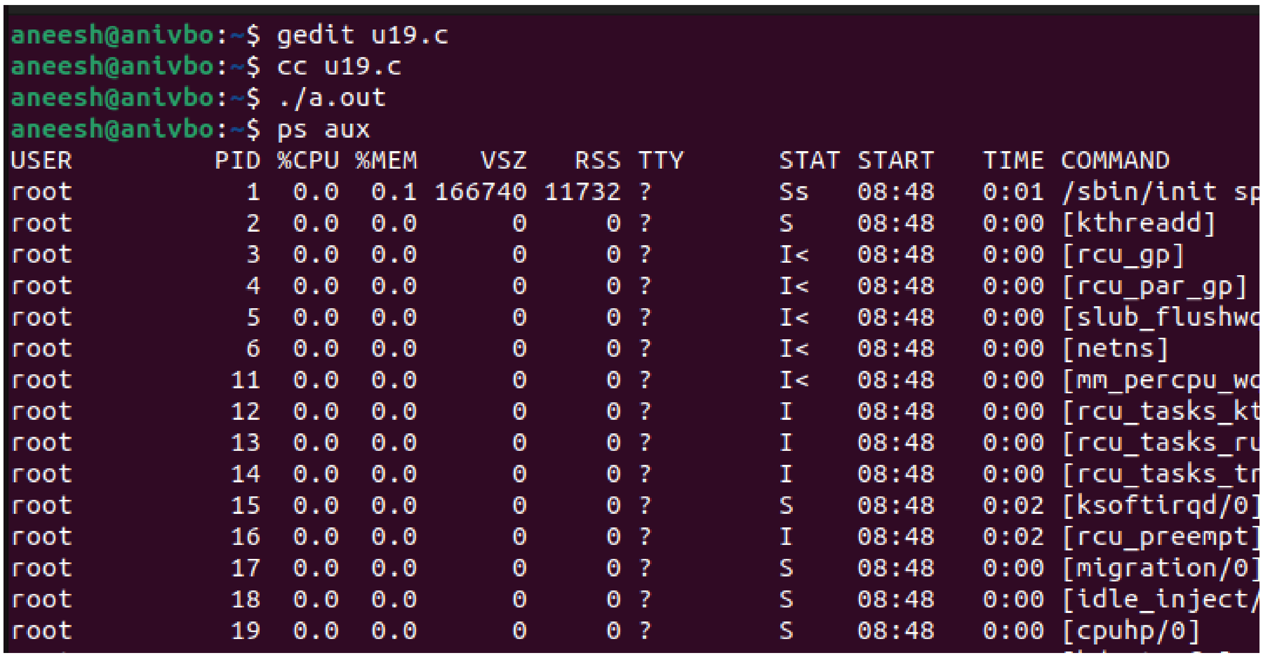
```
          }
          umask(0);
          if(chdir("/")<0){
                    exit(EXIT_FAILURE);
          }
          open("/dev/null",O_RDONLY);
          open("/dev/null",O_WRONLY);
          open("/dev/null",O_RDWR);
          close(STDIN_FILENO);
          close(STDOUT_FILENO);
          close(STDERR_FILENO);
}
int main(){
          create_daemon();
          openlog("daemon_ex",LOG_PID,LOG_DAEMON);
          while(1){
                    syslog(LOG_NOTICE,"Daemon is running...\n");
                    sleep(30);
          }
          closelog();
          return EXIT_SUCCESS;
}
```

Commands to Execute:

vi program_name.c
cc program_name.c
./a.out
ps aux (question mark depicts a daemon process)
tail -f /var/log/syslog (optional to print log statement)

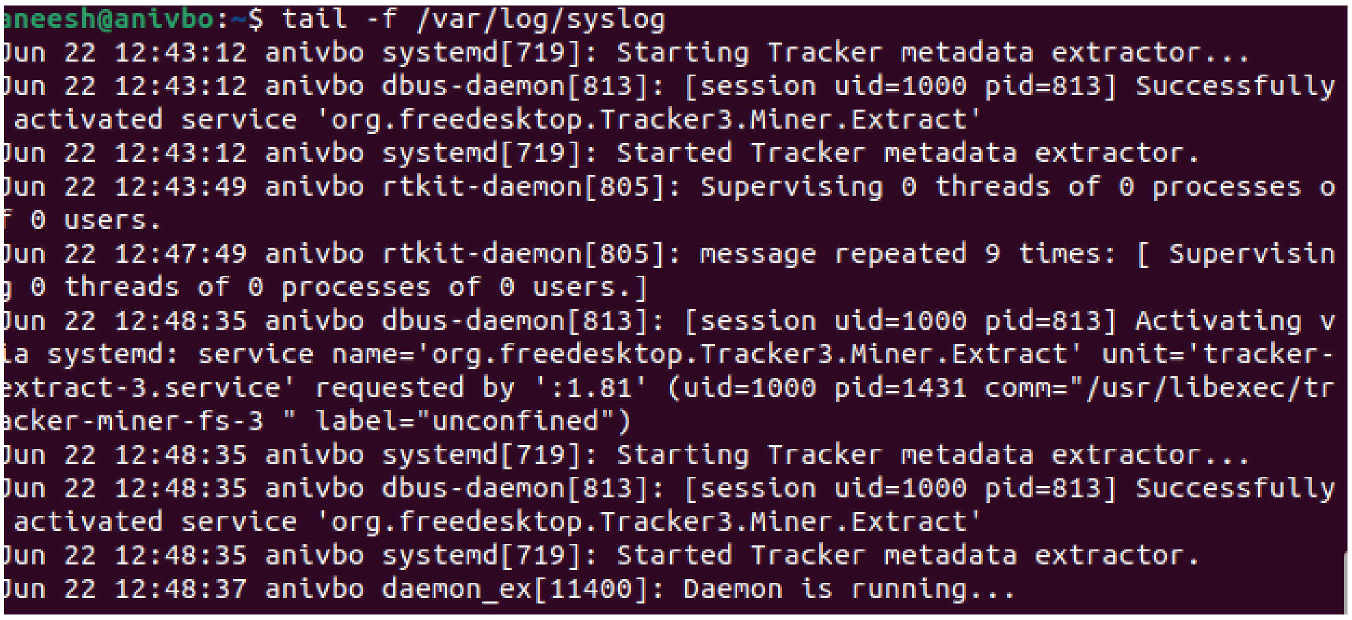
Output:

ps aux



```
aneesh@anivbo:~$ gedit u19.c
aneesh@anivbo:~$ cc u19.c
aneesh@anivbo:~$ ./a.out
aneesh@anivbo:~$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.1 166740 11732 ?        Ss   08:48   0:01 /sbin/init sp
root           2  0.0  0.0      0     0 ?        S    08:48   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   08:48   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   08:48   0:00 [rcu_par_gp]
root           5  0.0  0.0      0     0 ?        I<   08:48   0:00 [slub_flushwq
root           6  0.0  0.0      0     0 ?        I<   08:48   0:00 [netns]
root          11  0.0  0.0      0     0 ?        I<   08:48   0:00 [mm_percpu_wq
root          12  0.0  0.0      0     0 ?        I    08:48   0:00 [rcu_tasks_kt
root          13  0.0  0.0      0     0 ?        I    08:48   0:00 [rcu_tasks_ru
root          14  0.0  0.0      0     0 ?        I    08:48   0:00 [rcu_tasks_tr
root          15  0.0  0.0      0     0 ?        S    08:48   0:02 [ksoftirqd/0]
root          16  0.0  0.0      0     0 ?        I    08:48   0:02 [rcu_preempt]
root          17  0.0  0.0      0     0 ?        S    08:48   0:00 [migration/0]
root          18  0.0  0.0      0     0 ?        S    08:48   0:00 [idle_inject/
root          19  0.0  0.0      0     0 ?        S    08:48   0:00 [cpuhp/0]
```

tail -f /var/log/syslog(optional)



```
aneesh@anivbo:~$ tail -f /var/log/syslog
Jun 22 12:43:12 anivbo systemd[719]: Starting Tracker metadata extractor...
Jun 22 12:43:12 anivbo dbus-daemon[813]: [session uid=1000 pid=813] Successfully
 activated service 'org.freedesktop.Tracker3.Miner.Extract'
Jun 22 12:43:12 anivbo systemd[719]: Started Tracker metadata extractor.
Jun 22 12:43:49 anivbo rtkit-daemon[805]: Supervising 0 threads of 0 processes o
f 0 users.
Jun 22 12:47:49 anivbo rtkit-daemon[805]: message repeated 9 times: [ Supervisin
g 0 threads of 0 processes of 0 users.]
Jun 22 12:48:35 anivbo dbus-daemon[813]: [session uid=1000 pid=813] Activating v
ia systemd: service name='org.freedesktop.Tracker3.Miner.Extract' unit='tracker-
extract-3.service' requested by ':1.81' (uid=1000 pid=1431 comm="/usr/libexec/tr
acker-miner-fs-3 " label="unconfined")
Jun 22 12:48:35 anivbo systemd[719]: Starting Tracker metadata extractor...
Jun 22 12:48:35 anivbo dbus-daemon[813]: [session uid=1000 pid=813] Successfully
 activated service 'org.freedesktop.Tracker3.Miner.Extract'
Jun 22 12:48:35 anivbo systemd[719]: Started Tracker metadata extractor.
Jun 22 12:48:37 anivbo daemon_ex[11400]: Daemon is running...
```

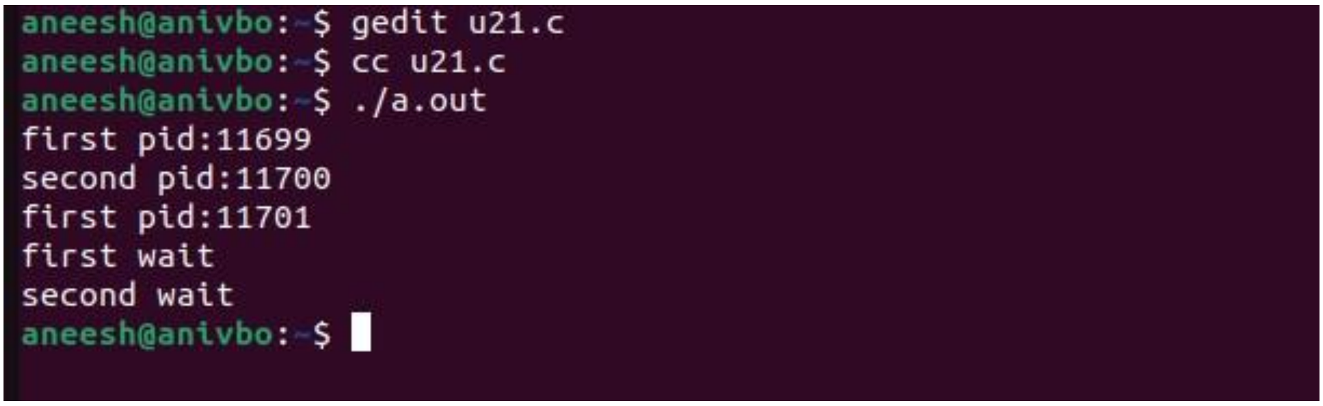
10.b) Demonstrate the working of wait and waitpid system calls with a program.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/wait.h>
#include<sys/types.h>
int main(){
        int st;
        pid_t pid1=fork(); pid_t pid2=fork();
        if(pid1==0){
          printf("first pid:%d\n",getpid());
          sleep(2);
          exit(0);
        }
        if(pid2==0){
          printf("second pid:%d\n",getpid());
          sleep(4);
          exit(0);
        }
        wait(&st);
        printf("first wait\n");
        sleep(1);
        waitpid(pid2,&st,0);
        printf("second wait\n");
        return 0;
}
```

Commands to Execute:

```
vi program_name.c
cc program_name.c
./a.out
```

Output:



```
aneesh@anivbo:~$ gedit u21.c
aneesh@anivbo:~$ cc u21.c
aneesh@anivbo:~$ ./a.out
first pid:11699
second pid:11700
first pid:11701
first wait
second wait
aneesh@anivbo:~$
```

11.a) Write a program to differentiate between dup and dup2 functions.

dup()

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int main() {
    int fd1 = 0, fd2 = 0;
    char buf[10] = "abcdef";
    if ((fd1 = open("t12.txt", O_RDWR, 0)) < 0) {
        printf("error");
    }
    fd2 = dup(fd1);
    printf("%d %d \n", fd1, fd2);
    write(fd1, buf, 6);
    return 0;
}
```

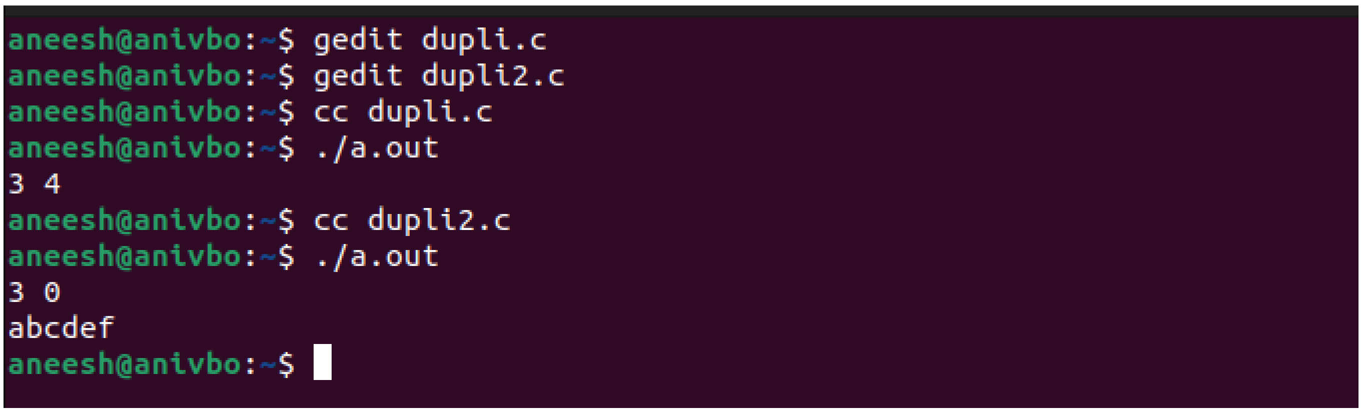
dup2()

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int main() {
    int fd11 = 0, fd12 = 0;
    char buf[10] = "abcdef";
    if ((fd11 = open("t12.txt", O_RDWR, 0)) < 0) {
        printf("error");
    }
    if (dup2(fd12, fd11) < 0) {
        printf("error");
    }
    printf("%d %d \n", fd11, fd12);
    write(fd11, buf, 6);
    return 0;
}
```

Commands to Execute:

vi program_name.c
cc program_name.c
./a.out
(run both codes separately)

Output:



```
aneesh@anivbo:~$ gedit dupli.c
aneesh@anivbo:~$ gedit dupli2.c
aneesh@anivbo:~$ cc dupli.c
aneesh@anivbo:~$ ./a.out
3 4
aneesh@anivbo:~$ cc dupli2.c
aneesh@anivbo:~$ ./a.out
3 0
abcdef
aneesh@anivbo:~$
```

11.b) Write a program to perform the following operations:

i) To create a child process.

ii) The child process should execute a separate program (using exec() function) that calculates the addition of two numbers by passing two integer values.

iii) Parent process should wait for the child to complete.

Main program.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
int main(int argc, char *argv[]) {
    if (argc != 3) {
        printf("Usage: %s num1 num2\n", argv[0]);
        exit(0);
    }
    pid_t pid = fork();
    if (pid < 0) {
        perror("fork failed");
        exit(0);
    } else if (pid == 0) {
        execl("./p23", "p23", argv[1], argv[2], (char *)NULL);
        perror("execl failed");
        exit(EXIT_FAILURE);
    } else {
        int status;
        waitpid(pid, &status, 0);
        if (WIFEXITED(status)) {
            printf("Child exited with status %d\n", WEXITSTATUS(status));
        } else {
            printf("Child terminated abnormally\n");
        }
    }
    return 0;
}
```

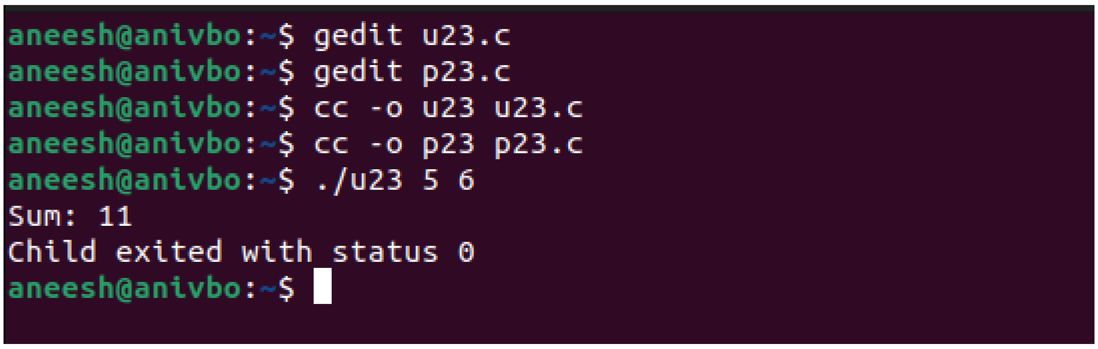
p23.c

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
    if (argc != 3) {
        printf("Usage: %s num1 num2\n", argv[0]);
        exit(0);
    }
    int num1 = atoi(argv[1]);
    int num2 = atoi(argv[2]);
    int sum = num1 + num2;
    printf("Sum: %d\n", sum);
    return 0;
}
```

Commands to Execute:

```
vi program_name.c
vi p23.c
cc -o main_program main_program.c
cc -o p23 p23.c
/main_program  5 6 (these numbers can be anything)
```

Output:



```
aneesh@anivbo:~$ gedit u23.c
aneesh@anivbo:~$ gedit p23.c
aneesh@anivbo:~$ cc -o u23 u23.c
aneesh@anivbo:~$ cc -o p23 p23.c
aneesh@anivbo:~$ ./u23 5 6
Sum: 11
Child exited with status 0
aneesh@anivbo:~$ 
```

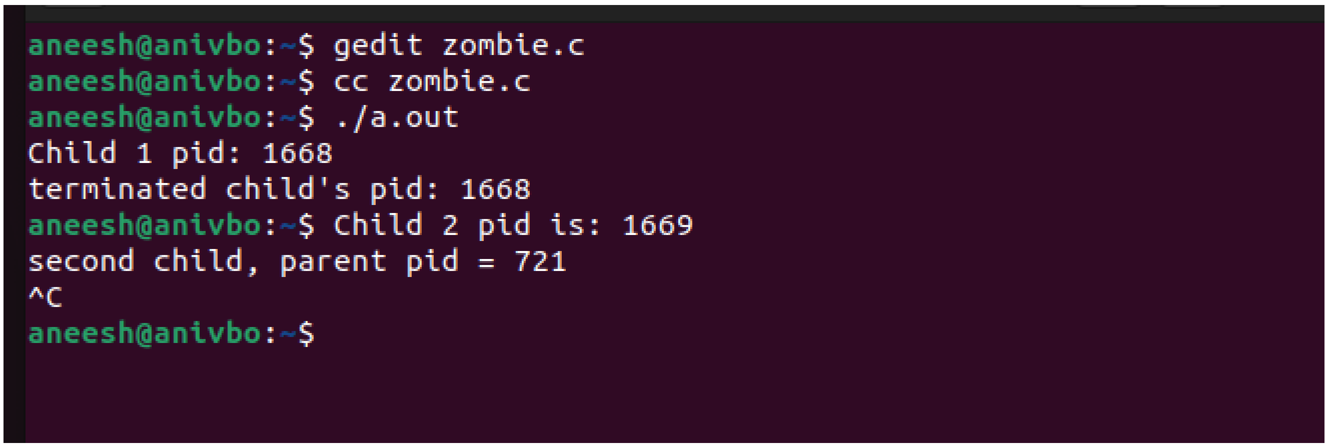
12.a) Write a program to demonstrate the zombie state of a process and provide the solution for the same.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<fcntl.h>
#include <sys/wait.h>
int main(void) {
    pid_t pid,pid1,pid2;
    if ((pid = fork()) < 0) {
        printf("fork error");
    } else if (pid == 0) { /* first child */
        if ((pid3 = fork()) < 0)
            printf("fork error");
        else if (pid3==0) {
            sleep(5);
            printf("Child pid is: %d\n",getpid());
            printf("second child, parent pid = %ld\n", (long)getppid());
            exit(0);
        }
        else{
            printf("Child pid: %d\n",getpid());
            exit(0);
        }
    }
    if ((pid2=waitpid(pid, NULL, 0)) != pid)
        printf("waitpid error");
        printf("terminated child's pid: %d\n",pid2);
    exit(0);
}
```

Commands to Execute:

```
vi program_name.c
cc program_name.c
./a.out
```

Output:



```
aneesh@anivbo:~$ gedit zombie.c
aneesh@anivbo:~$ cc zombie.c
aneesh@anivbo:~$ ./a.out
Child 1 pid: 1668
terminated child's pid: 1668
aneesh@anivbo:~$ Child 2 pid is: 1669
second child, parent pid = 721
^C
aneesh@anivbo:~$
```

12.b) Write a C program to perform the following operations

- **i) To create a child and parent process with the use of an echoall file.**
- **ii) The Child should execute a process that prints the user defined values of environment variables**
- **iii) The Parent should execute a process that prints default values for the environment variables.**

Main program

***** Remember to change the path in the first argument of both execle() and execlp(), set pat according to the location of your echoall file *****

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
void err_sys(const char *message) {
    perror(message);
    exit(1);
}
int main(void) {
    pid_t pid;
    char *env_init[] = { "USER=unknown", "PATH=/tmp", NULL };
    if ((pid = fork()) < 0) {
        err_sys("fork error");
    } else if (pid == 0) {
        if (execle("/home/aneesh/echoall", "echoall", "myarg1", "MY ARG2", (char *)0, env_init) < 0) {
            err_sys("execle error");

        }
    }
    if (waitpid(pid, NULL, 0) < 0) {
        err_sys("wait error");
    }
    if (execlp("/home/aneesh/echoall", "echoall", "only 1 arg", (char *)0) < 0) {
        err_sys("execlp error");
    }
    return 0;
}
```

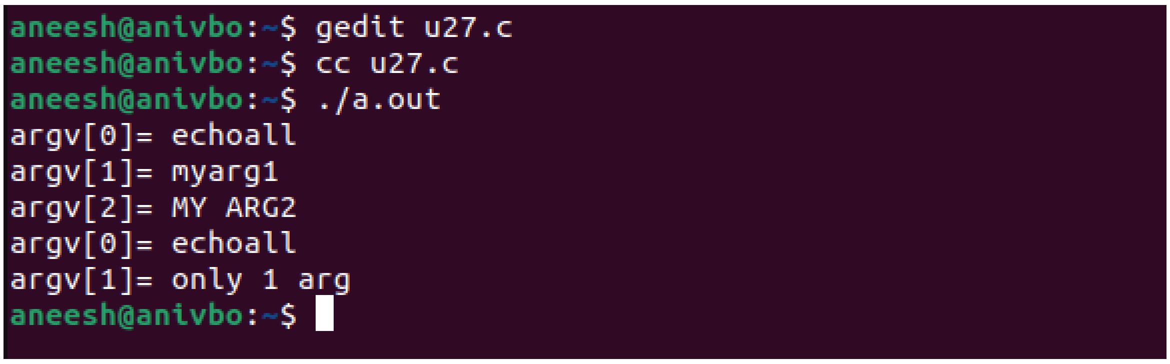
echoall.c file

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc,char *argv[]){
            int i;
            for(i=0;i<argc;i++){
                        printf("argv[%d]= %s\n",i,argv[i]);
            }
return 0;
}
```

Commands to Execute:

```
vi program_name.c
vi echoall.c
cc program_name.c
./a.out
```

Output:



```
aneesh@anivbo:~$ gedit u27.c
aneesh@anivbo:~$ cc u27.c
aneesh@anivbo:~$ ./a.out
argv[0]= echoall
argv[1]= myarg1
argv[2]= MY ARG2
argv[0]= echoall
argv[1]= only 1 arg
aneesh@anivbo:~$ 
```