Bahir Dar University

Software engineering department

System Call Implementation

Name: Dagmawit Amsalu

ID No.1601218

Course: Operating system and system programming

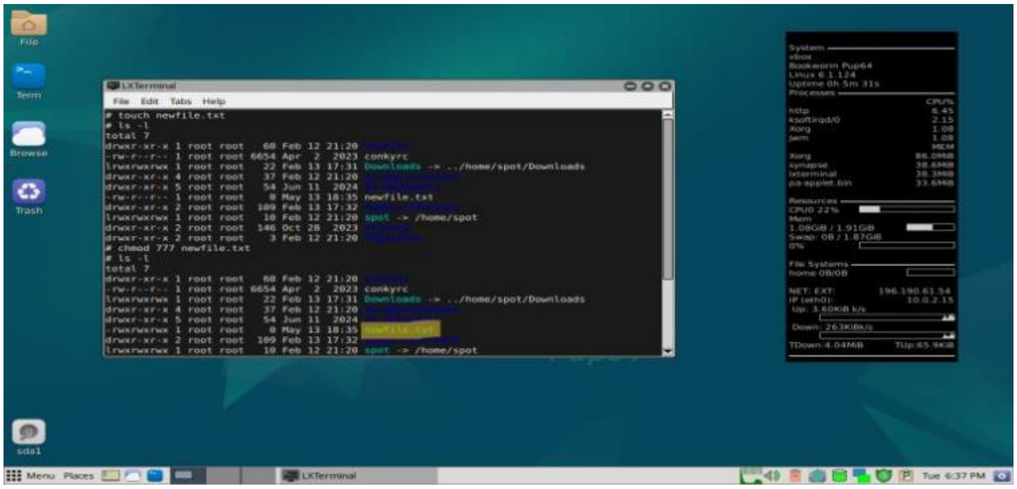
Submitted To: lecturer Wendimu B.

Implementing a system call

A system call is how a program talks to the operating system.Programs can't do important things like opening files, reading from the keyboard, or using the internet directly. They have to ask the operating system (like Linux) to do it for them. That asking is done using system calls. Common System Calls:

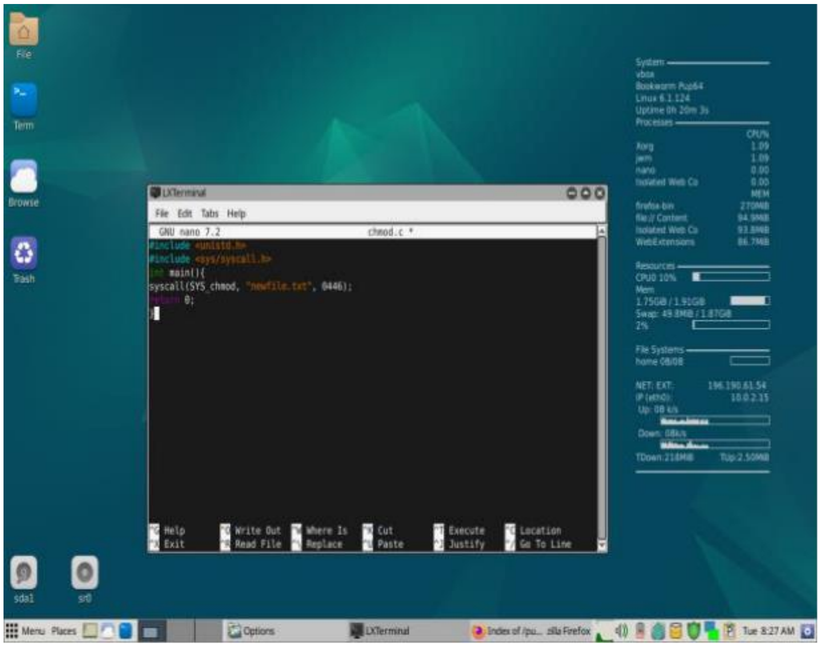
- open() – open a file
- read() – read from a file
- write() – write to a file
- close() – close a file
- fork() – create a new process
- exit() – end a program

The system call i will be demonstrating is chmod() changing the permission of a file.

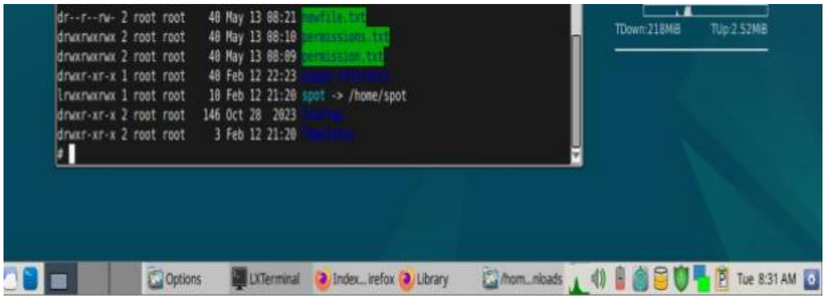


```
# touch newfile.txt
# ls -l
total 7
drwxr-xr-x 1 root root   60 Feb 12 21:20 [illegible]
-rw-r--r-- 1 root root 6654 Apr  2  2023 conkyrc
lrwxrwxrwx 1 root root   22 Feb 13 17:31 Downloads -> ../home/spot/Downloads
drwxr-xr-x 4 root root   37 Feb 12 21:20 [illegible]
drwxr-xr-x 5 root root   54 Jun 11  2024 [illegible]
-rw-r--r-- 1 root root    0 May 13 18:35 newfile.txt
drwxr-xr-x 2 root root  109 Feb 13 17:32 [illegible]
lrwxrwxrwx 1 root root   10 Feb 12 21:20 spot -> /home/spot
drwxr-xr-x 2 root root  146 Oct 28  2023 [illegible]
drwxr-xr-x 2 root root    3 Feb 12 21:20 [illegible]
# chmod 777 newfile.txt
# ls -l
total 7
drwxr-xr-x 1 root root   60 Feb 12 21:20 [illegible]
-rw-r--r-- 1 root root 6654 Apr  2  2023 conkyrc
lrwxrwxrwx 1 root root   22 Feb 13 17:31 Downloads -> ../home/spot/Downloads
drwxr-xr-x 4 root root   37 Feb 12 21:20 [illegible]
drwxr-xr-x 5 root root   54 Jun 11  2024 [illegible]
-rwxrwxrwx 1 root root    0 May 13 18:35 newfile.txt
drwxr-xr-x 2 root root  109 Feb 13 17:32 [illegible]
lrwxrwxrwx 1 root root   10 Feb 12 21:20 spot -> /home/spot
```

```
System
vbox
Bookworm Pup64
Linux 6.1.124
Uptime 0h 5m 31s
Processes
                    CPU%
http                6.45
ksoftirqd/0         2.15
Xorg                1.08
jwm                 1.08
                     MEM
Xorg              86.0MiB
synapse           38.6MiB
lxterminal        38.3MiB
pa-applet.bin     33.6MiB
Resources
CPU0 22%
Mem
1.08GiB / 1.91GiB
Swap: 0B / 1.87GiB
0%
File Systems
home 0B/0B
NET: EXT:      196.190.61.54
IP (eth0):         10.0.2.15
 Up: 3.60KiB k/s
 Down: 263KiBk/s
TDown:4.04MiB    TUp:65.9KiB
```



```
GNU nano 7.2                    chmod.c *
#include <unistd.h>
#include <sys/syscall.h>
int main(){
syscall(SYS_chmod, "newfile.txt", 0446);
return 0;
}
```



```
dr--r--rw- 2 root root    40 May 13 08:21 newfile.txt
drwxrwxrwx 2 root root    40 May 13 08:10 permissions.txt
drwxrwxrwx 2 root root    40 May 13 08:09 permission.txt
drwxr-xr-x 1 root root    40 Feb 12 22:23 [illegible]
lrwxrwxrwx 1 root root    10 Feb 12 21:20 spot -> /home/spot
drwxr-xr-x 2 root root   146 Oct 28  2023 [illegible]
drwxr-xr-x 2 root root     3 Feb 12 21:20 [illegible]
# 
```

This C program uses the syscall function to directly invoke the SYS_chmod system call, which changes the permissions of a file. The program attempts to modify the permissions of a fie named newfile.txt setting the permissions to 0446, which means the owner has read and write permissions, the group has read permissions and the others also have read permissions,The system call is part of the low-level interface provided by the operating system, bypassing higher-level C library functions like chmod(). This demonstrates a more manual and low-level way of manipulating file permissions in Unix-like systems.