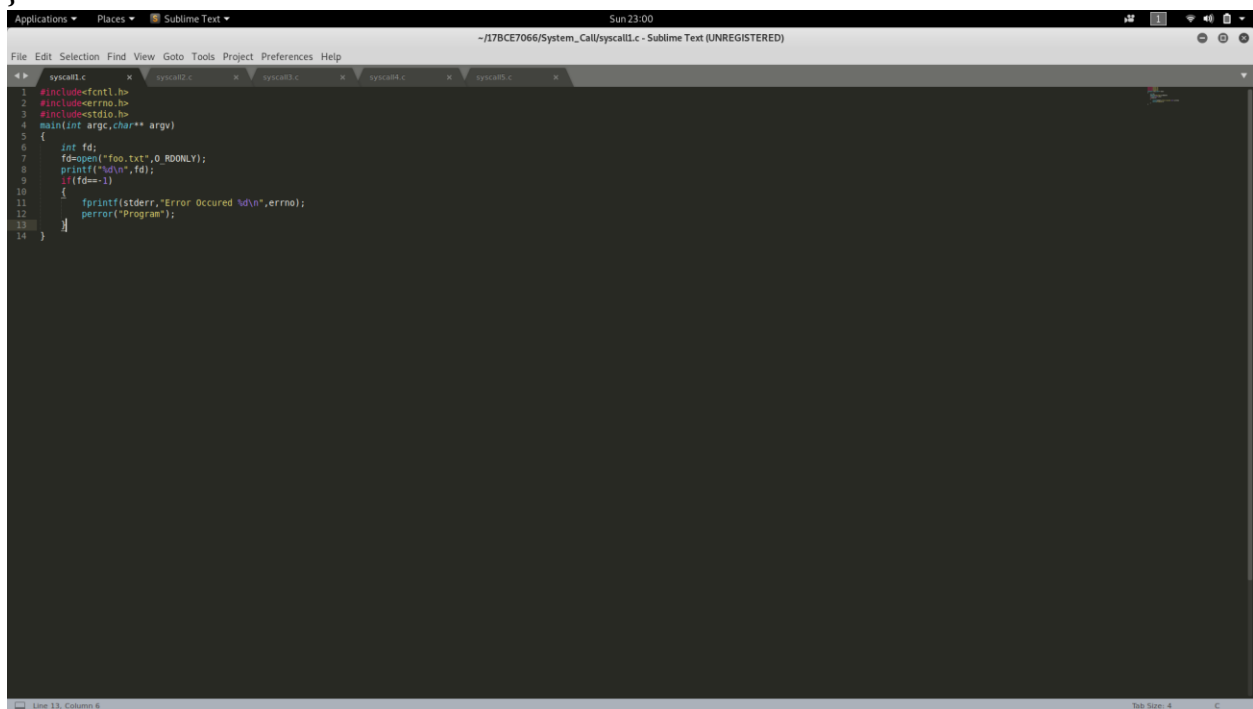


## System-Calls

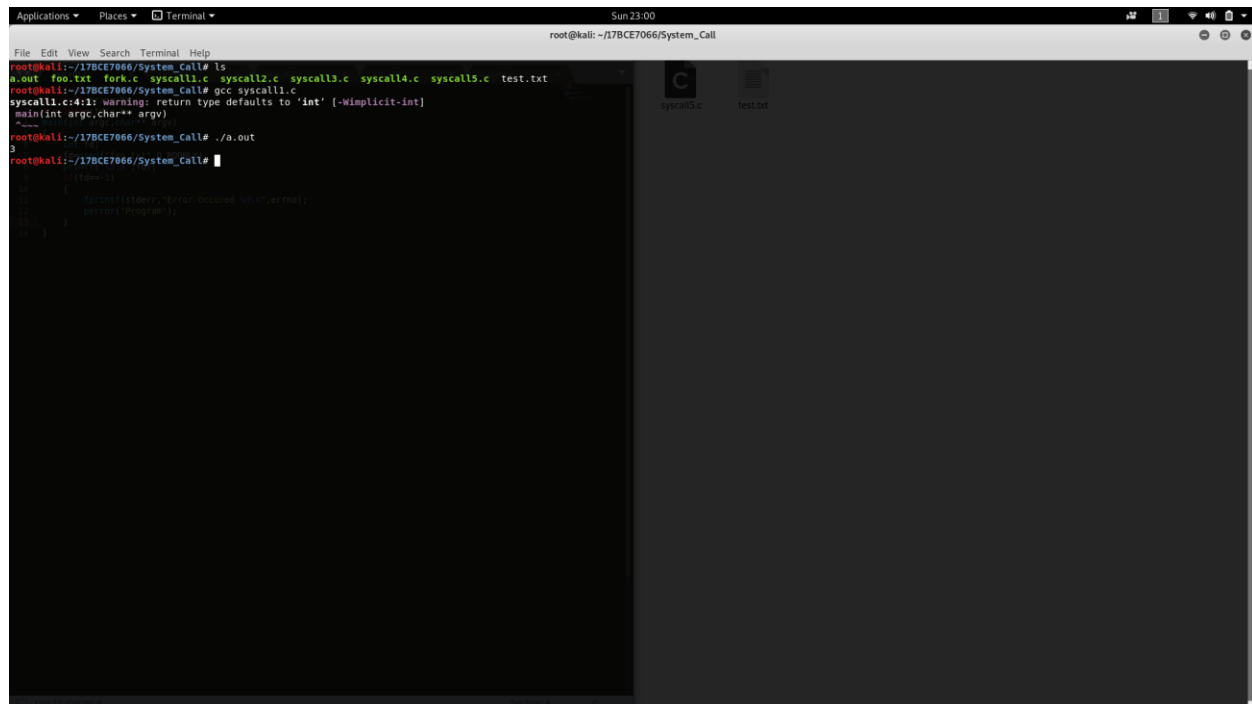
Aditya Jain (17BCE7066)

### Program 1: To open a file

```
#include<fcntl.h>
#include<errno.h>
#include<stdio.h>
main(int argc,char** argv)
{
    int fd;
    fd=open("foo.txt",O_RDONLY);
    printf("%d\n",fd);
    if(fd== -1)
    {
        fprintf(stderr,"Error Occured %d\n",errno);
        perror("Program");
    }
}
```

A screenshot of a Sublime Text editor window. The title bar shows "Sun 23:00" and the window title is "-/17BCE7066/System\_Call/syscall1.c - Sublime Text (UNREGISTERED)". The menu bar includes "File", "Edit", "Selection", "Find", "View", "Goto", "Tools", "Project", "Preferences", and "Help". The code editor shows the same C program as above, with line numbers 1 through 14 on the left margin. The code is syntax-highlighted: preprocessor directives are in blue, keywords like 'main', 'int', and 'if' are in red, and string literals are in green. The status bar at the bottom indicates "Line 13, Column 8" and "Tab Size: 4".

## Output:

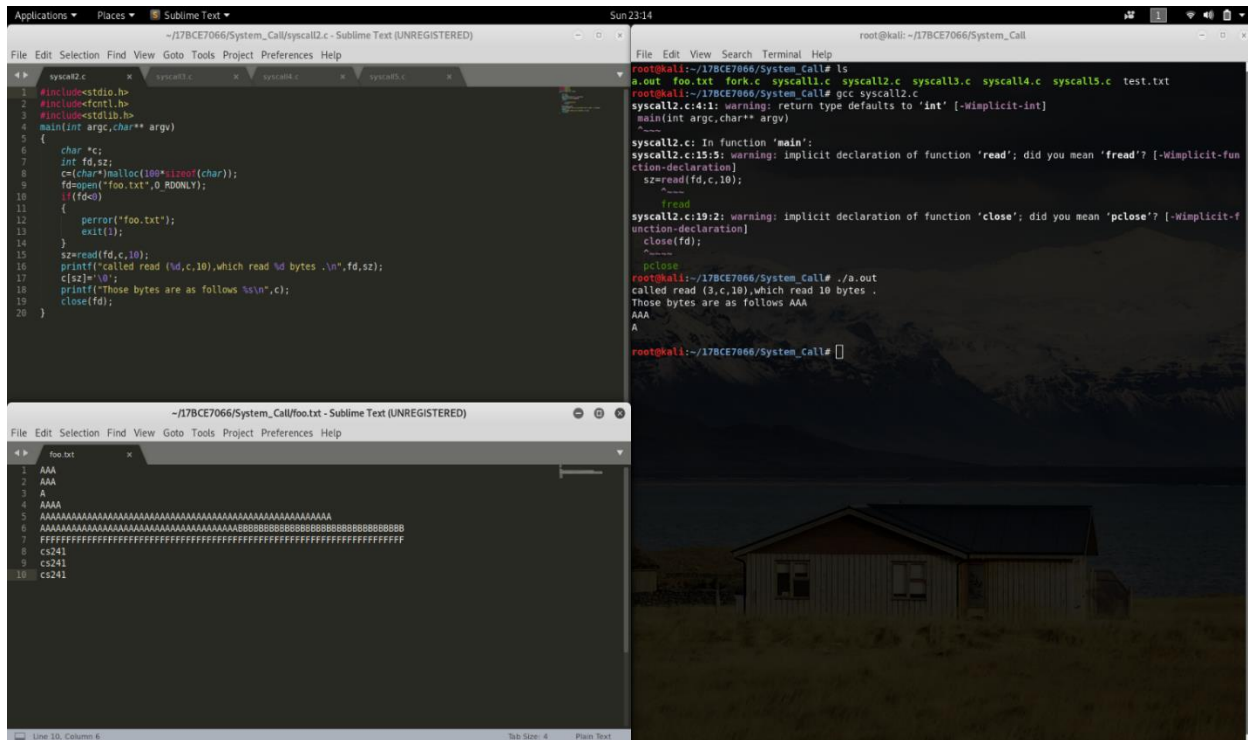


```
root@kali:~/178CE7066/System_Call# ls
a.out  foo.txt  fork.c  syscall2.c  syscall3.c  syscall4.c  syscall5.c  test.txt
root@kali:~/178CE7066/System_Call# gcc syscall1.c
syscall1.c:4:1: warning: return type defaults to 'int' [-Wimplicit-int]
main(int argc,char** argv)
^
root@kali:~/178CE7066/System_Call# ./a.out
called read (1,10),which read 10 bytes .\n
Those bytes are as follows 0\n
```

## Program 2: To read from a file

```
#include<stdio.h>
#include<fcntl.h>
#include<stdlib.h>
main(int argc,char** argv)
{
    char *c;
    int fd,sz;
    c=(char*)malloc(100*sizeof(char));
    fd=open("foo.txt",O_RDONLY);
    if(fd<0)
    {
        perror("foo.txt");
        exit(1);
    }
    sz=read(fd,c,10);
    printf("called read (%d,c,10),which read %d bytes .\n",fd,sz);
    c[sz]='\0';
    printf("Those bytes are as follows %s\n",c);
    close(fd);
}
```

## Output:



The screenshot shows a terminal window and a Sublime Text editor. The terminal window displays the output of a program that reads from a file named 'foo.txt' and prints the contents. The output is: 'called read (3,c,10), which read 10 bytes . Those bytes are as follows AAA A A'. The Sublime Text editor shows the source code of the program, which includes headers for stdio.h, fcntl.h, stdlib.h, and string.h. The main function opens 'foo.txt' in read mode, reads 10 bytes into a character array 'c', and prints the contents. The program also includes error handling for file opening and closing.

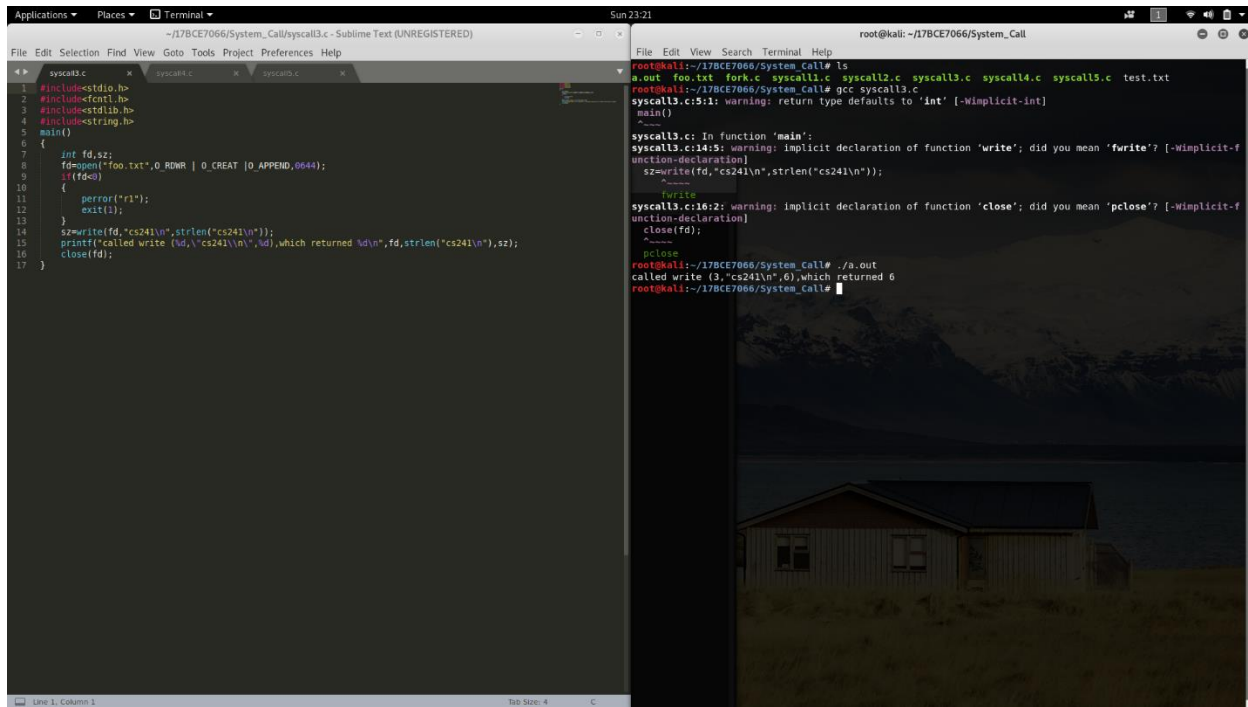
```
File Edit Selection Find View Goto Tools Project Preferences Help
1 #include<stdio.h>
2 #include<fcntl.h>
3 #include<stdlib.h>
4 #include<string.h>
5 void main()
6 {
7     int fd,sz;
8     fd=open("foo.txt",O_RDONLY | O_CREAT | O_APPEND,0644);
9     if(fd<0)
10     {
11         perror("r1");
12         exit(1);
13     }
14     sz=read(fd,c,10);
15     printf("called read (%d,c,10), which read %d bytes .\n",fd,sz);
16     c[sz]='\0';
17     printf("Those bytes are as follows %s\n",c);
18     close(fd);
19 }
20
```

```
root@kali:~/178CE7066/System_Call# ls
a.out foo.txt fork.c syscall1.c syscall2.c syscall3.c syscall4.c syscall5.c test.txt
root@kali:~/178CE7066/System_Call# gcc syscall2.c
syscall2.c:4:1: warning: return type defaults to 'int' [-Wimplicit-int]
main(int argc,char** argv)
^~~~~
syscall2.c: In function 'main':
syscall2.c:15:15: warning: implicit declaration of function 'read'; did you mean 'freed'? [-Wimplicit-f
unction-declaration]
    sz=read(fd,c,10);
              ^~~~~
syscall2.c:19:12: warning: implicit declaration of function 'close'; did you mean 'pclose'? [-Wimplicit-f
unction-declaration]
    close(fd);
           ^~~~~
pclose
root@kali:~/178CE7066/System_Call# ./a.out
called read (3,c,10), which read 10 bytes .
Those bytes are as follows AAA
A
A
root@kali:~/178CE7066/System_Call#
```

## Program 3: To write in a file

```
#include<stdio.h>
#include<fcntl.h>
#include<stdlib.h>
#include<string.h>
void main()
{
    int fd,sz;
    fd=open("foo.txt",O_RDWR | O_CREAT | O_APPEND,0644);
    if(fd<0)
    {
        perror("r1");
        exit(1);
    }
    sz=write(fd,"cs241\n",strlen("cs241\n"));
    printf("called write (%d,\"cs241\n\",%d), which returned
%d\n",fd,strlen("cs241\n"),sz);
    close(fd);
}
```

## Output:



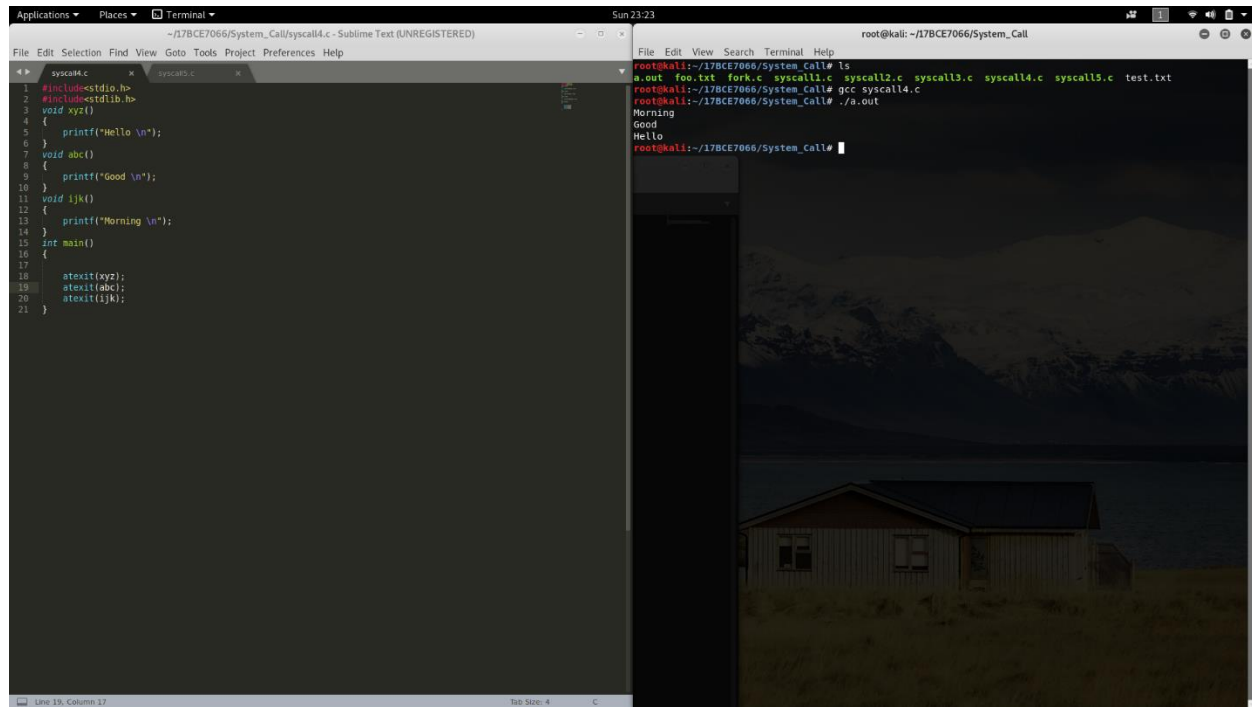
```
Applications ▾ Places ▾ Terminal ▾ Sun 23:21
~/178CE7066/System_Call/syscall3.c - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
1 #include<stdio.h>
2 #include<fcntl.h>
3 #include<stdlib.h>
4 #include<string.h>
5 main()
6 {
7     int fd,sz;
8     fd=open("foo.txt",O_RDWR | O_CREAT | O_APPEND,0644);
9     if(fd<0)
10     {
11         perror("r!");
12         exit(1);
13     }
14     sz=write(fd,"cs241\n",strlen("cs241\n"));
15     printf("called write (%d,\"cs241\n\",%d),which returned %d\n",fd,strlen("cs241\n"),sz);
16     close(fd);
17 }

root@kali:~/178CE7066/System_Call# ls
a.out foo.txt fork.c syscall2.c syscall3.c syscall4.c syscall5.c test.txt
root@kali:~/178CE7066/System_Call# gcc syscall3.c
syscall3.c:5:1: warning: return type defaults to 'int' [-Wimplicit-int]
main()
^
syscall3.c: In function 'main':
syscall3.c:14:5: warning: implicit declaration of function 'write'; did you mean 'fwrite'? [-Wimplicit-f
unction-declaration]
sz=write(fd,"cs241\n",strlen("cs241\n"));
^~~~~~
syscall3.c:15:2: warning: implicit declaration of function 'close'; did you mean 'pclose'? [-Wimplicit-f
unction-declaration]
close(fd);
^~~~~~
root@kali:~/178CE7066/System_Call# ./a.out
called write (3,"cs241\n",6),which returned 6
root@kali:~/178CE7066/System_Call#
```

## Program 4: To demonstrate atexit()

```
#include<stdio.h>
#include<stdlib.h>
void xyz()
{
    printf("Hello \n");
}
void abc()
{
    printf("Good \n");
}
void ijk()
{
    printf("Morning \n");
}
int main()
{
    atexit(xyz);
    atexit(abc);
    atexit(ijk);
}
```

## Output:



The screenshot displays a development environment with Sublime Text and a terminal. The Sublime Text window, titled '~/.7BCE7066/System\_Call/syscall4.c - Sublime Text (UNREGISTERED)', shows a C program in a file named 'syscall4.c'. The code includes standard headers, defines a 'xyz' function that prints 'Hello\n', an 'abc' function that prints 'Good\n', and a 'main' function that calls 'atexit' for each and prints 'Morning\n'. The terminal window, titled 'root@kali: ~/.7BCE7066/System\_Call', shows the execution of the program. It lists files in the directory, compiles 'syscall4.c' with 'gcc', and runs the resulting 'a.out' binary, which produces the output: 'Morning\n', 'Good\n', and 'Hello\n'.

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 void xyz()
4 {
5     printf("Hello\n");
6 }
7 void abc()
8 {
9     printf("Good\n");
10 }
11 void ijk()
12 {
13     printf("Morning\n");
14 }
15 int main()
16 {
17     atexit(xyz);
18     atexit(abc);
19     atexit(ijk);
20 }
21
```

```
root@kali: ~/.7BCE7066/System_Call# ls
a.out  foo.txt  fork.c  syscall1.c  syscall2.c  syscall3.c  syscall4.c  syscall5.c  test.txt
root@kali: ~/.7BCE7066/System_Call# gcc syscall4.c
root@kali: ~/.7BCE7066/System_Call# ./a.out
Morning
Good
Hello
root@kali: ~/.7BCE7066/System_Call#
```