

**OPERATING SYSTEMS (IE2050)**

**2ND YEAR - 2ND SEMESTER**

**ASSINGMENT – 1**

**2020**

|  |  |
| --- | --- |
| **Name** | K.K.P De Alwis |
| **IT Number** | IT18106848 |

Submitted to

Sri Lanka Institute of Information Technology

Date-2020.05.06

**Declaration**

I certify that this report does not incorporate without acknowledgement, any assignment submitted for the degree or diploma in any university, and to the best of my knowledge this document was created by myself for the purpose of Operating System Module assignment. The referred documents are mentioned under the Reference.

**Content**

Task 1 - Calculating Average of a Dataset Obtain from a File Using Threads.

1.1 Introduction About C POSIX Threads……………………………………………………1

1.2 Program execution procedure……………………………………………………………..2

1.3 Execution Sequence Explanation of the Program…………………………………………2

Task 2 - Program That Lecturer can upload, download, edit and read case studies and Student can download and read case studies

2.1 Introduction to System V Shared Memory...……………………………………………...4

2.2 Program Execution Procedure…………………………………………………………….4

2.3Execution Sequence Explanation of the Program………………………………………….6

3 References…………………………………………………………………………………...7

**Task 01 – Calculating Average of a Dataset Obtain from a File Using Threads.**

**1.1 Introduction About C POSIX Threads**

To implement parallelism in shared memory multiprocessor architectures, Threads can be used. For UNIX system a standardized C language thread programming interface has been specified by IEEE POSIX 1003.c standard. POSIX Threads or Pthread implements using that standardized interface specified by IEEE.

A thread consists of a thread ID, a program counter, a register set and a stack. Thread shares its code section, data section, and other operation system resources with other threads which belongs to the same process. The processes that contain more than one thread can perform more than one task at a time.

The subroutines comprise the Pthread can be informally grouped into four major groups;

**Thread Management;**

Routines that work directly on threads for creating, detaching, joining and etc.

In this task multiple threads are going to be created to calculate average of a datasets line by line implementing Pthreads. The also include functions to thread attributes.

**Mutexes;**

Routines that used to deal with synchronization among threads. It is an abbreviation for “mutual exclusion”.

Mutex functions provide for creating, destroying, locking and unlocking mutexes.

**Condition Variables;**

Routines that address communications between threads that shares a mutex. Based upon programmer specified conditions. This group includes functions to create, destroy, wait and signal based upon specified variable values. Functions to set condition variable attributes are also included.

**Synchronization;**

Routines that manage read/write locks and barriers.

In this program Thread Management, Mutexes and Synchronization are going to be used.

“pthread.h” header file should include in each source file that uses the Pthreads library. All identifiers in the Pthreads library begins with **pthread\_.** In this program following identifiers in Pthread library are going to be used;

* pthread\_t – Used to define a thread identifier attribute
* pthread\_create (thread\_id, attr, start\_routine, arg) – Create a new thread and make it executable.
* phtread\_join (thread\_id, status) – Subroutine that blocks the calling thread until the specified thread\_id thread terminates.
* pthread\_exit (status) – Subroutine used to terminate threads.
* pthread\_mutex\_t – Used to define a mutex variable
* pthread\_mutex\_lock (mutex) – Routine used by a thread to acquire a lock on a specified mutex variable. If the mutex is locked by another thread, this call will block the calling thread until the mutex is unlocked.
* pthread\_mutex\_unlock (mutex) – Unlock a mutex if called by the owning thread of a mutex
* pthread\_mutex\_destroy (mutex) – Used to free a mutex object which no longer needed.

**1.2 Program Execution Procedure**

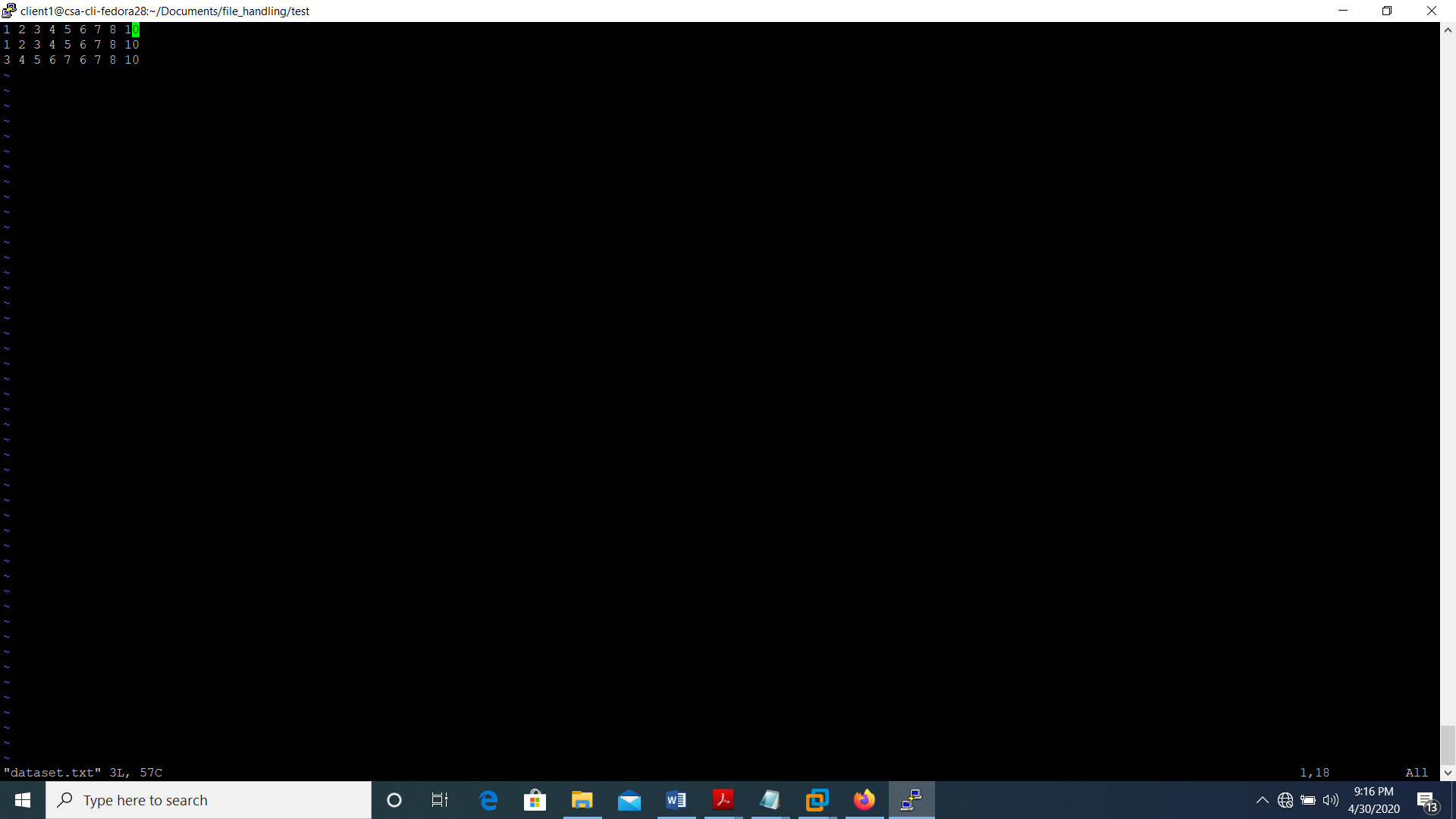
Create a file with integer data datasets in the same directory that program is contained. File should contain integers line by line and it should be name as “dataset.txt”. The “dataset.txt file should not contain empty lines without integer values if it contain a empty line program will give an error message.

Figure 1.1 – format of data for dataset.txt file

Then compile the program and see weather the “average.txt” file is created under the same directory which program is contain and the average and sum of each data set will be stored on that file.

**1.3 Execution Sequence Explanation of the Program**

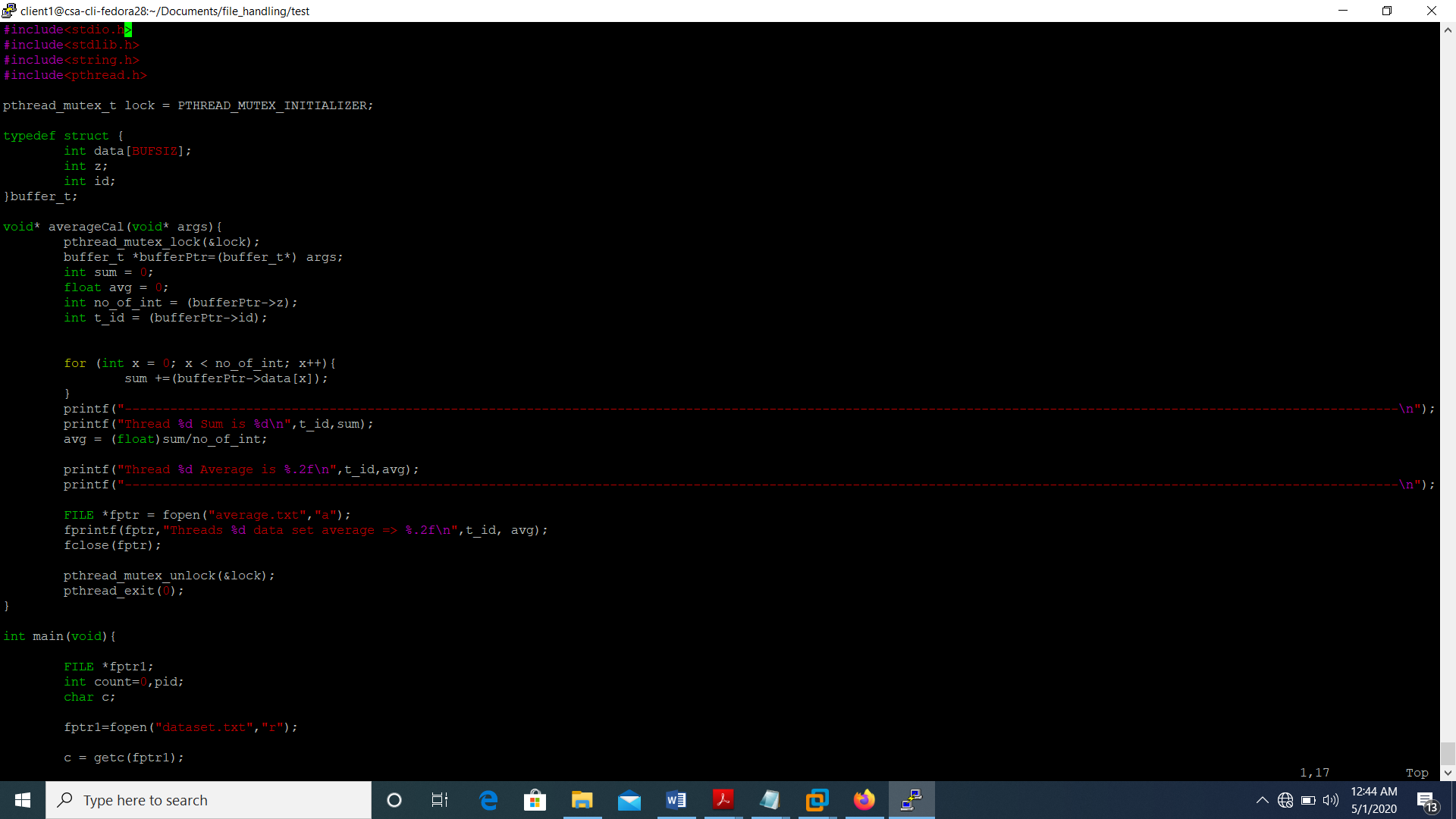
In the developed program there are two major functions main and averageCal. In the main function first “dataset.txt file” is opened and count the number of integer dataset lines in that file and close the file.

Again, the file is opened using a FILE pointer and create a structure array and pthread\_t structure data type array according to the number of lines read by the program earlier.

Then again, the file read line by line and insert the read integers on one line, one by one to the integer data array that contain inside the data structure array that have been created.

After that we pass that data structure array element one by one to pthread\_create function with the corresponding pthread\_t structure type data and averageCal function which is the start routine that a thread will execute after a thread is created.

In averageCal function first we going to lock the mutex variable which is declared in statically as a global variable, using pthread\_mutex\_lock function to avoid simultaneous access of multiple threads to the critical section of the averageCal function that will stimulate the race condition in the program. Then the passed by reference structure array element that loaded into the argument of the averageCal function is assign to a pointer of buff\_t structure data type.

 Figure 1.2 – assign args value to pointer of buffer\_t structure data type

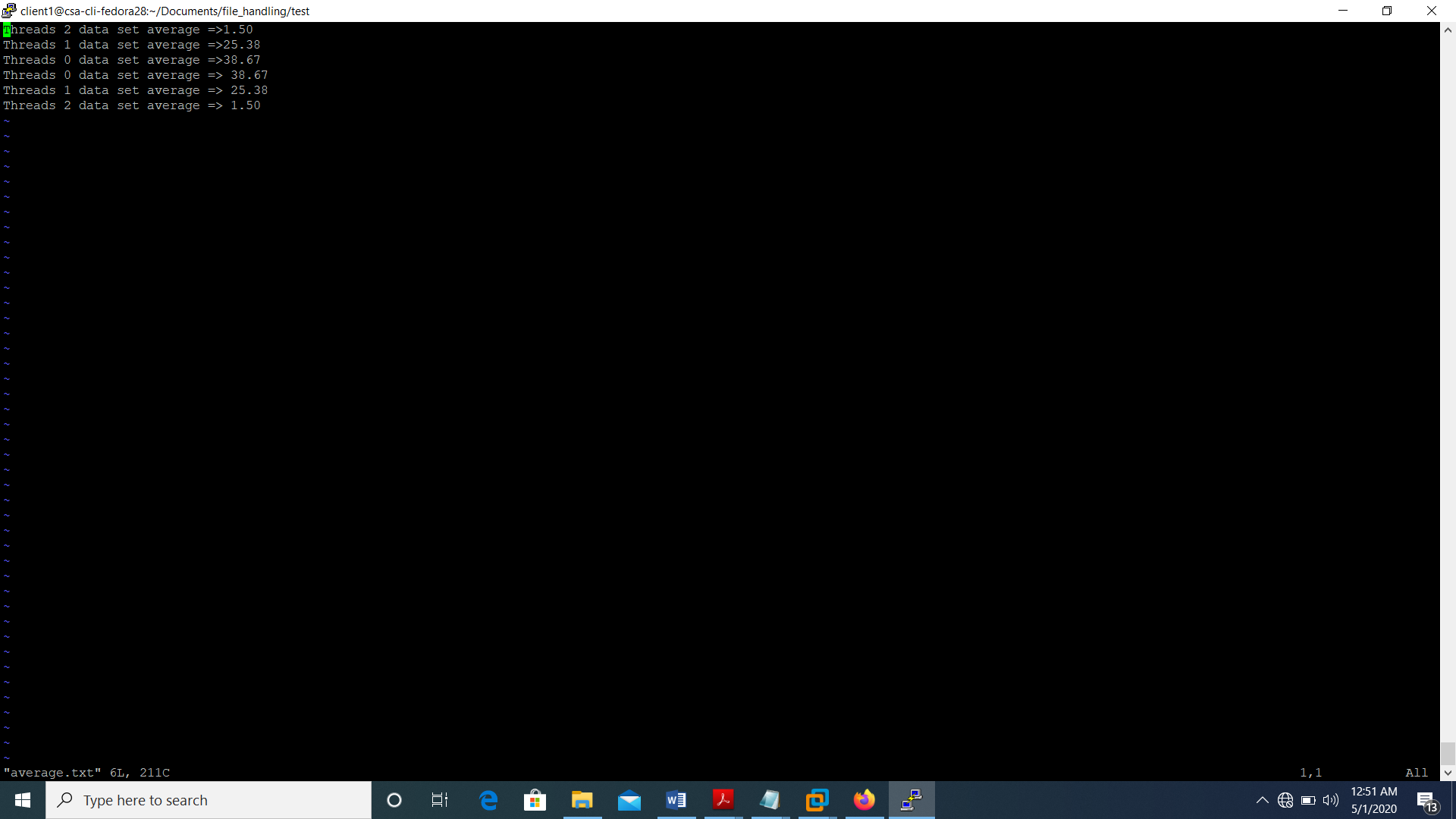
Then the using integer array that contain inside that structure average of each line of data set in dataset.txt file is calculated. At the end of that function a file called average.txt is opened and the calculated average value of each line of the dataset.txt file is printed on that file.

Figure 1.3 – printed average values of each line of dataset.txt file into average.txt file.

**Task 2 – Program That Lecturer can upload, download, edit and read case studies and Student can download and read case studies**

**2.1 Introduction to System V Shared Memory**

Shared memory is one of an Inter Process Communication (IPC) mechanism available in Linux and Unix systems. A shared memory segment is created by the kernel and mapped to the data segment of the address space of a requesting process. Shared memory comes in two flavors the traditional System V shared memory and the newer POSIX shared memory. For this assignment task traditional System V shared memory has been used.

“sys/ipc.h”, “sys/shm.h” header files should be included to used system V shared memory in the program.

To use a System V IPC mechanism System V IPC key is required. To generate IPC key ftok function is used.

In this program following System V shared memory calls have been used;

* shmget (key\_t key, size\_t size, int shmflg) – Used to get a shared memory segment associated with the given key.
* shmat (int shmid, const void \*shmaddr, int shmflg) – Used to attach a shared memory segment identified by shmid.
* shmctl (int shmid, int cmd, struct shmid\_ds \*buf) – Used to control operations of a System V shared memory identified by shmid.

**2.2 Program Execution Procedure**

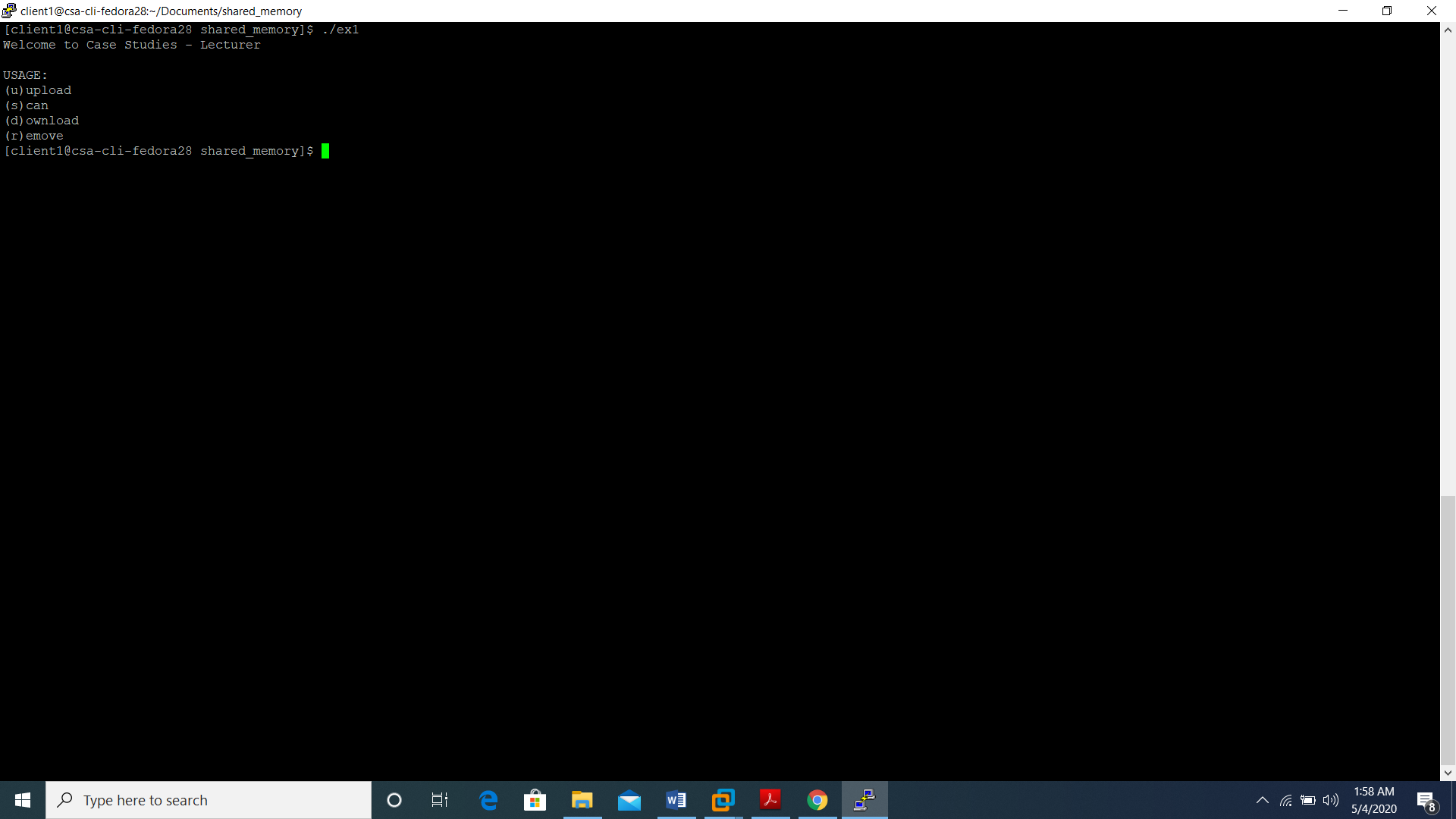
After compile the program, run the program using executable file and it will display command line arguments that should enter with the executable file name to perform each function in the program.

Figure 2.1 – Output of that give with only Executable File name

**Upload a Content to The Shared Memory**

To upload a content to the created shared memory first a file with the case study should be created in the same folder that program is contained. After that upload should provide as second command line argument with executable file name. Then the case study file name should be provided as an input while executing the program.

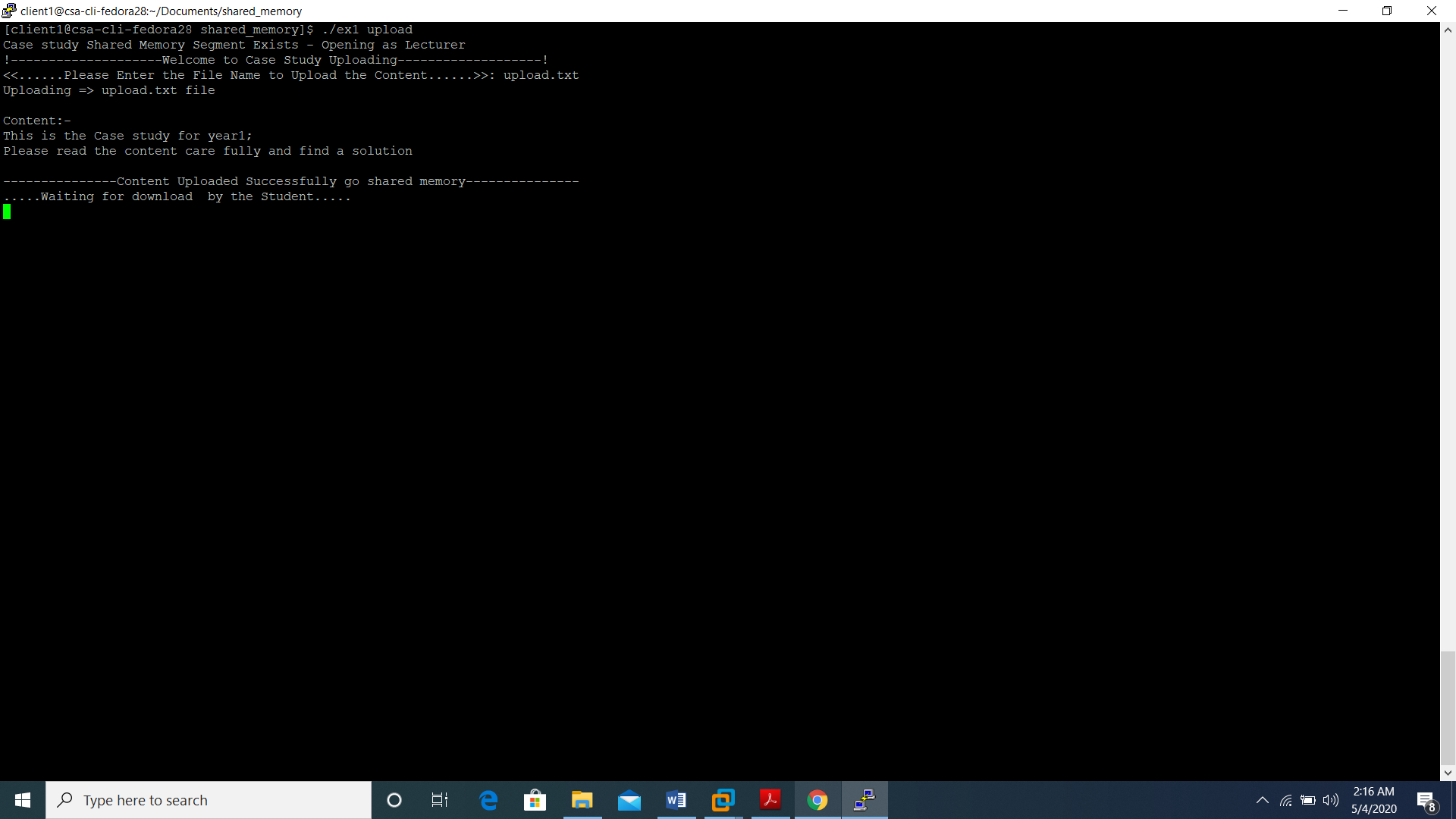


Figure 2.2 – Uploading a file to the shared memory

**Read the Content in The Shared Memory**

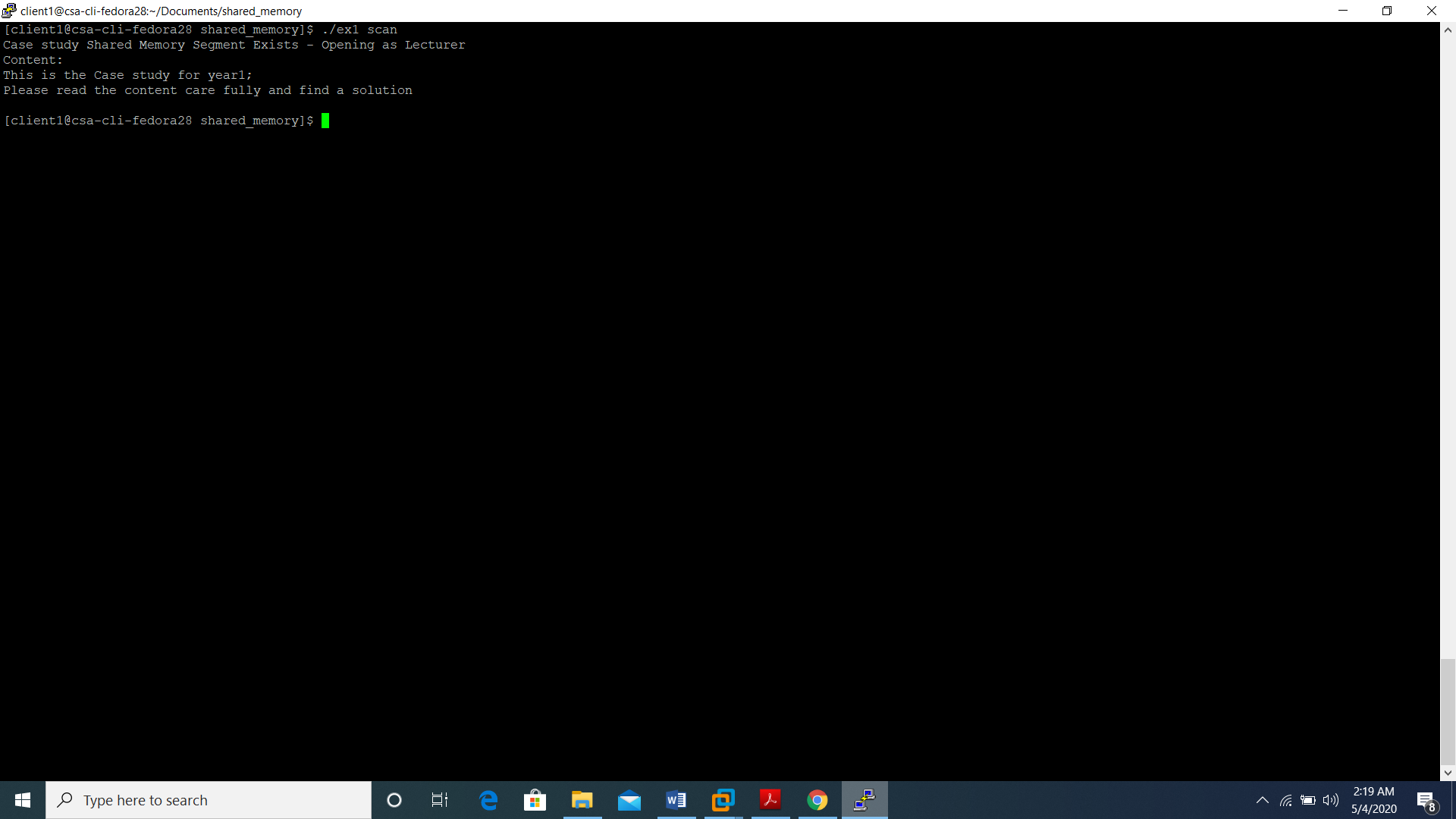
Then to read the content in the shared memory, scan should provide as a command line argument with the executable file name. Uploading should be done before read the content in the shared memory otherwise program will display a waiting massage.

Figure 2.3 – Read the content in the Shared memory

**Download the Content in The Shared Memory**

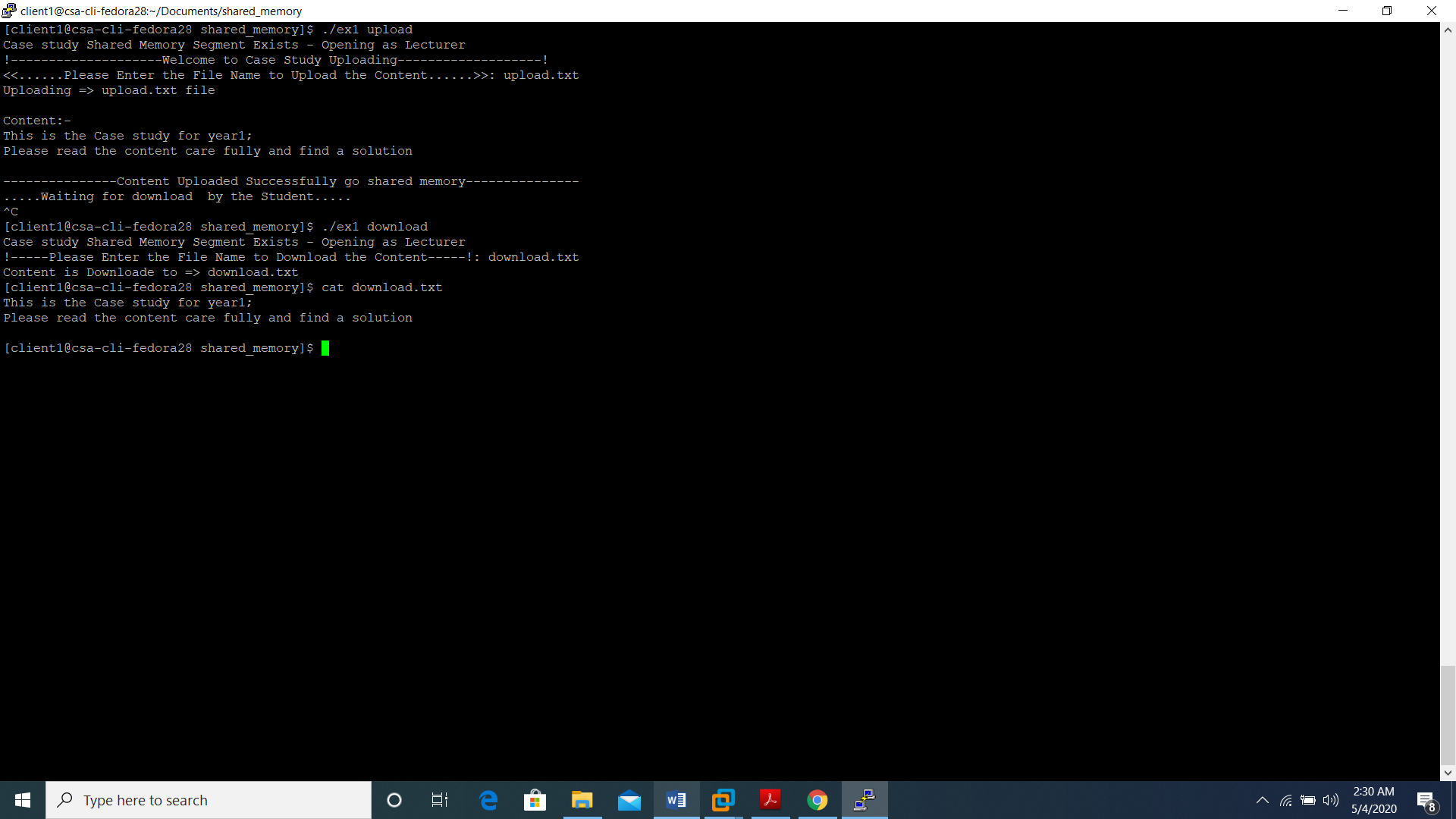
To download the content in the shared memory download should provide as a command line argument with the executable file name. Then the file name that should download the content in the shared memory should provide as an input while executing the program. If a content is doesn’t upload to shared memory program will display a waiting massage.

Figure 2.4 – Download the content in the shared memory to file

**Delete the Content in The Shared Memory**

To delete the created shared memory, delete should provide as a command line argument with the executable file name. Before delete the content in the shared memory student must download the content in the shared memory.

**Important**

Lecturer have the privileges to upload, download, read, and delete the content in the shared memory. Student have the privileges to download and read the content in the shared memory. But until the Student download the content in the shared memory Lecturer have to wait update or remove the content in the shared memory. Also readshm(), downloadshm() functions have to wait until a content is uploaded to shared memory.

**2.3 Execution Sequence Explanation of the Program**

To create a shared memory segment that associate with the generated key from ftok() shmget system call is used.

After creating the shared memory using shmat system call the created shared memory is attached to data section of the program by char pointer type variable.

After attaching using the pointer the content in the shared memory can be manipulated through that pointer.

* To upload a content to the shared memory uploadshm() function is used.
* To read the content in a shared memory readshm() function is used.
* To download the content in a shared memory to a file downloadshm(() function is used.
* To remove shared memory with it content removeshm() function is used.

At the beginning of the program if argc value is one that means the argument is only the program name that in executable mode will trigger the description() function that explain how to execute other functions available in the program.

In the switch case of the program second command line argument first character value is compared with each case in it. Then according to that upload, scan, download and remove functions are triggered.

If upload is triggered it provide an input field to enter the file name that contain the case study and after entering the file name function will read the file and upload the content into the shared memory. When scan is triggered the content in shared memory will get displayed in the screen but to display the content in the shared memory first the upload function must run. Likewise, when download triggered the content in the shared memory will get downloaded to the file and remove will delete the content in the shared memory. For download the content in the shared memory first a case study should upload to the shared memory otherwise function will wait until a content get uploaded to the shared memory. For remove the content in the shared memory student should download the content in the shared memory otherwise remove function will remain until student download the content in the shared memory.

**3** **References**

[1]"POSIX Threads Programming", *Computing.llnl.gov*, 2020. [Online]. Available: https://computing.llnl.gov/tutorials/pthreads/#Pthread. [Accessed: 02- May- 2020].

[2]"Linux System V and POSIX IPC Examples", *Hildstrom.com*, 2020. [Online]. Available: http://hildstrom.com/projects/ipc\_sysv\_posix/index.html. [Accessed: 02- May- 2020].

[3]*Operating Systems*- *Lab Exercise 3*., 1st ed. SLIIT, 2018, p. 8.

[7]*Operating Systems-Lab Exercise 6*, 1st ed. SLIIT, 2018, p. 6.