System Software (CS306)

Assignment - 1

**U19CS012**

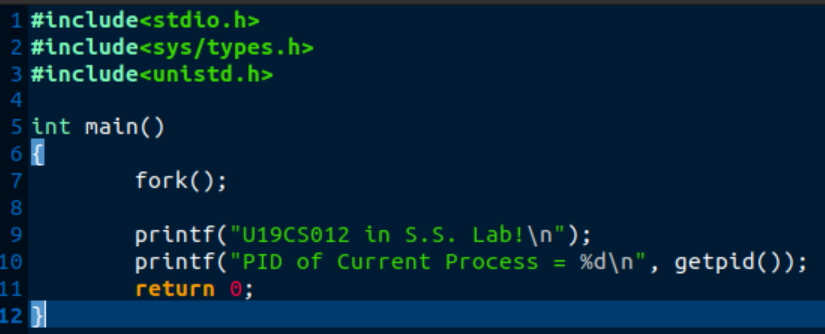
**Aim**: To study the **Basics of** **System Call** and **System Library**.

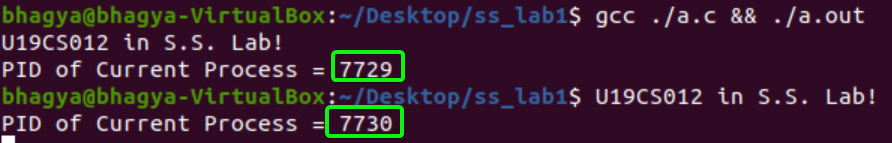
System Calls:

A.) **fork()** [Process Control System Call]

* **Purpose**: Used to **create a new separate & duplicate process** [Child Process] which runs concurrently with the parent process.
* On success, the **PID of the child process** is returned in the parent, and **0** is returned in the child.
* On failure, -**1** is returned to the parent, no child process is created, and err\_no is set appropriately.

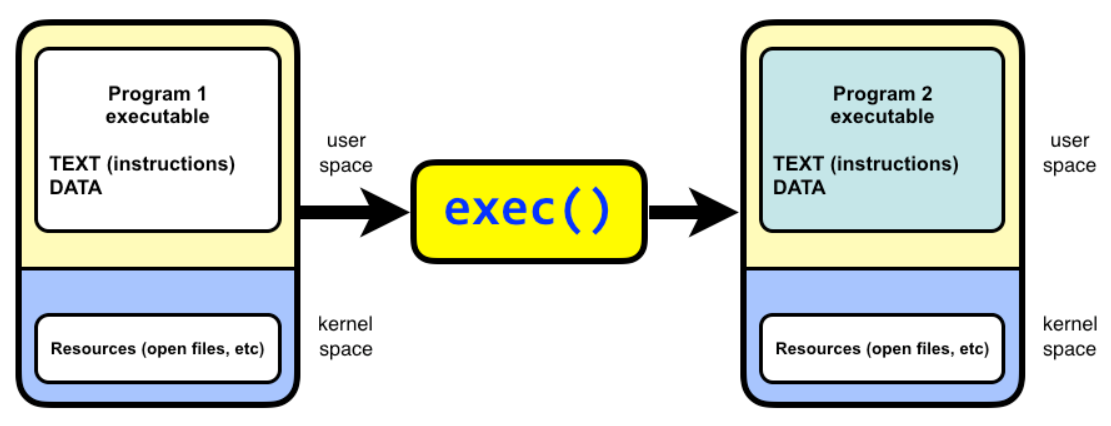
**Example**:



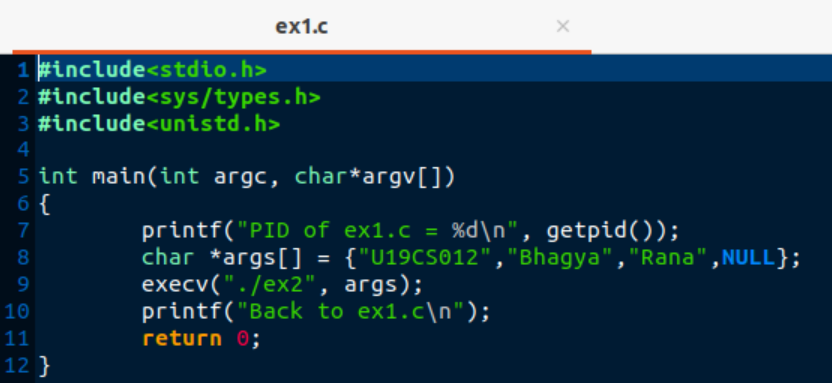


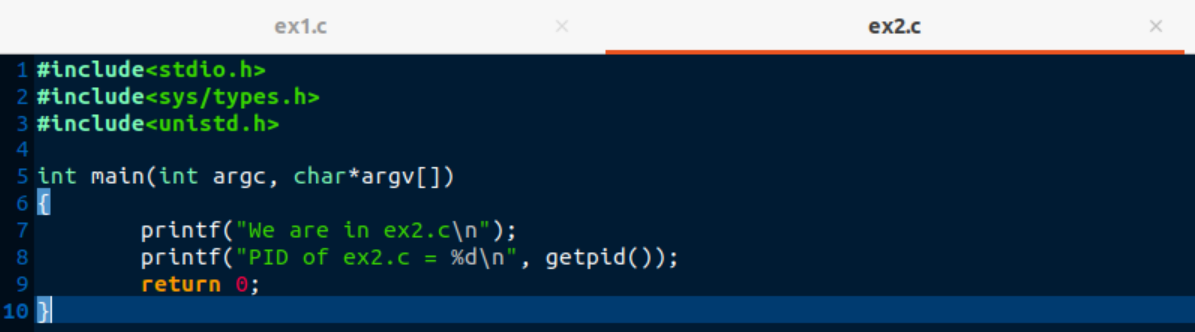
**B.) exec()** [Process Control System Call]

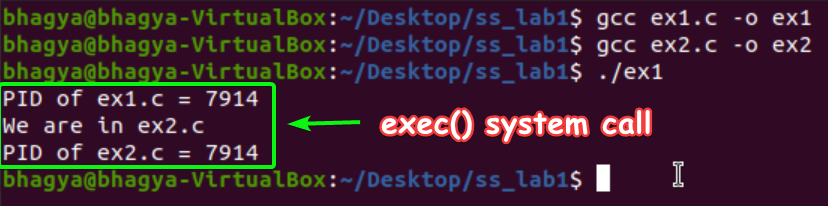
**Purpose**: To execute a file which is residing in an active process (user wants to launch a new file or program in the same process.)



**Example**:



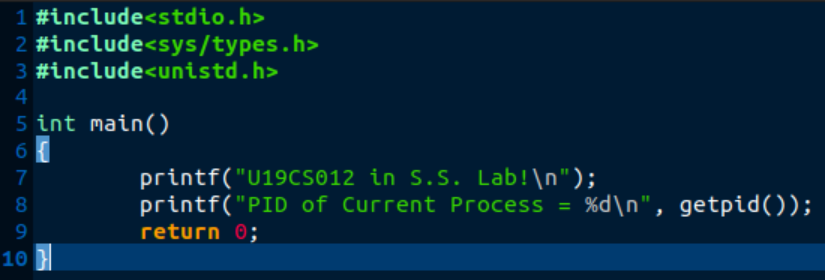


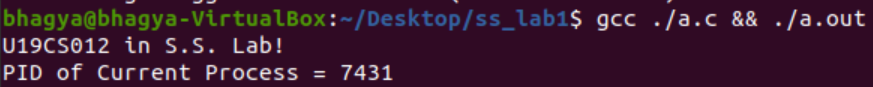


C.) **getpid()** [Process Control System Call]

**Purpose**: to get Process ID of Current Process

**Example**:





D.) **exit()** [Process Control System Call]

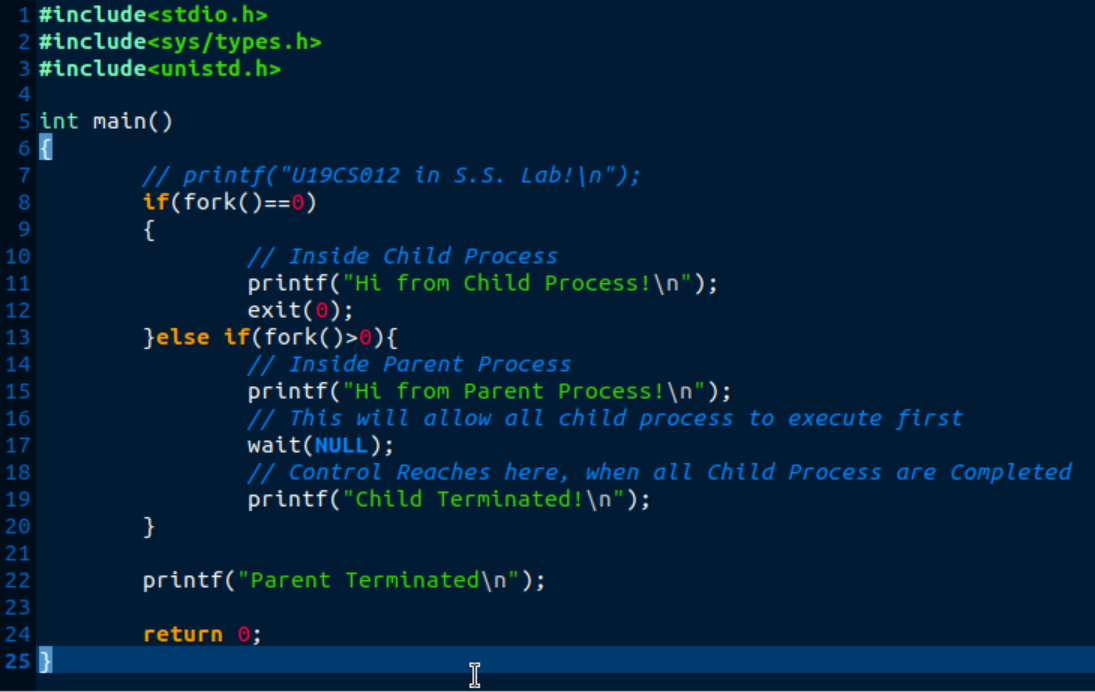
**Purpose**:

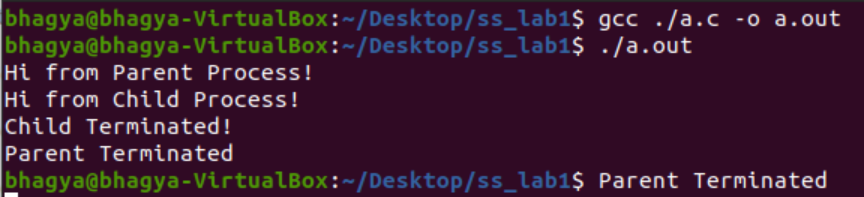
* A process terminates when it **finishes executing its final statement** and asks the operating system to **delete it** using the exit() system call.
* At that point, the process may return a **status value** (typically an integer) to its parent process (via the **wait() system call**).
* Only **Parent process** can kill its Child Process.

E.) wait()

**Purpose**: Helps the Parent Process to wait, until all its child processes are complete.

**Example**:

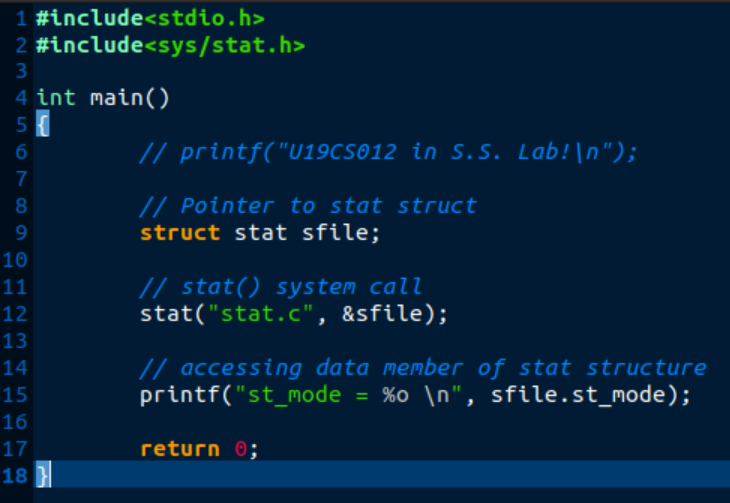


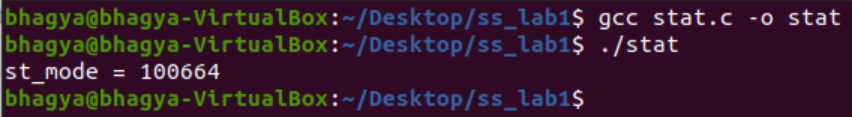


F.) stat()

**Purpose**: To check the **status of a file** such as to check when the file was accessed

**Example**:





G.) opendir()

**Purpose**:

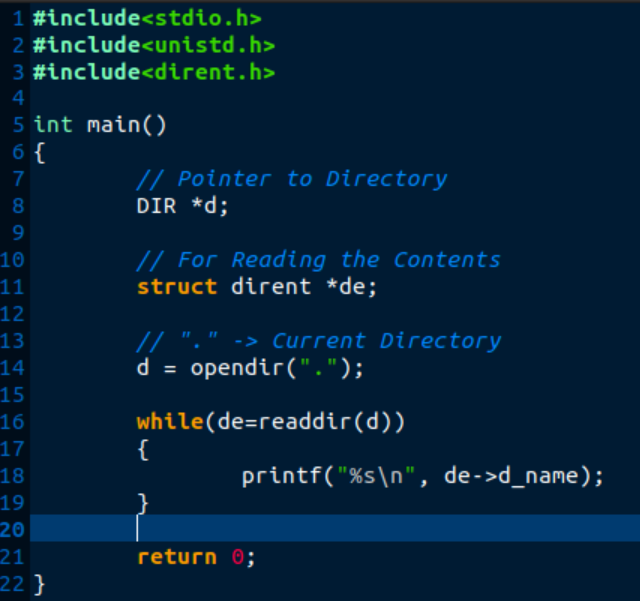
* The opendir() function shall open a directory stream corresponding to the directory named by the dirname argument.
* The opendir() function **opens a directory** and returns a pointer to the directory stream
* The stream is positioned at the **first entry in the directory.**

1. readdir()

**Purpose**:

* readdir - read a directory
* The readdir() function gives **next directory entry** in the directory stream
* It returns NULL on reaching the end of the directory stream

**Example**:

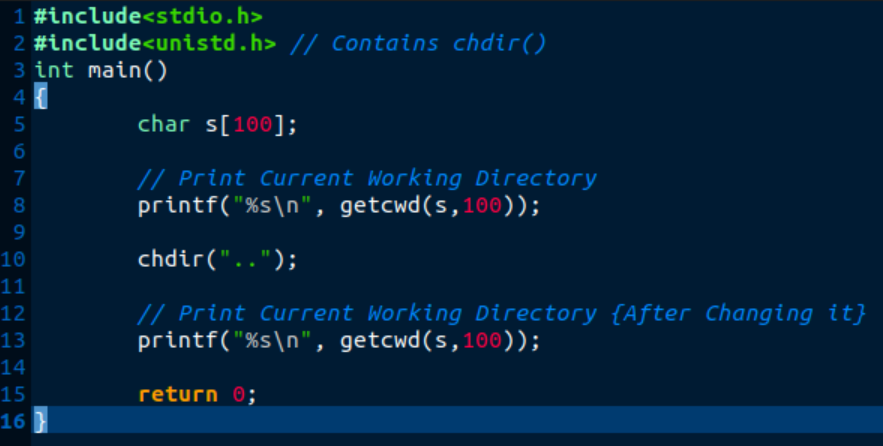


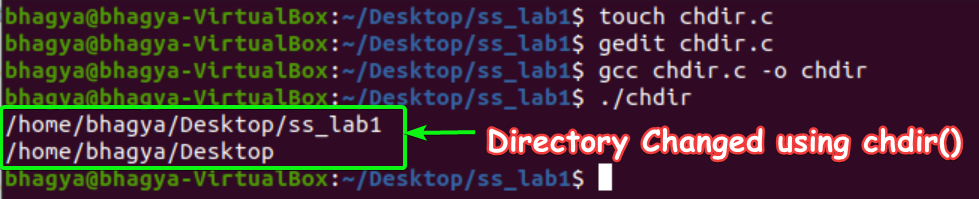


I.) chdir()

**Purpose**: changes the current working directory to that specified in path

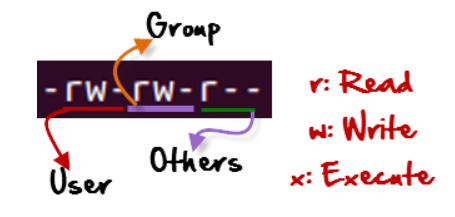
**Example**:

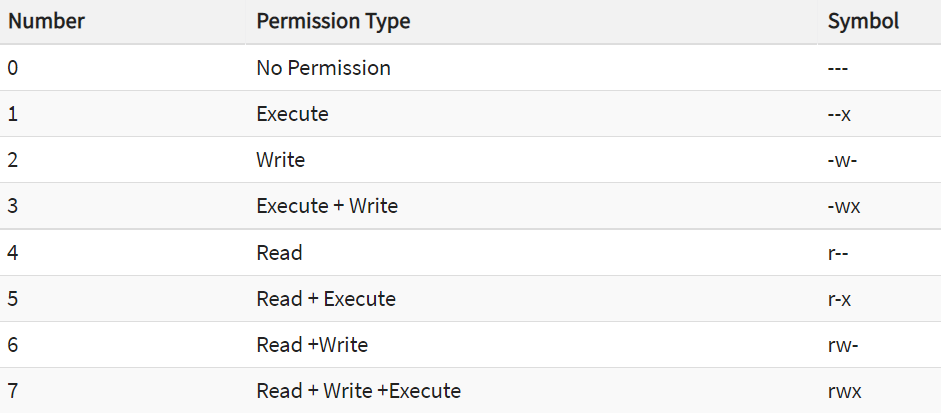




J.) chmod()

**Purpose**:  change permissions of a file



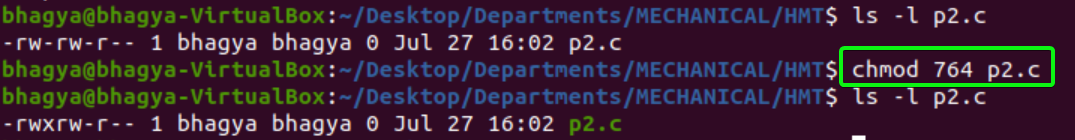


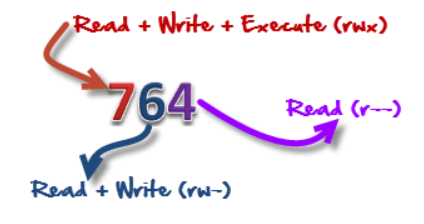
**Example**:

A.) Checking Current File Permissions



B.) chmod 764 and Checking Permissions Again



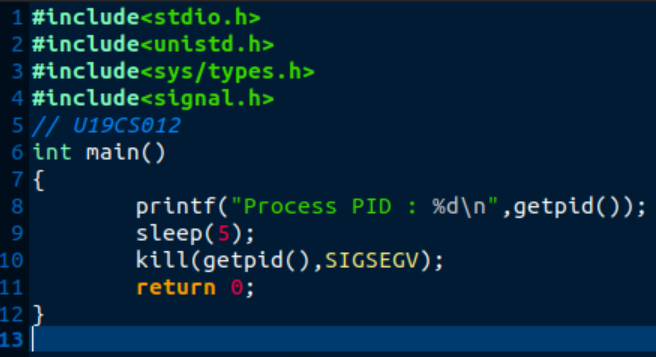


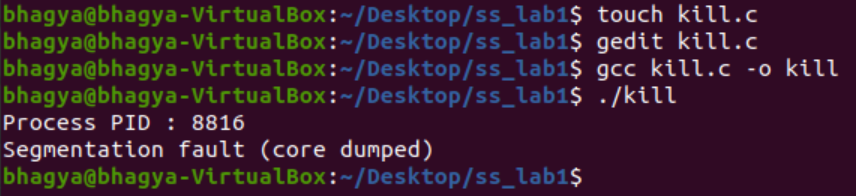
K.) kill()

**Purpose**:

The kill() system call can be used to send any signal to any process group or process.

**Example**:





#include <fcntl.h> for read(), write(), open() & close().

L.) read()

**Purpose**:

int read( int handle, void \*buffer, int nbyte );

* The read() function attempts to read nbytes from the file associated with handle, and places the characters read into buffer.
* The function returns the number of bytes read.
* On end-of-file, 0 is returned, on error it returns -1, setting errno to indicate the type of error that occurred.

M.) write()

**Purpose**:

int write( int handle, void \*buffer, int nbyte );

* The write() function attempts to **write nbytes from buffer to the file** associated with handle.
* The function returns the **number of bytes written to the file**.
* A return value of -1 indicates an error, with errno set appropriately.

N.) open()

**Purpose**:

open( char \*filename, int access, int permission );

* The open() function returns an integer value, which is used to refer to the file.
* If unsuccessful, it returns -1, and sets the global variable errno to indicate the error type.

O.)close()

**Purpose**:

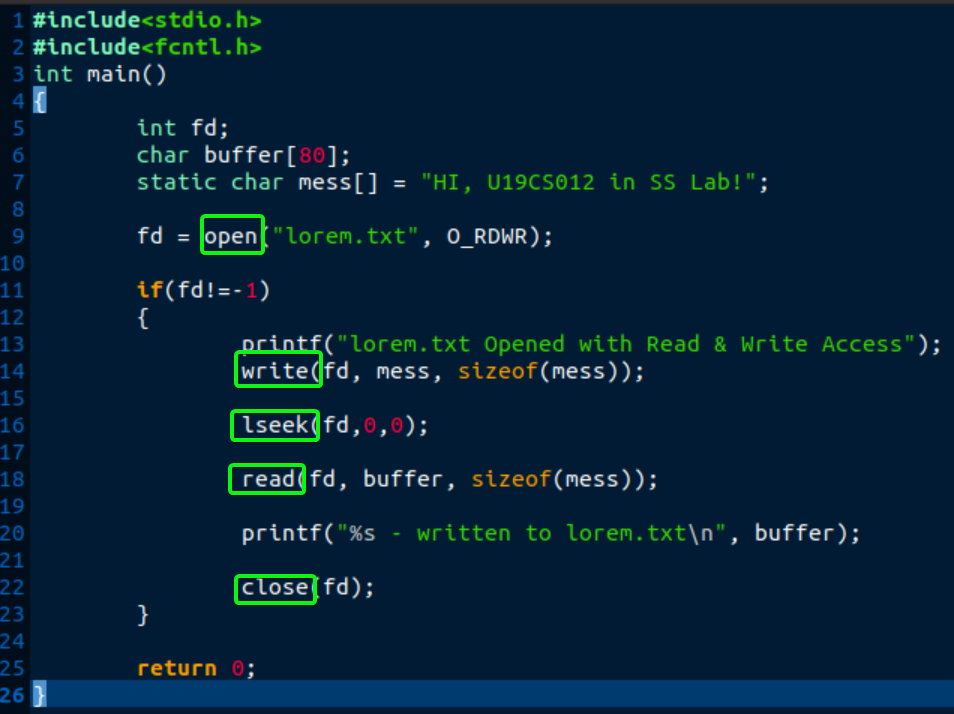
int close( int handle );

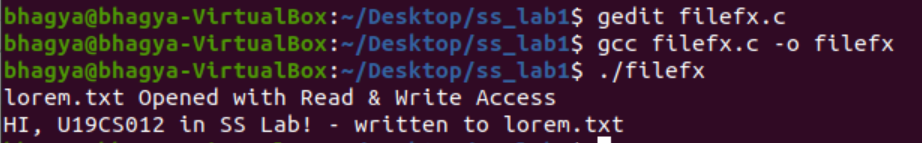
* The close() function closes the file associated with handle.
* The function returns 0 if successful, -1 to indicate an error, with errno set appropriately.

P.) lseek()

**Purpose**: used to change the location of the read/write pointer of a file descriptor.

**Example** for open(), read(), write(), lseek(), close() :



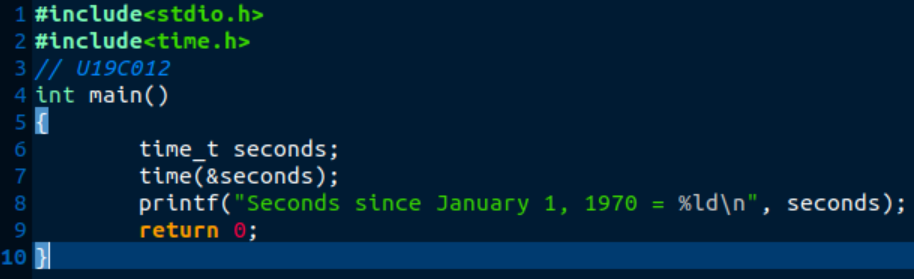


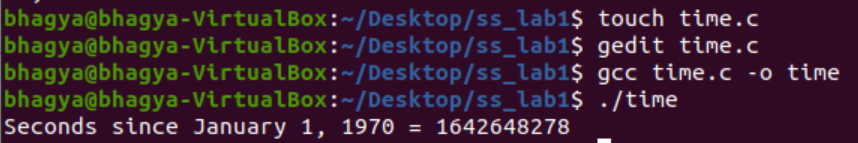
Q.) time()

**Purpose**: the time as the number of seconds since the Epoch

* The time() function is defined in time.h (ctime in C++) header file.
* This function returns the time since 00:00:00 UTC, January 1, 1970 (Unix timestamp) in seconds.
* If second is not a null pointer, the returned value is also stored in the object pointed to by second.

**Example**:

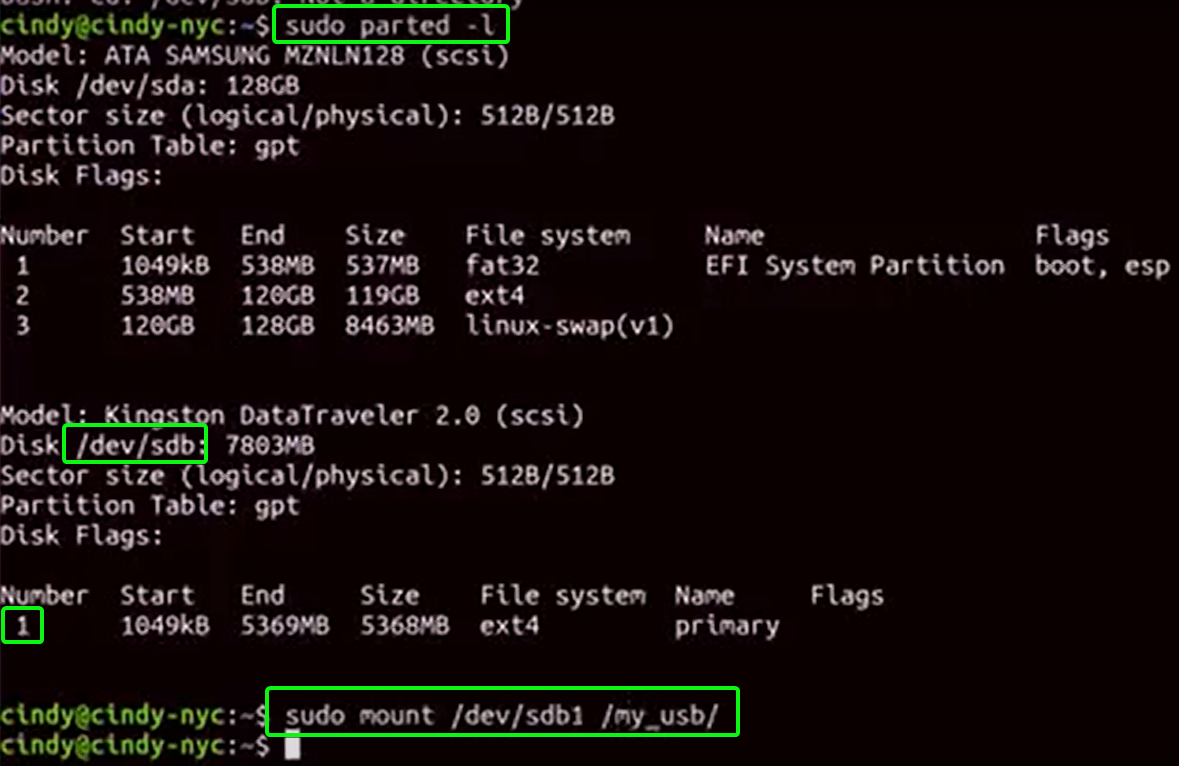




R.) mount()

**Purpose**: mount a filesystem

**Example**:

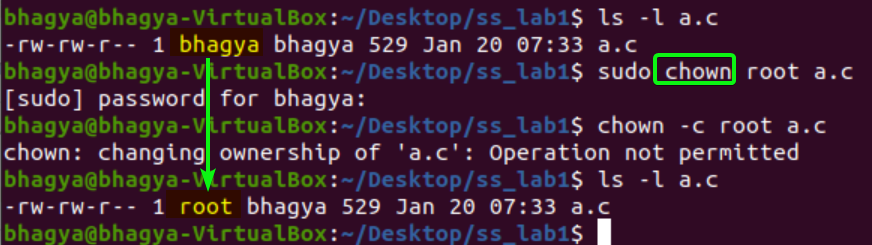


S.) chown()

**Purpose**:

chown command is used to change the file Owner or group. Whenever you want to change ownership you can use chown command.

**Example**:



**SUBMITTED BY**: U19CS012

BHAGYA VINOD RANA