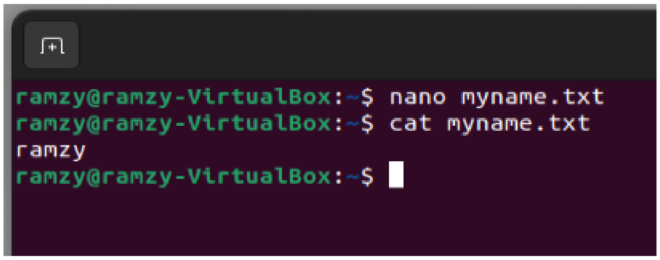
Name : Ramzy Izza Wardhana
NIM : 21/472698/PA/20322
Class : IUP CS B – Computer Network and OS Lab

Assignment 3 – Introduction to Linux and System Calls

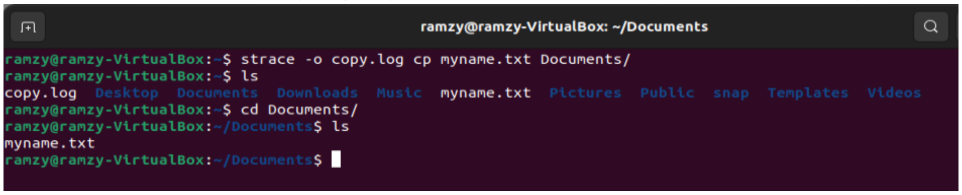
1. In your home directory, create a text file titled **myname.txt** and write down your name inside it. Afterwards, execute and save the strace results for the copy (**cp**) command to copy the myname.txt file from your home directory to the **Documents** directory. Save the results of the strace command in a file titled **copy.log**. Then, filter the **copy.log** file to find the **Documents** path inside. Explains what the system calls in the filtering results did during the copy command. Please include the screenshots of what you did.

   a. Create *myname.txt* and write my name inside it



```
ramzy@ramzy-VirtualBox:~$ nano myname.txt
ramzy@ramzy-VirtualBox:~$ cat myname.txt
ramzy
ramzy@ramzy-VirtualBox:~$ 
```

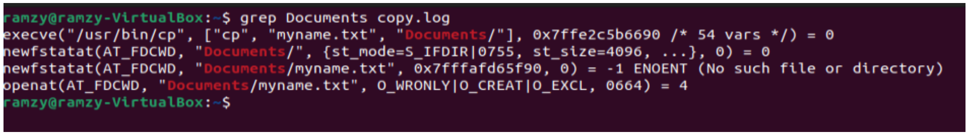
   b. Strace the result of copying myname.txt to the Documents directory



```
ramzy@ramzy-VirtualBox:~$ strace -o copy.log cp myname.txt Documents/
ramzy@ramzy-VirtualBox:~$ ls
copy.log  Desktop  Documents  Downloads  Music  myname.txt  Pictures  Public  snap  Templates  Videos
ramzy@ramzy-VirtualBox:~$ cd Documents/
ramzy@ramzy-VirtualBox:~/Documents$ ls
myname.txt
ramzy@ramzy-VirtualBox:~/Documents$ 
```

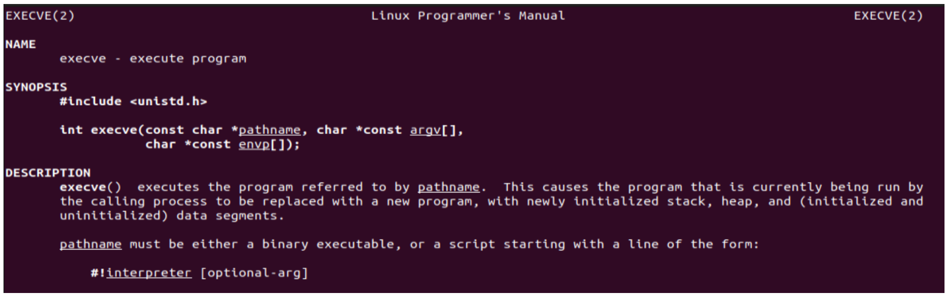
   Note that copy.log containing strace results is saved in home directory, and the copied file of myname.txt is saved in Documents/ directory

   c. Filter the copy.log file using grep to find the keyword Documents path and explain the system calls in the filtering process



```
ramzy@ramzy-VirtualBox:~$ grep Documents copy.log
execve("/usr/bin/cp", ["cp", "myname.txt", "Documents/"], 0x7ffe2c5b6690 /* 54 vars */) = 0
newfstatat(AT_FDCWD, "Documents/", {st_mode=S_IFDIR|0755, st_size=4096, ...}, 0) = 0
newfstatat(AT_FDCWD, "Documents/myname.txt", 0x7fffafd65f90, 0) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "Documents/myname.txt", O_WRONLY|O_CREAT|O_EXCL, 0664) = 4
ramzy@ramzy-VirtualBox:~$
```

   By using manpages-dev, we may explain these system calls:



```
EXECVE(2)                     Linux Programmer's Manual                     EXECVE(2)

NAME
       execve - execute program

SYNOPSIS
       #include <unistd.h>

       int execve(const char *pathname, char *const argv[],
                  char *const envp[]);

DESCRIPTION
       execve()  executes the program referred to by pathname.  This causes the program that is currently being run by
       the calling process to be replaced with a new program, with newly initialized stack, heap, and (initialized and
       uninitialized) data segments.

       pathname must be either a binary executable, or a script starting with a line of the form:

           #!interpreter [optional-arg]
```

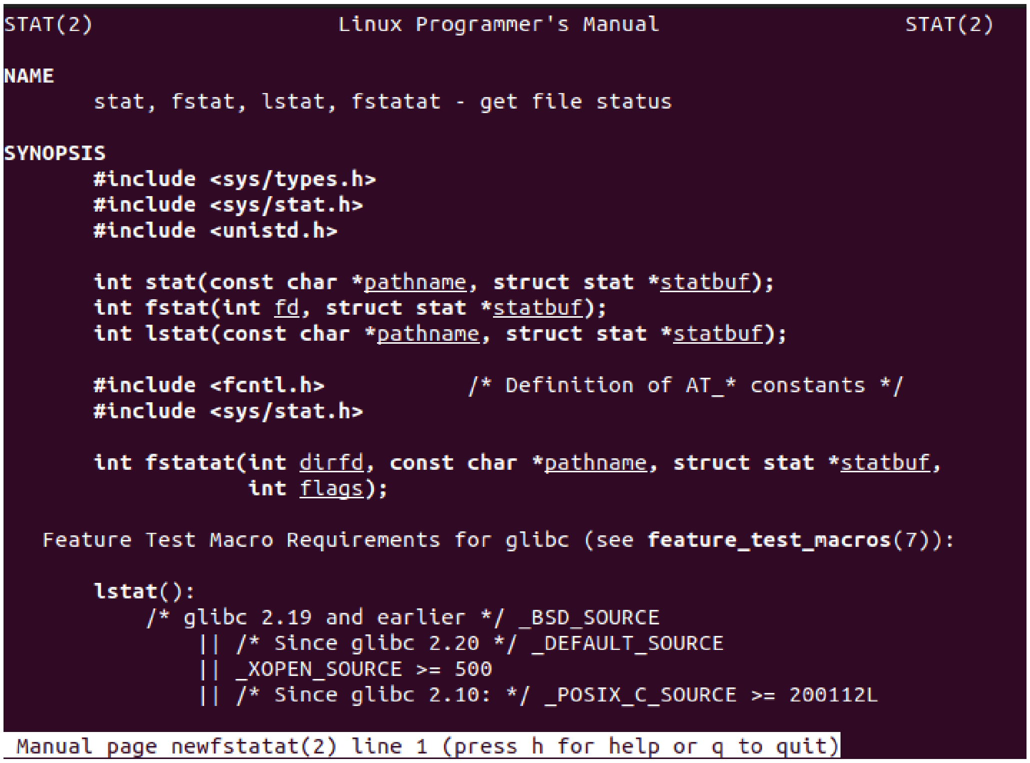
execve("/usr/bin/cp", ["cp", "myname.txt", "Documents/"], 0x7ffc7943e530 /* 53 vars */) = 0

/usr/bin/cp (first parameter): denotes the pathname that we want to execute

myname.txt (second parameter): contains the filename associated with the file that are being executed.

Documents/ (third parameter): array of pointers to strings used to pass by as a new environment for the new program.

In shorts, execve() is a system calls that is used to execute a program that corresponds to the pathname given.



```
STAT(2)                    Linux Programmer's Manual                   STAT(2)

NAME
       stat, fstat, lstat, fstatat - get file status

SYNOPSIS
       #include <sys/types.h>
       #include <sys/stat.h>
       #include <unistd.h>

       int stat(const char *pathname, struct stat *statbuf);
       int fstat(int fd, struct stat *statbuf);
       int lstat(const char *pathname, struct stat *statbuf);

       #include <fcntl.h>           /* Definition of AT_* constants */
       #include <sys/stat.h>

       int fstatat(int dirfd, const char *pathname, struct stat *statbuf,
                   int flags);

   Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

       lstat():
           /* glibc 2.19 and earlier */ _BSD_SOURCE
               || /* Since glibc 2.20 */ _DEFAULT_SOURCE
               || _XOPEN_SOURCE >= 500
               || /* Since glibc 2.10: */ _POSIX_C_SOURCE >= 200112L

 Manual page newfstatat(2) line 1 (press h for help or q to quit)
```

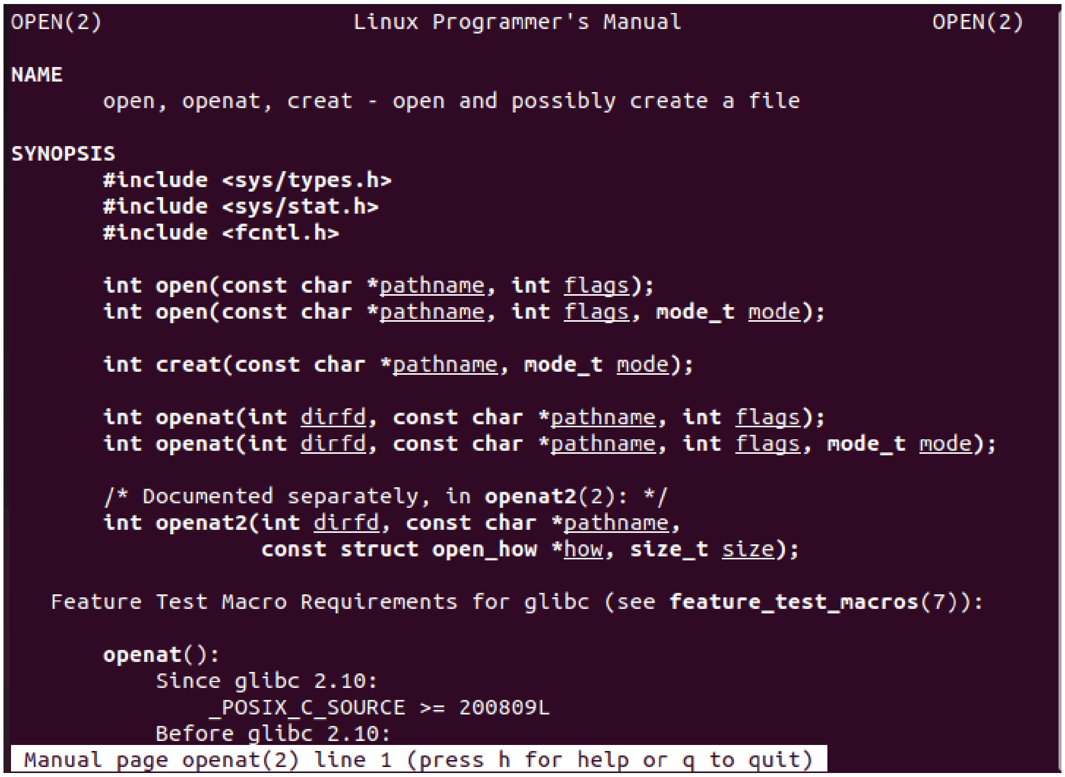
newfstatat(AT_FDCWD, "Documents/", {st_mode=S_IFDIR|0755, st_size=4096, ...}, 0) = 0

According to the man 2 newfstatat(), newfastatat() has 4 parameters namely:
int dirfd: **AT_FDCWD**
char *pathname: **Documents/**
struct stat *statbuf: **{st_mode=S_IFDIR|0755, st_size=4096,**
int flags: **0**

This system call is used to return the information about a file using the buffer pointed by statbuf. The fstatat() then will obtain the information and details about the file marked by the pathname in which the file is specified by the file descriptor fd.

newfstatat(AT_FDCWD, "Documents/myname.txt", 0x7ffcdc4ada60, 0) = -1 ENOENT (No such file or directory)

On the third system call line, it still using the newfstatat() but in this case, the function return -1. Meaning that it returns an error of ENOENT caused by no such file or directory detected



```
OPEN(2)                    Linux Programmer's Manual                   OPEN(2)

NAME
       open, openat, creat - open and possibly create a file

SYNOPSIS
       #include <sys/types.h>
       #include <sys/stat.h>
       #include <fcntl.h>

       int open(const char *pathname, int flags);
       int open(const char *pathname, int flags, mode_t mode);

       int creat(const char *pathname, mode_t mode);

       int openat(int dirfd, const char *pathname, int flags);
       int openat(int dirfd, const char *pathname, int flags, mode_t mode);

       /* Documented separately, in openat2(2): */
       int openat2(int dirfd, const char *pathname,
                   const struct open_how *how, size_t size);

   Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

       openat():
           Since glibc 2.10:
               _POSIX_C_SOURCE >= 200809L
           Before glibc 2.10:
 Manual page openat(2) line 1 (press h for help or q to quit)
```

openat(AT_FDCWD, "Documents/myname.txt",O_WRONLY|O_CREAT|O_EXCL, 0664) = 4

Moving on to the last system call that is openat() which accepts 4 paramaters mainly:

- int dirfd: **AT_FDCWD**
- char *pathname: **Documents/myname.txt**
- int flags: **O_WRONLY|O_CREAT|O_EXCL**
- mode_t mode: 0664

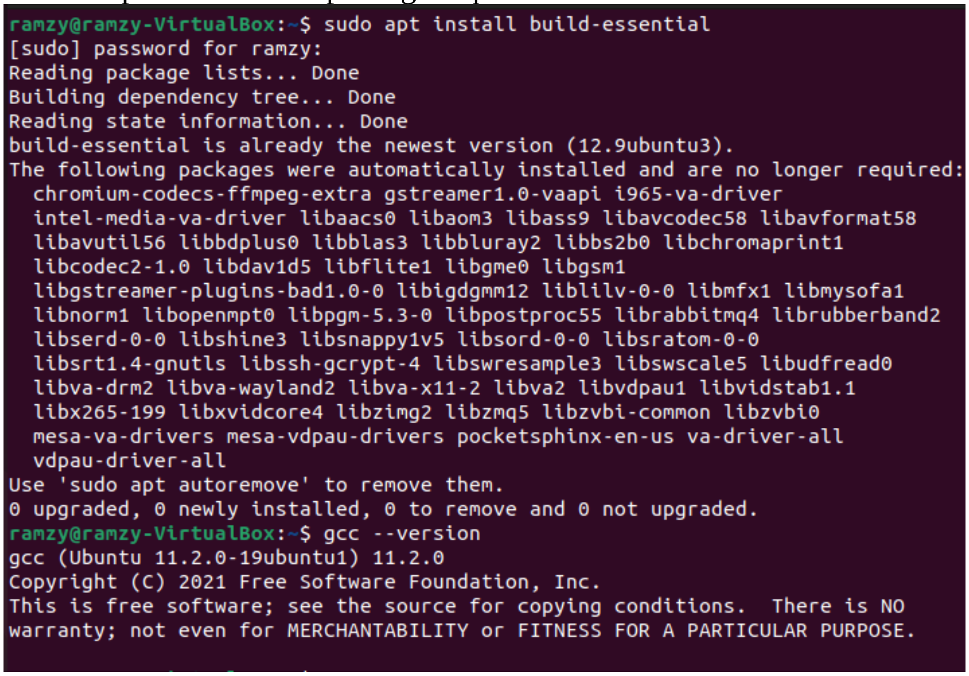
returning 4

openat() system call has the function to open a file specified by the pathname. In the case that the specified file does not exist, then it will created O_CREAT in flags. The return value of openat() is file descriptor. In addition, openat() will interpret the pathname that is relative to the directory referred by the file descriptor dirfd

Since the pathname is relative and dirfd has the special value **AT_FDCWD**, then pathname will be interpreted in relative to the current working directory.

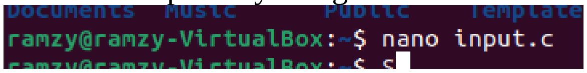
2. Create a program that utilizes system calls in the C programming language to accept an input from the keyboard and save it to a file titled output.txt. Write the source code inside the PDF and Include the screenshots during the compilation and execution of that program, as well as the content of the **output.txt** file.

a. Install dependencies and package required for C

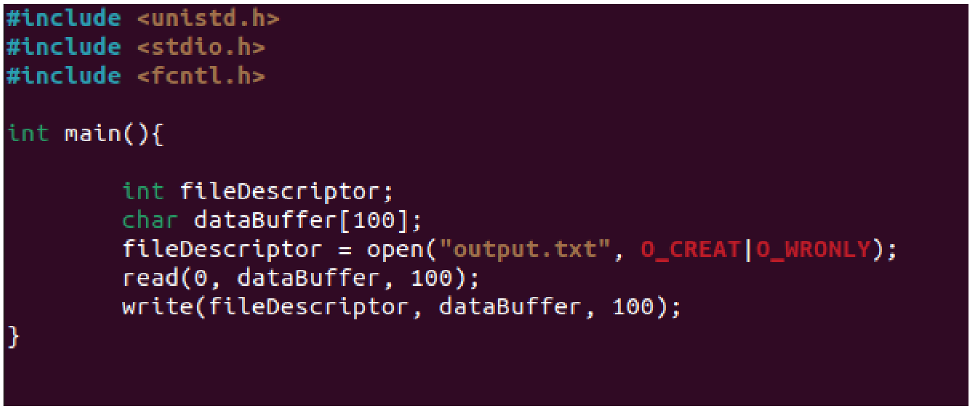


```
ramzy@ramzy-VirtualBox:~$ sudo apt install build-essential
[sudo] password for ramzy:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
build-essential is already the newest version (12.9ubuntu3).
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi i965-va-driver
  intel-media-va-driver libaacs0 libaom3 libass9 libavcodec58 libavformat58
  libavutil56 libbdplus0 libblas3 libbluray2 libbs2b0 libchromaprint1
  libcodec2-1.0 libdav1d5 libflite1 libgme0 libgsm1
  libgstreamer-plugins-bad1.0-0 libigdgmm12 liblilv-0-0 libmfx1 libmysofa1
  libnorm1 libopenmpt0 libpgm-5.3-0 libpostproc55 librabbitmq4 librubberband2
  libserd-0-0 libshine3 libsnappy1v5 libsord-0-0 libsratom-0-0
  libsrt1.4-gnutls libssh-gcrypt-4 libswresample3 libswscale5 libudfread0
  libva-drm2 libva-wayland2 libva-x11-2 libva2 libvdpau1 libvidstab1.1
  libx265-199 libxvidcore4 libzimg2 libzmq5 libzvbi-common libzvbi0
  mesa-va-drivers mesa-vdpau-drivers pocketsphinx-en-us va-driver-all
  vdpau-driver-all
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
ramzy@ramzy-VirtualBox:~$ gcc --version
gcc (Ubuntu 11.2.0-19ubuntu1) 11.2.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

b. Create the input.c by using nano and write the code

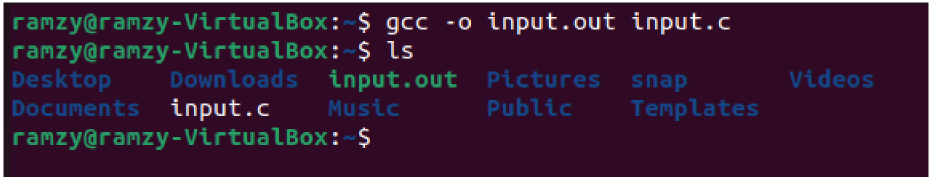


```
Documents  Music      Public     Template
ramzy@ramzy-VirtualBox:~$ nano input.c
ramzy@ramzy-VirtualBox:~$ s
```



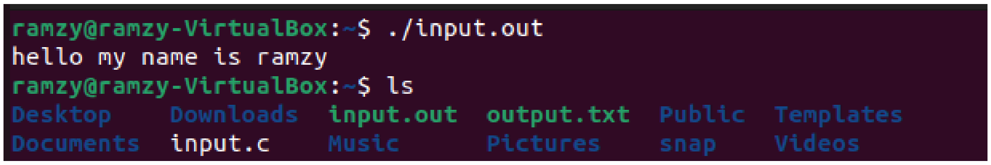
```
#include <unistd.h>
#include <stdio.h>
#include <fcntl.h>

int main(){

        int fileDescriptor;
        char dataBuffer[100];
        fileDescriptor = open("output.txt", O_CREAT|O_WRONLY);
        read(0, dataBuffer, 100);
        write(fileDescriptor, dataBuffer, 100);
}
```

c. Compile the program



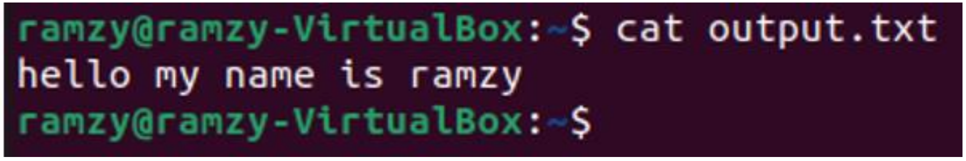
```
ramzy@ramzy-VirtualBox:~$ gcc -o input.out input.c
ramzy@ramzy-VirtualBox:~$ ls
Desktop    Downloads  input.out  Pictures  snap       Videos
Documents  input.c    Music      Public    Templates
ramzy@ramzy-VirtualBox:~$
```

d. Run the compiled program and input any text from keyboard then output to a file named *output.txt which* contains the input.



```
ramzy@ramzy-VirtualBox:~$ ./input.out
hello my name is ramzy
ramzy@ramzy-VirtualBox:~$ ls
Desktop    Downloads  input.out  output.txt  Public  Templates
Documents  input.c    Music      Pictures    snap    Videos
```

e. Content of *output.txt*



```
ramzy@ramzy-VirtualBox:~$ cat output.txt
hello my name is ramzy
ramzy@ramzy-VirtualBox:~$
```

Source Code:

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

int main(){

        int fileDescriptor;
        char dataBuffer[100];
        fileDescriptor = open("output.txt", O_CREAT|O_WRONLY);
        read(0, dataBuffer, 100);
        write(fileDescriptor, dataBuffer, 100);

}
```