

```
/*
```

1. Write a program using file operations that demonstrates copying of data from input file and write into output file, until it reaches end of file data.

```
*/
```

```
#include<stdio.h>
```

```
#include<fcntl.h>
```

```
#include<sys/types.h>
```

```
#include<sys/stat.h>
```

```
#include<unistd.h>
```

```
int main(){
```

```
    int fd1,fd2;
```

```
    char readbuf[50];
```

```
    fd1=open("a1r.txt",O_RDONLY,777);
```

```
    fd2=open("a1w.txt",O_CREAT|O_RDWR,777);
```

```
    while((read(fd1,readbuf,50))>0){}
```

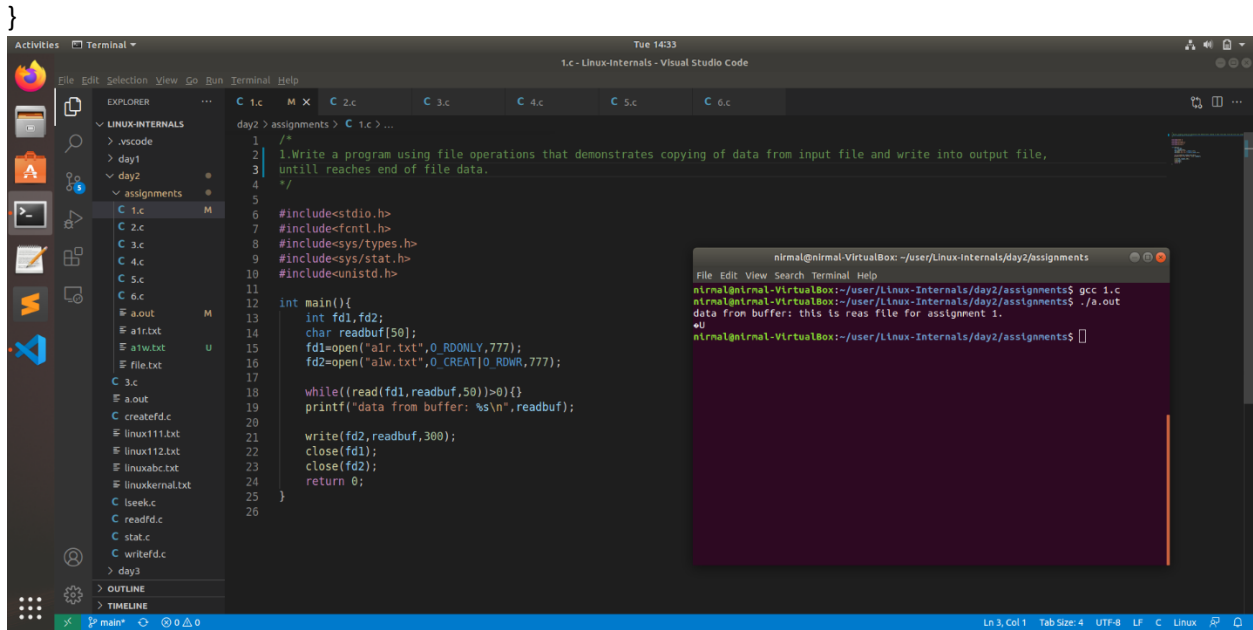
```
    printf("data from buffer: %s\n",readbuf)
```

```
    write(fd2,readbuf,300);
```

```
    close(fd1);
```

```
    close(fd2);
```

```
    return 0;
```



```
/*
```

2. Write a program that demonstrates repositioning of file offset using SEEK\_SET, SEEK\_END and SEEK\_END.

```
*/
```

```
#include<stdio.h>
```

```
#include<fcntl.h>
```

```
#include<sys/types.h>
```

```
#include<sys/stat.h>
```

```
#include<unistd.h>
```

```
int main(){
```

```
    int fd,fd1,len;
```

```
    char writebuf[300]="holla, description of lseek operation:";
```

```
    fd=open("linuxkernel.txt",O_CREAT|O_RDWR,777);
```

```

len = write(fd,writebuf,300);

printf("return value of write option:%d\n",len);

printf("SEEK_SET:%ld\n",lseek(fd,0,SEEK_SET));

lseek(fd,4,SEEK_SET);

printf("SEEK_CURR:%ld\n",lseek(fd,0,SEEK_CUR));

printf("SEEK_END:%ld\n",lseek(fd,0,SEEK_END));

printf("SEEK_SET:%ld\n",lseek(fd,0,SEEK_SET));


close(fd);

return 0;

}

```

```

nirmal@nirmal-VirtualBox: ~/user/Linux-Internals/day2/assignments
File Edit View Search Terminal Help
nirmal@nirmal-VirtualBox:~/user/Linux-Internals/day2/assignments$ gcc 2.c
nirmal@nirmal-VirtualBox:~/user/Linux-Internals/day2/assignments$ ./a.out
return value of write option:300
SEEK_CURR:4
SEEK_END:300
SEEK_SET:0
nirmal@nirmal-VirtualBox:~/user/Linux-Internals/day2/assignments$

```

/\*3. Write program that returns "ls -l " kind of structure of information from an existing file or open file.\*/

```
#include<stdio.h>
```

```
#include<fcntl.h>
```

```
#include<unistd.h>
```

```
#include<sys/stat.h>
```

```
#include<sys/types.h>
```

```
int main(){
```

```
    struct stat st;
```

```
    int fd;
```

```
    stat("lseek.c", &st);
```

```
    printf("File size =%lu\n", (st.st_size));
```

```
    printf("File inode =%lu \n", st.st_ino);
```

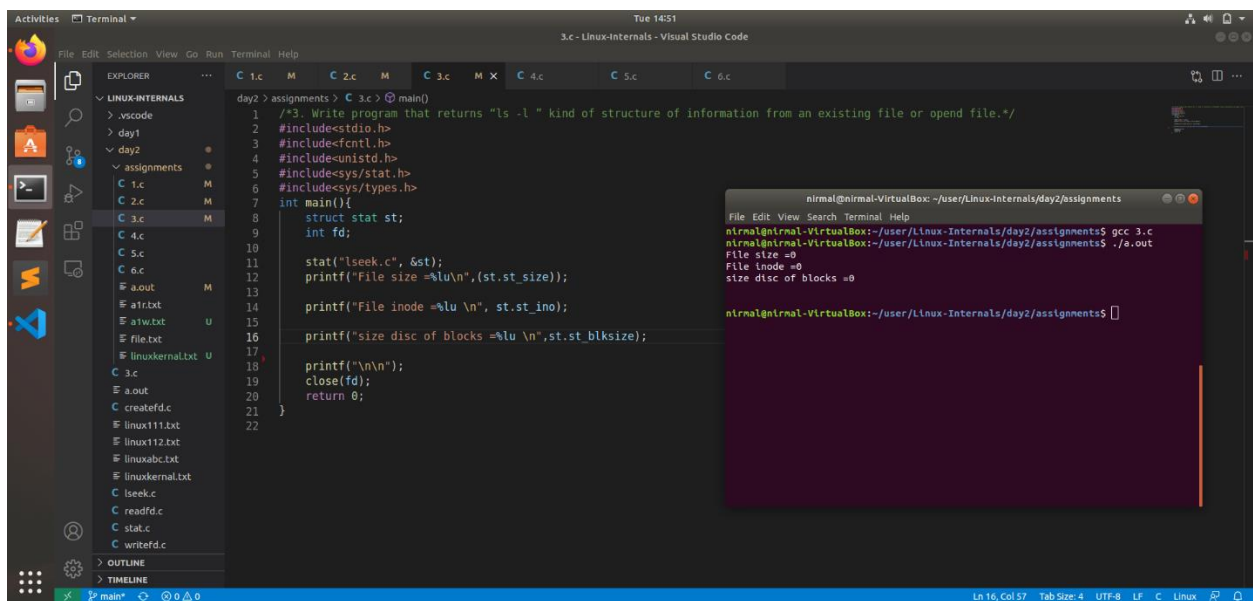
```
    printf("size disc of blocks =%lu \n", st.st_blksize);
```

```
    printf("\n\n");
```

```
    close(fd);
```

```
    return 0;
```

```
}
```



The screenshot shows the Visual Studio Code interface. The Explorer pane on the left displays the file structure of a project named '3.c - Linux-Internals'. The main editor window shows the C program code, which is the same as the one provided in the text blocks. The Output pane on the right shows the execution results of the program, which are: file size =0, file inode =0, and size disc of blocks =0. The terminal window at the bottom shows the command 'gcc 3.c' and the resulting output.

```
day2 > assignments > C 3.c > main()
1 /*3. Write program that returns "ls -l " kind of structure of information from an existing file or open file.*/
2 #include<stdio.h>
3 #include<fcntl.h>
4 #include<unistd.h>
5 #include<sys/stat.h>
6 #include<sys/types.h>
7 int main(){
8     struct stat st;
9     int fd;
10
11     stat("lseek.c", &st);
12     printf("File size =%lu\n", (st.st_size));
13
14     printf("File inode =%lu \n", st.st_ino);
15
16     printf("size disc of blocks =%lu \n", st.st_blksize);
17
18     printf("\n\n");
19     close(fd);
20     return 0;
21 }
22
```

```
nirmal@nirmal-VirtualBox: ~/user/Linux-Internals/day2/assignments
File Edit View Search Terminal Help
nirmal@nirmal-VirtualBox:~/user/Linux-Internals/day2/assignments$ gcc 3.c
nirmal@nirmal-VirtualBox:~/user/Linux-Internals/day2/assignments$ ./a.out
file size =0
file inode =0
size disc of blocks =0

nirmal@nirmal-VirtualBox:~/user/Linux-Internals/day2/assignments$
```

```
/*
```

5. Write a program that creates a file with a 4K bytes free space. (Such files are called files with holes.)

```
*/
```

```
#include <sys/types.h>
```

```
#include <sys/stat.h>
```

```
#include <fcntl.h>
```

```
#include <unistd.h>
```

```
#include <stdio.h>
```

```
char buf1[]="LAB ";
```

```
char buf2[]="OS Linux";
```

```
int main(){
```

```
    int fd;
```

```
    if ((fd=creat("file.txt", 0666)) < 0) {
```

```
        printf("Creation error");
```

```
        exit(1);
```

```
    }
```

```
    if (write(fd, buf1, sizeof(buf1)) < 0){
```

```
        printf("Writing error");
```

```
        exit(2);
```

```
    }
```

```
    if (lseek(fd, 4096, SEEK_SET) < 0){
```

```
        printf("Positioning error");
```

```
        exit(3);
```

```
    }
```

```
    if (write(fd, buf2, sizeof(buf2)) < 0){
```

```
        printf("Writing error");
```

```
        exit(2);
```

```
    }
```

```
}
```

/\*

Trace the execution of the program with the help of the following commands:

ls -l

stat file.txt

od -c file.txt

\*/

The screenshot shows a Visual Studio Code editor with a C program in the main editor and its execution output in a terminal. The program is a simple file creation and manipulation utility. The terminal output shows the program's execution, including file creation, file listing, and file statistics.

```
1 /*
2  5. Write a program that creates a file with a 4K bytes free space. (Such files are called files with holes.)
3  */
4  #include <sys/types.h>
5  #include <sys/stat.h>
6  #include <fcntl.h>
7  #include <unistd.h>
8  #include <stdio.h>
9  #include <stdlib.h>
10 char buf1[] = "LAB ";
11 char buf2[] = "OS Linux";
12
13 int main() {
14     int fd;
15     if ((fd = creat("file.txt", 0666)) < 0) {
16         printf("Creation error");
17         exit(1);
18     }
19     if (write(fd, buf1, sizeof(buf1)) < 0) {
20         printf("Writing error");
21         exit(2);
22     }
23     if (lseek(fd, 4096, SEEK_SET) < 0) {
24         printf("Positioning error");
25         exit(3);
26     }
27     if (write(fd, buf2, sizeof(buf2)) < 0) {
28         printf("Writing error");
29         exit(2);
30     }
31 }
32 */
```

The terminal output shows the program's execution, including file creation, file listing, and file statistics.

```
nirmal@nirmal-VirtualBox:~/Linux-Internals/day2/assignments$ ls -l
total 60
-rw-rw-r-- 1 nirmal nirmal 527 Mar 15 14:32 1.c
-rw-rw-r-- 1 nirmal nirmal 676 Mar 15 14:47 2.c
-rw-rw-r-- 1 nirmal nirmal 464 Mar 15 14:50 3.c
-rw-rw-r-- 1 nirmal nirmal 587 Mar 15 15:00 4.c
-rw-rw-r-- 1 nirmal nirmal 780 Mar 15 15:05 5.c
-rw-rw-r-- 1 nirmal nirmal 36 Mar 8 14:33 air.txt
-rwxr-xr-x 1 nirmal nirmal 300 Mar 15 14:31 a1w.txt
-rwxr-xr-x 1 nirmal nirmal 0 Mar 15 15:01 a4w1.txt
-rwxr-xr-x 1 nirmal nirmal 60 Mar 15 15:01 a4w.txt
-rwxr-xr-x 1 nirmal nirmal 8544 Mar 15 15:05 a.out
-rw-rw-r-- 1 nirmal nirmal 4105 Mar 15 15:05 file.txt
-rwxr-xr-x 1 nirmal nirmal 300 Mar 15 14:47 linuxkernel.txt
nirmal@nirmal-VirtualBox:~/Linux-Internals/day2/assignments$ stat file.txt
  File: file.txt
  Size: 4105      Blocks: 16      IO Block: 4096   regular file
Device: 801h/2049d Inode: 660518  Links: 1
Access: (0664/-rw-rw-r-- )  Uid: ( 1000/  nirmal)   Gid: ( 1000/  nirmal)
Access: 2022-03-08 16:47:55.609921501 +0530
Modify: 2022-03-15 15:05:42.644864084 +0530
Change: 2022-03-15 15:05:42.644864084 +0530
 Birth: -
```

```
/*
```

6.write a program that used to check for the existance of the file and also check whether we can open file for read,write,execute or not.

```
*/
```

```
#include<stdio.h>
```

```
#include<fcntl.h>
```

```
#include<unistd.h>
```

```
#include<stdlib.h>
```

```
#include<string.h>
```

```
int main(){
```

```
    int fd4;
```

```
    fd4 = open("sample.txt", O_RDWR, 777); //to read the file(in readonly mode)
```

```
    if (fd4>0) { //checking file exist
```

```
        printf("file exist\n");
```

```
        char to_write[] = "this lines will be added to file. this one is also be added";
```

```
        int leng = strlen(to_write);
```

```
        char to_read[leng];
```

```
        if(write(fd4, to_write, leng)<0){ // writing permission in file
```

```
            printf("failed to write\n");
```

```
        }
```

```
        else if(read(fd4, to_read, leng)<0){ //reading permission in file
```

```
            printf("unable to read the file\n");
```

```
        }
```

```
        else{
```

```
            printf("read write successfull\n");
```

```
        }
```

```
    }
```

```
    else{
```

```
        printf("not exists\n");
```

```

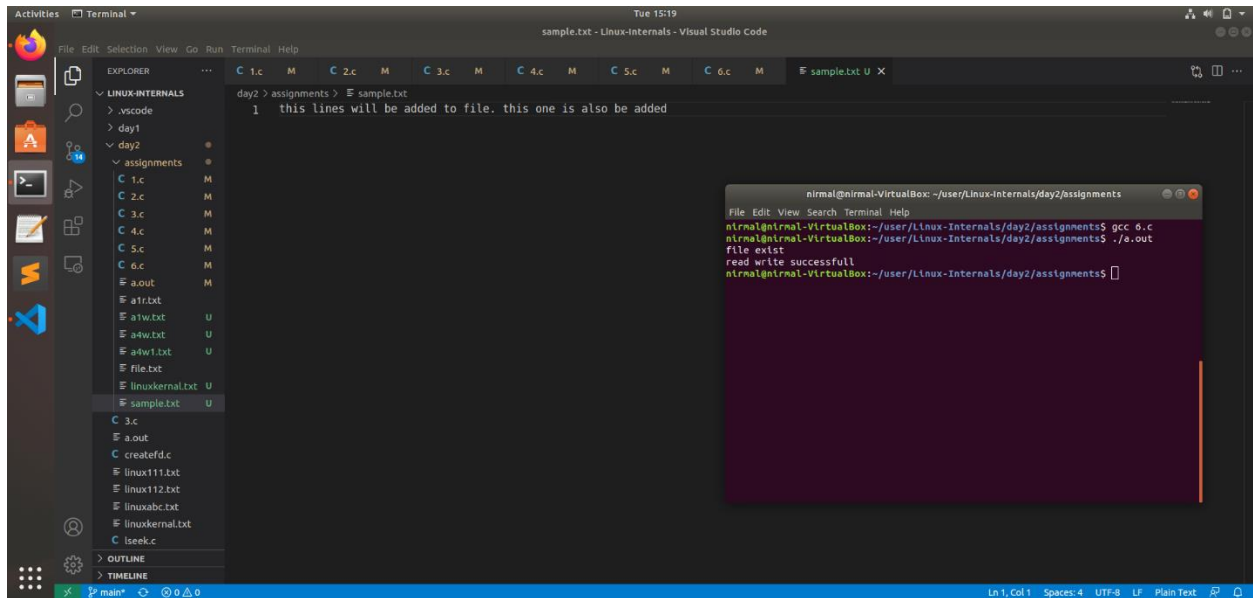
}

int close(int fd4);

return 0;

}

```



```

/*

4. Write a program that implements all file operations(open/creat/write/read/lseek/close).

```

```

*/

#include<stdio.h>

#include<fcntl.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<unistd.h>

```

```

int main(){

    int fd1,fd2,fd3,len=60,rr;

    int x;

    char writebuf[60]="new linux file for all fille ops";

```



```
char readbuf[60];

fd1 = open ("a4w.txt",O_CREAT | O_RDWR,777);           //open
write(fd1,writebuf,len);                                //write

fd1 = open ("sample.txt",O_RDONLY,777);                 //open
read(fd1,readbuf,len);
//read
printf("read data is:%s\n",readbuf);

fd2 = creat("a4w1.txt",777);                             //creat
close(fd2);
close(fd1);
    //close
return 0;
}
```

