

Assignment 02

Computer Systems

2023PCS0034

Mohammad Faisal Sayed

1) Linux Commands

1. Check current working directory:

Command: **pwd**

Description: Prints the current working directory.

Output:

```
ubuntu@Ubuntu:~$ pwd
/home/ubuntu
```

2. Check the contents in the current directory:

Command: **ls**

Description: Lists the contents of the current directory.

Output:

```
ubuntu@Ubuntu:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  snap  Templates  Videos
```

3. Create a new directory:

Command: **mkdir new_directory**

Description: Creates a new directory named "new_directory".

Output:

```
ubuntu@Ubuntu:~$ mkdir newFolder
ubuntu@Ubuntu:~$ ls
Desktop  Documents  Downloads  Music  newFolder  Pictures  Public  snap  Templates  Videos
```

4. Copy and move files/folders:

Command: **cp source_path destination_path**

Description: Copies files/folders from the source to the destination.

Output:

```

ubuntu@Ubuntu:~$ mkdir srcFolder destFolder
ubuntu@Ubuntu:~$ cd srcFolder/
ubuntu@Ubuntu:~/srcFolder$ touch main.py
ubuntu@Ubuntu:~/srcFolder$ ls
main.py
ubuntu@Ubuntu:~/srcFolder$ cp main.py ../destFolder/
ubuntu@Ubuntu:~/srcFolder$ ls
main.py
ubuntu@Ubuntu:~/srcFolder$ cd ..
ubuntu@Ubuntu:~$ cd destFolder/
ubuntu@Ubuntu:~/destFolder$ ls
main.py

```

Command: **mv source_path destination_path**

Description: Moves files/folders from the source to the destination.

Output:

```

root@Ubuntu:/home/ubuntu# ls
Desktop destFolder Documents Downloads Music newFolder Pictures Public snap srcFolder Templates Videos
root@Ubuntu:/home/ubuntu# mv ./srcFolder/main.py Desktop/
root@Ubuntu:/home/ubuntu# ls
Desktop destFolder Documents Downloads Music newFolder Pictures Public snap srcFolder Templates Videos
root@Ubuntu:/home/ubuntu# cd Desktop/
root@Ubuntu:/home/ubuntu/Desktop# ls
main.py
root@Ubuntu:/home/ubuntu/Desktop# $

```

Download and unzip a .tar file:

Command: **wget URL/to/your/file.tar**

Description: Downloads a .tar file from a specified URL.

```

root@Ubuntu:/home/ubuntu/Desktop# wget https://getsamplefiles.com/download/tar/sample-1.tar
--2023-08-19 12:27:00-- https://getsamplefiles.com/download/tar/sample-1.tar
Resolving getsamplefiles.com (getsamplefiles.com)... 68.183.46.179
Connecting to getsamplefiles.com (getsamplefiles.com)|68.183.46.179|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1101824 (1.1M) [application/x-tar]
Saving to: 'sample-1.tar'

sample-1.tar           100%[=====] 1.05M  933KB/s  in 1.2s

2023-08-19 12:27:08 (933 KB/s) - 'sample-1.tar' saved [1101824/1101824]

root@Ubuntu:/home/ubuntu/Desktop# tar -xvf sample-1.tar
sample-1/
sample-1/sample-1.webp
sample-1/sample-1_1.webp
sample-1/sample-5 (1).jpg
sample-1/sample-5.webp
root@Ubuntu:/home/ubuntu/Desktop#

```



Command: **tar -xvf file.tar**

Description: Extracts the contents of the downloaded .tar file.

Output:

```
root@Ubuntu:/home/ubuntu/Desktop# wget https://getsamplefiles.com/download/tar/sample-1.tar
--2023-08-19 12:27:06-- https://getsamplefiles.com/download/tar/sample-1.tar
Resolving getsamplefiles.com (getsamplefiles.com)... 68.183.46.179
Connecting to getsamplefiles.com (getsamplefiles.com)[68.183.46.179]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1101824 (1.1M) [application/x-tar]
Saving to: 'sample-1.tar'

sample-1.tar          100%[=====] 1.05M  933KB/s   in 1.2s

2023-08-19 12:27:08 (933 KB/s) - 'sample-1.tar' saved [1101824/1101824]

root@Ubuntu:/home/ubuntu/Desktop# tar -xvf sample-1.tar
sample-1/
sample-1/sample-1.webp
sample-1/sample-1_1.webp
sample-1/sample-5 (1).jpg
sample-1/sample-5.webp
root@Ubuntu:/home/ubuntu/Desktop#
```

5. Show contents of a file:

Command: **cat file.txt**

Description: Displays the contents of "file.txt" in the terminal.

Output:

```
main.py
ubuntu@Ubuntu:~/destFolder$ cat main.py
print("Hello World")
ubuntu@Ubuntu:~/destFolder$ ls
main.py
ubuntu@Ubuntu:~/destFolder$ cat main.py
print("Hello World")
ubuntu@Ubuntu:~/destFolder$
```

This is the content of the file.

It can have multiple lines.

6. Search for a keyword within a file:

Command: **grep "keyword" file.txt**

Description: Searches for the specified "keyword" within "file.txt" and displays matching lines.

Output:

```
print("Hello World")
ubuntu@Ubuntu:~/destFolder$ grep "Hello" main.py
print("Hello World")
```

This line contains the keyword.

Another line with the keyword.

7. Show difference between two files:

Command: **diff file1.txt file2.txt**

Description: Compares the contents of "file1.txt" and "file2.txt" and shows the differences.

Output:

```
ubuntu@Ubuntu:~/destFolder$ echo "Content of file1">>file1.txt && echo "Content of file2">>file2.txt
ubuntu@Ubuntu:~/destFolder$ diff file1.txt file2.txt
1c1
< Content of file1
---
> Content of file2
ubuntu@Ubuntu:~/destFolder$ ls -l
```

8. View and change permissions of a file:

Command: **ls -l file.txt**

Description: Displays the permissions and other information of "file.txt".

Output:

```
ubuntu@Ubuntu:~/destFolder$ ls -l
total 12
-rw-rw-r-- 1 ubuntu ubuntu 17 Aug 19 11:34 file1.txt
-rw-rw-r-- 1 ubuntu ubuntu 17 Aug 19 11:34 file2.txt
-rw-rw-r-- 1 ubuntu ubuntu 21 Aug 19 11:31 main.py
ubuntu@Ubuntu:~/destFolder$ chmod 777 file1.txt
ubuntu@Ubuntu:~/destFolder$ ls -l
total 12
-rwxrwxrwx 1 ubuntu ubuntu 17 Aug 19 11:34 file1.txt
-rw-rw-r-- 1 ubuntu ubuntu 17 Aug 19 11:34 file2.txt
-rw-rw-r-- 1 ubuntu ubuntu 21 Aug 19 11:31 main.py
ubuntu@Ubuntu:~/destFolder$
```

9. Command: **chmod permissions file.txt**

Description: Changes the permissions of "file.txt" (e.g., `chmod 755 file.txt`).

Output:

```
ubuntu@Ubuntu:~/destFolder$ ls -l
total 12
-rw-rw-r-- 1 ubuntu ubuntu 17 Aug 19 11:34 file1.txt
-rw-rw-r-- 1 ubuntu ubuntu 17 Aug 19 11:34 file2.txt
-rw-rw-r-- 1 ubuntu ubuntu 21 Aug 19 11:31 main.py
ubuntu@Ubuntu:~/destFolder$ chmod 777 file1.txt
ubuntu@Ubuntu:~/destFolder$ ls -l
total 12
-rwxrwxrwx 1 ubuntu ubuntu 17 Aug 19 11:34 file1.txt
-rw-rw-r-- 1 ubuntu ubuntu 17 Aug 19 11:34 file2.txt
-rw-rw-r-- 1 ubuntu ubuntu 21 Aug 19 11:31 main.py
ubuntu@Ubuntu:~/destFolder$
```

10. List of running processes:

Command: **ps aux**

Description: Displays a list of all running processes.

Output:

```
root@ubuntu:/home/ubuntu/Desktop# ps -ef | grep firefox
ubuntu      4589      1736  11 12:22 ?        00:00:09 /snap/firefox/2987/usr/lib/firefox/firefox
ubuntu      4726      4589   0 12:22 ?        00:00:00 /snap/firefox/2987/usr/lib/firefox/firefox -contentproc -parentBuildID 202
30807082803 -prefsLen 27555 -prefMapSize 232220 -appDir /snap/firefox/2987/usr/lib/firefox/browser {25143190-5394-4d50-b5c9-3a
14bbbceeb4} 4589 true socket
ubuntu      4738      4589   2 12:22 ?        00:00:01 /snap/firefox/2987/usr/lib/firefox/firefox -contentproc -childID 1 -isForB
rowser -prefsLen 27741 -prefMapSize 232220 -jsInitLen 242416 -parentBuildID 20230807082803 -appDir /snap/firefox/2987/usr/lib/
firefox/browser {b28bdf06-4141-45da-bc2f-fb6e50cfad1b} 4589 true tab
ubuntu      4901      4589   0 12:22 ?        00:00:00 /snap/firefox/2987/usr/lib/firefox/firefox -contentproc -childID 2 -isForB
rowser -prefsLen 33391 -prefMapSize 232220 -jsInitLen 242416 -parentBuildID 20230807082803 -appDir /snap/firefox/2987/usr/lib/
firefox/browser {ea9aa98a-e91f-4700-a613-1c60b0430859} 4589 true tab
ubuntu      5182      4589   0 12:22 ?        00:00:00 /snap/firefox/2987/usr/lib/firefox/firefox -contentproc -childID 3 -isForB
rowser -prefsLen 28725 -prefMapSize 232220 -jsInitLen 242416 -parentBuildID 20230807082803 -appDir /snap/firefox/2987/usr/lib/
firefox/browser {3718bed3-7759-46a2-a255-e19d69531056} 4589 true tab
ubuntu      5189      4589   0 12:22 ?        00:00:00 /snap/firefox/2987/usr/lib/firefox/firefox -contentproc -childID 4 -isForB
rowser -prefsLen 28725 -prefMapSize 232220 -jsInitLen 242416 -parentBuildID 20230807082803 -appDir /snap/firefox/2987/usr/lib/
firefox/browser {b69f73b3-8e3e-4dad-8b22-254e9a69ca21} 4589 true tab
ubuntu      5206      4589   0 12:22 ?        00:00:00 /snap/firefox/2987/usr/lib/firefox/firefox -contentproc -childID 5 -isForB
rowser -prefsLen 28725 -prefMapSize 232220 -jsInitLen 242416 -parentBuildID 20230807082803 -appDir /snap/firefox/2987/usr/lib/
firefox/browser {f07a8ba4-28e0-4e1c-a754-238aefc0d927} 4589 true tab
```

11. Kill a process:

Command: **kill PID**

Description: Terminates the process with the specified Process ID (PID).

Output:

```
root@ubuntu:/home/ubuntu/Desktop# ps -ef | grep firefox
ubuntu      4589      1736  11 12:22 ?        00:00:09 /snap/firefox/2987/usr/lib/firefox/firefox
ubuntu      4726      4589   0 12:22 ?        00:00:00 /snap/firefox/2987/usr/lib/firefox/firefox -contentproc -parentBuildID 202
30807082803 -prefsLen 27555 -prefMapSize 232220 -appDir /snap/firefox/2987/usr/lib/firefox/browser {25143190-5394-4d50-b5c9-3a
14bbbceeb4} 4589 true socket
ubuntu      4738      4589   2 12:22 ?        00:00:01 /snap/firefox/2987/usr/lib/firefox/firefox -contentproc -childID 1 -isForB
rowser -prefsLen 27741 -prefMapSize 232220 -jsInitLen 242416 -parentBuildID 20230807082803 -appDir /snap/firefox/2987/usr/lib/
firefox/browser {b28bdf06-4141-45da-bc2f-fb6e50cfad1b} 4589 true tab
ubuntu      4901      4589   0 12:22 ?        00:00:00 /snap/firefox/2987/usr/lib/firefox/firefox -contentproc -childID 2 -isForB
rowser -prefsLen 33391 -prefMapSize 232220 -jsInitLen 242416 -parentBuildID 20230807082803 -appDir /snap/firefox/2987/usr/lib/
firefox/browser {ea9aa98a-e91f-4700-a613-1c60b0430859} 4589 true tab
ubuntu      5182      4589   0 12:22 ?        00:00:00 /snap/firefox/2987/usr/lib/firefox/firefox -contentproc -childID 3 -isForB
rowser -prefsLen 28725 -prefMapSize 232220 -jsInitLen 242416 -parentBuildID 20230807082803 -appDir /snap/firefox/2987/usr/lib/
firefox/browser {3718bed3-7759-46a2-a255-e19d69531056} 4589 true tab
ubuntu      5189      4589   0 12:22 ?        00:00:00 /snap/firefox/2987/usr/lib/firefox/firefox -contentproc -childID 4 -isForB
rowser -prefsLen 28725 -prefMapSize 232220 -jsInitLen 242416 -parentBuildID 20230807082803 -appDir /snap/firefox/2987/usr/lib/
firefox/browser {b69f73b3-8e3e-4dad-8b22-254e9a69ca21} 4589 true tab
ubuntu      5206      4589   0 12:22 ?        00:00:00 /snap/firefox/2987/usr/lib/firefox/firefox -contentproc -childID 5 -isForB
rowser -prefsLen 28725 -prefMapSize 232220 -jsInitLen 242416 -parentBuildID 20230807082803 -appDir /snap/firefox/2987/usr/lib/
firefox/browser {f07a8ba4-28e0-4e1c-a754-238aefc0d927} 4589 true tab
```

12. Use of editors: vim, emacs, nano:

Description: You can use editors like Vim, Emacs, or Nano to edit text files directly from the terminal.

- Vim: **vim filename**

- Emacs: **emacs filename**

- Nano: **nano filename**

Output:

```
root@Ubuntu:/home/ubuntu# cd destFolder/
root@Ubuntu:/home/ubuntu/destFolder# ls
file1.txt  file2.txt  main.py
root@Ubuntu:/home/ubuntu/destFolder# apt install vim
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  vim-runtime
Suggested packages:
  ctags vim-doc vim-scripts
The following NEW packages will be installed:
  vim vim-runtime
0 upgraded, 2 newly installed, 0 to remove and 23 not upgraded.
Need to get 8,559 kB of archives.
After this operation, 37.6 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 vim-runtime all 2:8.2.3995-1ubuntu2.10 [6,827 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 vim amd64 2:8.2.3995-1ubuntu2.10 [1,732 kB]
Fetched 8,559 kB in 4s (2,119 kB/s)
Selecting previously unselected package vim-runtime.
(Reading database ... 167806 files and directories currently installed.)
Preparing to unpack .../vim-runtime_2%3a8.2.3995-1ubuntu2.10_all.deb ...
Adding 'diversion of /usr/share/vim/vim82/doc/help.txt to /usr/share/vim/vim82/doc/help.txt.vim-tiny by vim-runtime'
Adding 'diversion of /usr/share/vim/vim82/doc/tags to /usr/share/vim/vim82/doc/tags.vim-tiny by vim-runtime'
Unpacking vim-runtime (2:8.2.3995-1ubuntu2.10) ...
Selecting previously unselected package vim.
Preparing to unpack .../vim_2%3a8.2.3995-1ubuntu2.10_amd64.deb ...
Unpacking vim (2:8.2.3995-1ubuntu2.10) ...
Setting up vim-runtime (2:8.2.3995-1ubuntu2.10) ...
Setting up vim (2:8.2.3995-1ubuntu2.10) ...
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vim (vim) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vimdiff (vimdiff) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/rvim (rvim) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/rview (rview) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vi (vi) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/view (view) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/ex (ex) in auto mode
Processing triggers for man-db (2.10.2-1) ...
root@Ubuntu:/home/ubuntu/destFolder# vim newFile.txt
root@Ubuntu:/home/ubuntu/destFolder# cat newFile.txt
this is my content in the file which I am creating using vim text editor.
```

2) Write a C program for the logic GATE operation

```
#include <stdio.h>
// Logic gate functions
int AND(int a, int b) {
    return a && b;
}

int OR(int a, int b) {
    return a || b;
}

int NOT(int a) {
    return !a;
}

int XOR(int a, int b) {
    return (a || b) && !(a && b);
}

int NAND(int a, int b) {
    return !(a && b);
}

int main() {
    int a, b;
    int choice;

    printf("Enter values (0 or 1) for a and b: ");
    scanf("%d %d", &a, &b);

    printf("Select an operation:\n");
    printf("1. AND\n");
    printf("2. OR\n");
    printf("3. NOT a and NOT b\n");
    printf("4. XOR\n");
    printf("5. NAND\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

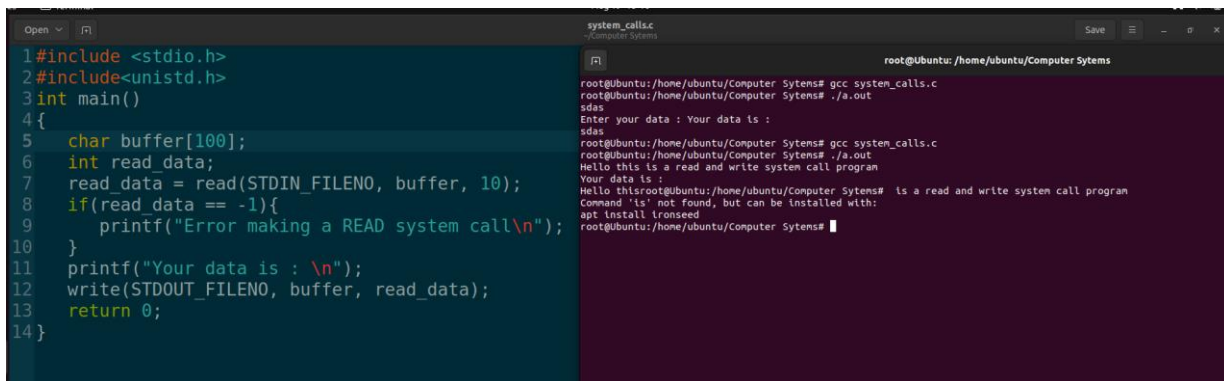
    switch (choice) {
        case 1:
            printf("AND: %d\n", AND(a, b));
            break;
        case 2:
            printf("OR: %d\n", OR(a, b));
            break;
        case 3:
            printf("NOT a: %d, NOT b: %d\n", NOT(a), NOT(b));
```

```
        break;
    case 4:
        printf("XOR: %d\n", XOR(a, b));
        break;
    case 5:
        printf("NAND: %d\n", NAND(a, b));
        break;
    default:
        printf("Invalid choice\n");
        break;
}

return 0;
}
```


3) C programs for system calls

a. Read() & Write()

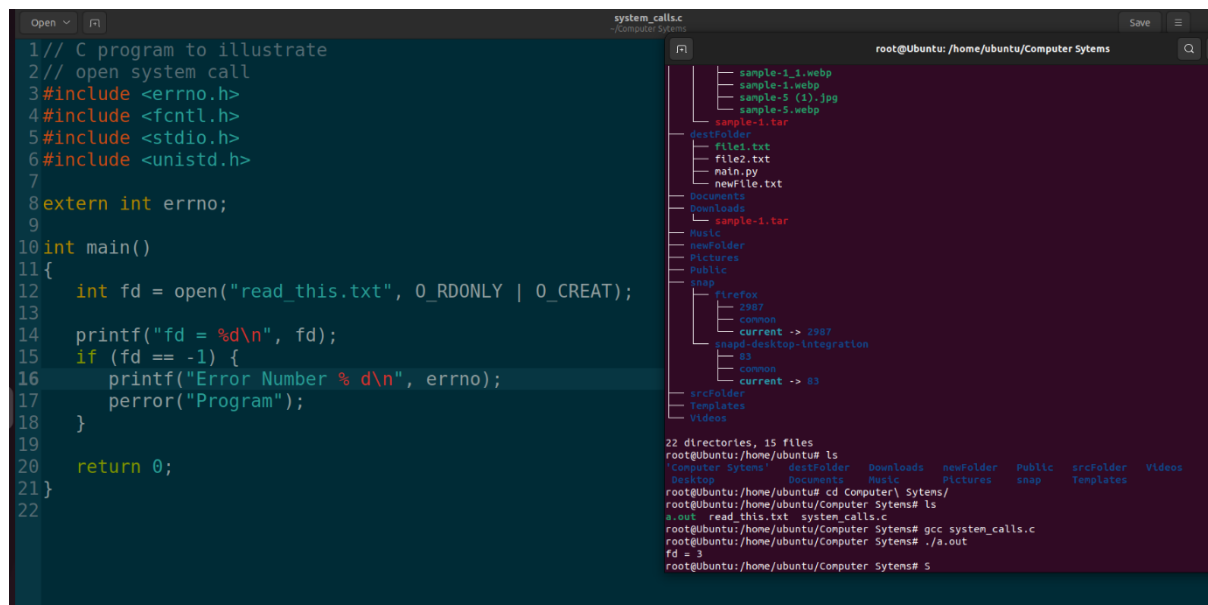


The screenshot shows a C program named `system_calls.c` in a text editor and its execution in a terminal. The program uses `read()` and `write()` system calls. It prompts the user for data, reads it, and then writes it back. The terminal output shows the user entering 'sdas' and the program responding with 'Hello this is a read and write system call program'.

```
1#include <stdio.h>
2#include<unistd.h>
3int main()
4{
5    char buffer[100];
6    int read_data;
7    read_data = read(STDIN_FILENO, buffer, 10);
8    if(read_data == -1){
9        printf("Error making a READ system call\n");
10    }
11    printf("Your data is : \n");
12    write(STDOUT_FILENO, buffer, read_data);
13    return 0;
14}
```

```
root@Ubuntu: /home/ubuntu/Computer Sytems
root@Ubuntu:/home/ubuntu/Computer Sytems# gcc system_calls.c
root@Ubuntu:/home/ubuntu/Computer Sytems# ./a.out
sdas
Enter your data : Your data is :
sdas
root@Ubuntu:/home/ubuntu/Computer Sytems# gcc system_calls.c
root@Ubuntu:/home/ubuntu/Computer Sytems# ./a.out
Hello this is a read and write system call program
Your data is :
sdas
root@Ubuntu:/home/ubuntu/Computer Sytems#
```

2. Open()

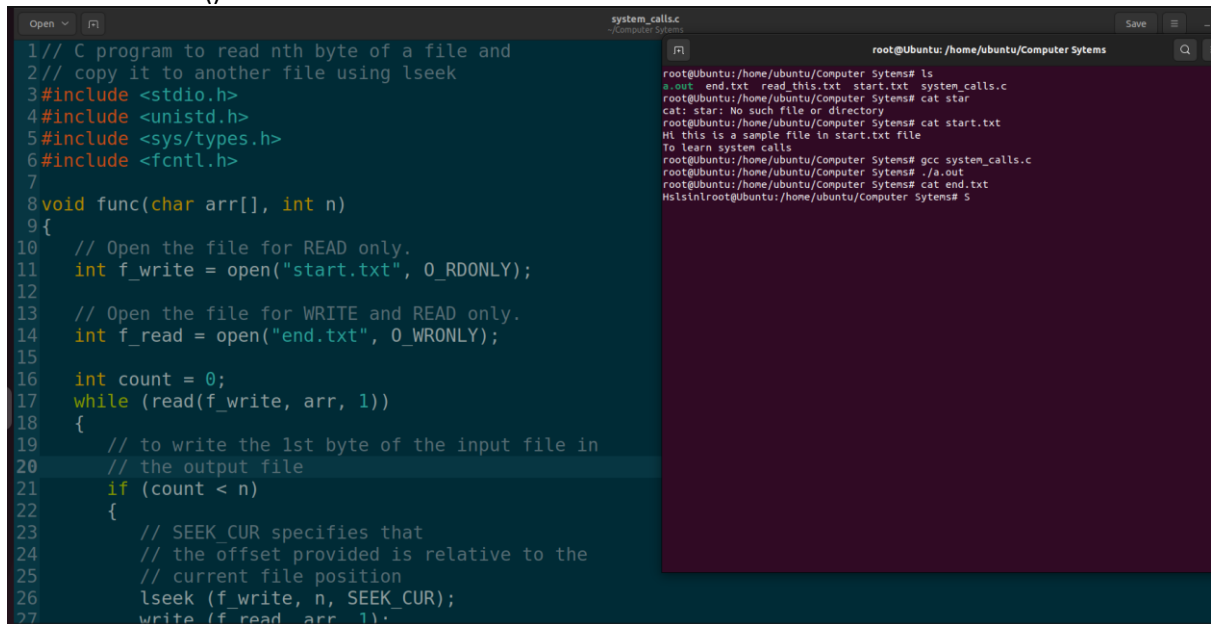


The screenshot shows a C program named `system_calls.c` in a text editor and its execution in a terminal. The program uses `open()` system call to create a file. It prompts the user for a file name, opens it, and then prints the file descriptor. The terminal output shows the user entering 'read_this.txt' and the program responding with 'fd = 3'.

```
1// C program to illustrate
2// open system call
3#include <errno.h>
4#include <fcntl.h>
5#include <stdio.h>
6#include <unistd.h>
7
8extern int errno;
9
10int main()
11{
12    int fd = open("read_this.txt", O_RDONLY | O_CREAT);
13
14    printf("fd = %d\n", fd);
15    if (fd == -1) {
16        printf("Error Number %d\n", errno);
17        perror("Program");
18    }
19
20    return 0;
21}
22
```

```
root@Ubuntu:/home/ubuntu/Computer Sytems# ls
'Computer Sytems'  destFolder  Downloads  newFolder  Public  srcFolder  Videos
Desktop            Documents  Music      Pictures   snap    Templates
root@Ubuntu:/home/ubuntu/Computer Sytems# cd Computer Sytems/
root@Ubuntu:/home/ubuntu/Computer Sytems# ls
a.out  read_this.txt  system_calls.c
root@Ubuntu:/home/ubuntu/Computer Sytems# gcc system_calls.c
root@Ubuntu:/home/ubuntu/Computer Sytems# ./a.out
fd = 3
root@Ubuntu:/home/ubuntu/Computer Sytems#
```

3. Lseek()

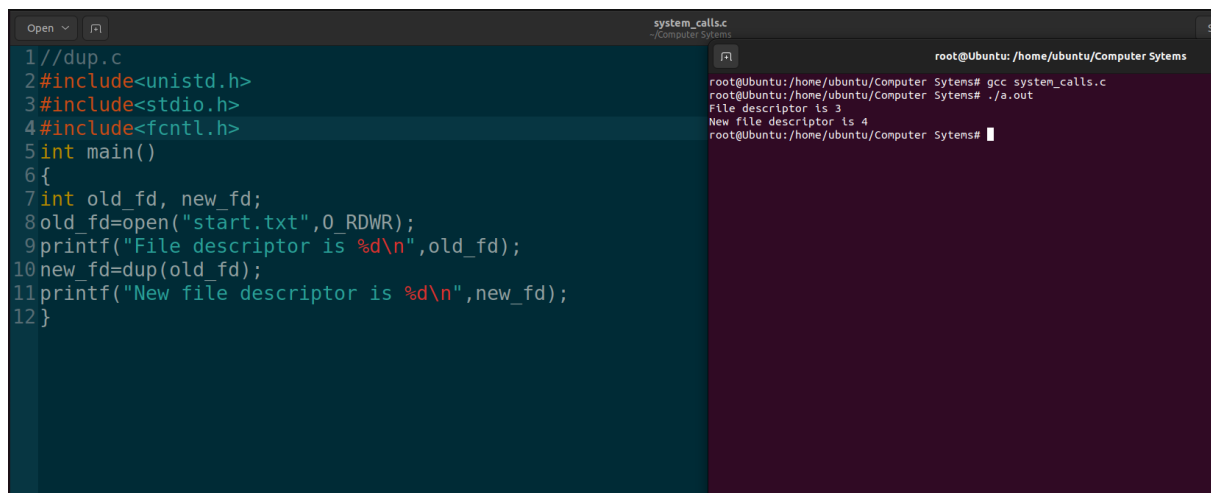


The screenshot shows a code editor with a C program named `system_calls.c` and a terminal window. The C program implements a function `func` that reads a file `start.txt` and writes its contents to `end.txt` using `lseek` to move the file pointer to the current position (`SEEK_CUR`) before writing. The terminal shows the execution of the program, including the compilation with `gcc` and the execution of the resulting binary `./a.out`.

```
1// C program to read nth byte of a file and
2// copy it to another file using lseek
3#include <stdio.h>
4#include <unistd.h>
5#include <sys/types.h>
6#include <fcntl.h>
7
8void func(char arr[], int n)
9{
10    // Open the file for READ only.
11    int f_write = open("start.txt", O_RDONLY);
12
13    // Open the file for WRITE and READ only.
14    int f_read = open("end.txt", O_WRONLY);
15
16    int count = 0;
17    while (read(f_write, arr, 1))
18    {
19        // to write the 1st byte of the input file in
20        // the output file
21        if (count < n)
22        {
23            // SEEK_CUR specifies that
24            // the offset provided is relative to the
25            // current file position
26            lseek (f_write, n, SEEK_CUR);
27            write (f_read, arr, 1);
28        }
29        count++;
30    }
31}
```

```
root@Ubuntu:/home/ubuntu/Computer Systems# ls
a.out end.txt read this.txt start.txt system_calls.c
root@Ubuntu:/home/ubuntu/Computer Systems# cat start.txt
cat: start: No such file or directory
root@Ubuntu:/home/ubuntu/Computer Systems# cat start.txt
Hi this is a sample file in start.txt file
To learn system calls
root@Ubuntu:/home/ubuntu/Computer Systems# gcc system_calls.c
root@Ubuntu:/home/ubuntu/Computer Systems# ./a.out
root@Ubuntu:/home/ubuntu/Computer Systems# cat end.txt
Hs1stnroot@Ubuntu:/home/ubuntu/Computer Systems#
```

4. Dup()



The screenshot shows a code editor with a C program named `dup.c` and a terminal window. The C program demonstrates the use of `dup` to create a duplicate of an existing file descriptor. It opens `start.txt` and then duplicates the file descriptor to create a new one. The terminal shows the compilation with `gcc` and the execution of the resulting binary `./a.out`, which prints the file descriptors.

```
1//dup.c
2#include<unistd.h>
3#include<stdio.h>
4#include<fcntl.h>
5int main()
6{
7    int old_fd, new_fd;
8    old_fd=open("start.txt",O_RDWR);
9    printf("File descriptor is %d\n",old_fd);
10    new_fd=dup(old_fd);
11    printf("New file descriptor is %d\n",new_fd);
12}
```

```
root@Ubuntu:/home/ubuntu/Computer Systems# gcc dup.c
root@Ubuntu:/home/ubuntu/Computer Systems# ./a.out
File descriptor is 3
New file descriptor is 4
root@Ubuntu:/home/ubuntu/Computer Systems#
```

5. Fork()

```
1
2 #define _POSIX_SOURCE
3 #include <sys/types.h>
4 #include <stdio.h>
5 #include <unistd.h>
6 #include <stdlib.h>
7 #include <sys/wait.h>
8
9 int main() {
10     pid_t pid;
11     int status;
12
13     if ((pid = fork()) < 0)
14         perror("fork() error");
15     else if (pid == 0) {
16         puts("This is the child.");
17         printf("Child's pid is %d and my parent's is %d\n",
18             (int) getpid(), (int) getppid());
19         exit(42);
20     }
21     else {
22         puts("This is the parent.");
23         printf("Parent's pid is %d and my child's is %d\n",
24             (int) getpid(), (int) pid);
25         puts("I'm waiting for my child to complete.");
26         if (wait(&status) == -1)
27             perror("wait() error");
28     }
29 }
```

root@Ubuntu:/home/ubuntu/Computer Systems# gcc system_calls.c
root@Ubuntu:/home/ubuntu/Computer Systems# ./a.out
File descriptor is 3
New file descriptor is 4
root@Ubuntu:/home/ubuntu/Computer Systems# ls
a.out end.txt read_this.txt start.txt system_calls.c
root@Ubuntu:/home/ubuntu/Computer Systems# gcc system_calls.c
system_calls.c:13:1: warning: return type defaults to 'int' [-Wimplicit-int]
13 | main() {
 | ^~~~~~
root@Ubuntu:/home/ubuntu/Computer Systems# gcc system_calls.c
root@Ubuntu:/home/ubuntu/Computer Systems# ./a.out
This is the parent.
Parent's pid is 23745 and my child's is 23746
I'm waiting for my child to complete.
This is the child.
Child's pid is 23746 and my parent's is 23745
The child exited with status of 42
root@Ubuntu:/home/ubuntu/Computer Systems#

6. Orphan process

```
1 // A C program to demonstrate Orphan Process.
2 // Parent process finishes execution while the
3 // child process is running. The child process
4 // becomes orphan.
5 #include <stdio.h>
6 #include <sys/types.h>
7 #include <unistd.h>
8
9 int main()
10 {
11     // Create a child process
12     int pid = fork();
13
14     if (pid > 0)
15         printf("in parent process");
16
17     // Note that pid is 0 in child process
18     // and negative if fork() fails
19     else if (pid == 0)
20     {
21         sleep(30);
22         printf("in child process");
23     }
24
25     return 0;
26 }
27
```

root@Ubuntu:/home/ubuntu/Computer Systems# gcc system_calls.c
root@Ubuntu:/home/ubuntu/Computer Systems# ./a.out
File descriptor is 3
New file descriptor is 4
root@Ubuntu:/home/ubuntu/Computer Systems# ls
a.out end.txt read_this.txt start.txt system_calls.c
root@Ubuntu:/home/ubuntu/Computer Systems# gcc system_calls.c
system_calls.c:13:1: warning: return type defaults to 'int' [-Wimplicit-int]
13 | main() {
 | ^~~~~~
root@Ubuntu:/home/ubuntu/Computer Systems# gcc system_calls.c
root@Ubuntu:/home/ubuntu/Computer Systems# ./a.out
This is the parent.
Parent's pid is 23745 and my child's is 23746
I'm waiting for my child to complete.
This is the child.
Child's pid is 23746 and my parent's is 23745
The child exited with status of 42
root@Ubuntu:/home/ubuntu/Computer Systems# gcc system_calls.c
root@Ubuntu:/home/ubuntu/Computer Systems# ./a.out
In parent processroot@Ubuntu:/home/ubuntu/Computer Systems#