Assignment 2 report

Real-time Operating System - 48450

Group No. : Group 9

Student Name: Jeong Bin Lee & Quang Le

Student ID: 12935084 & 11993654

Note: This is a guide template to help you complete the assignment. There will be no penalty on your assignment score if you do not use this template. You are welcome to design your own template for the assignment.

Table of Contents

I. Introduction

The purpose of this lab is to implement the concepts of pipes, threads, and semaphores to create a program which utilises 'real-time' reading and writing through the pipe-line concept from an input file "data.txt" to an output file "output.txt". To build the program, type "*make"* which will run "makefile" by first removing existing builds and then compiling the program again using "build.sh".

II. Theory of operation

Pipes

Pipes is a one-way communication IPC to pass parameters from one program process to another. Pipes is created from the parent process, before 'forking' to the child process.

Threads

Threads are the single sequential flow of control within a process, however multiple threads can be running in an instance. The use of multi-threading allows the benefits of operating multiple processes at one time, offering better responsiveness, shared resources and scalability.

Semaphores

Semaphores are used as a signalling concept, to notify that a thread is awaiting resources from another thread in order to continue its process. In short, it may be used in a similar sense to a traffic light. Semaphores are used to control the flow of this assignment.

Mutex

Mutex are used as a locking-mechanism to lock/unlock shared resources for the calling thread. In other terms, it locks the resource to be mutually exclusive to the producing thread and sends other processes into a queue until it is unlocked by the thread. However, mutex was not used at all for this assignment.

III. Operating condition

In this section, a summary about your understanding of the assignment is given. In order to complete the assignment, you might give a summary about key points that the assignment contains. In addition, you might explain the relationships between these points.

Thread A – The operation of thread A is to read one line of text from a given input file and pass the data to thread B using a pipe. Then signal semaphore B.

Thread B – The operation of thread B is to receive the data from thread A and relay it to thread C. This can be done using shared memory or message passing. Then signal semaphore C.

Thread C – The operation of thread C is to receive the data from thread B and check if the end of header has been reached. If the end of header is detected, the next set of data received will be written to a file called "output.txt" then signal semaphore A.

IV. Implementation

1. Method

In this section, you might list how you solve the problem and how you complete your program. You might write your software design strategy about your programming.

The approach to the assignment is as per below, initially starting with Thread A > Thread B > Thread C:

Thread A

1. Begin by setting the variables or parameters to read the data from the data file.
2. Read each line from the data file and print to confirm that the correct line is being read.
3. Write to the pipe.

Thread B

1. Read the contents from the pipe.
2. Test the contents by printing the received data and checking if it's the correct data.
3. Parse the data to Thread C using shared variable.

Thread C

1. Read the contents from the shared variable.
2. If the end of file has been reached, close semaphores/threads/file.
3. If end of header has been reached, toggle write to file flag to write data to "output.txt" file.

Print the contents of the "output.txt" file to check that the correct data has been passed.

Debug Section

All other errors have been fixed using trial and error to resolve.

Use semaphores where required for each thread to ensure that the data has been correctly passed between the threads.

2. Flow chart and/or Gantt Chart

You might draft a flow chart and/or a Gantt chart about this assignment.

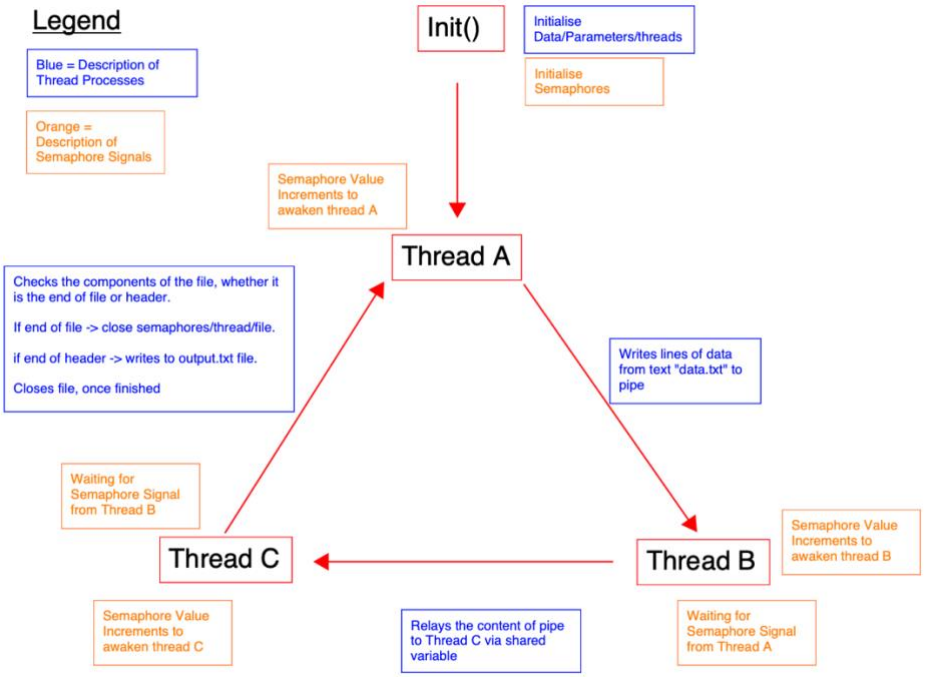

Figure 4.2a) Flow diagram of program

V. Experiments

1. Hypothesis

If the semaphore is not applied, the threads go from Thread A > Thread B > Thread C. However, the threads will not be operating in synchronous order, thus causing a segmentation fault, leading to the threads not functioning as intended.

When semaphores are used, threads go through a sequential process, thus successfully outputting the content region as per flow diagram in figure 4.2A. Semaphores are used as a signalling mechanism, where threads are either in a running or waiting state. Once the semaphore is signalled, the next thread has one line of data to process, i.e., once thread A is signalled, thread B has one line of data to relay to thread C.

2. Results

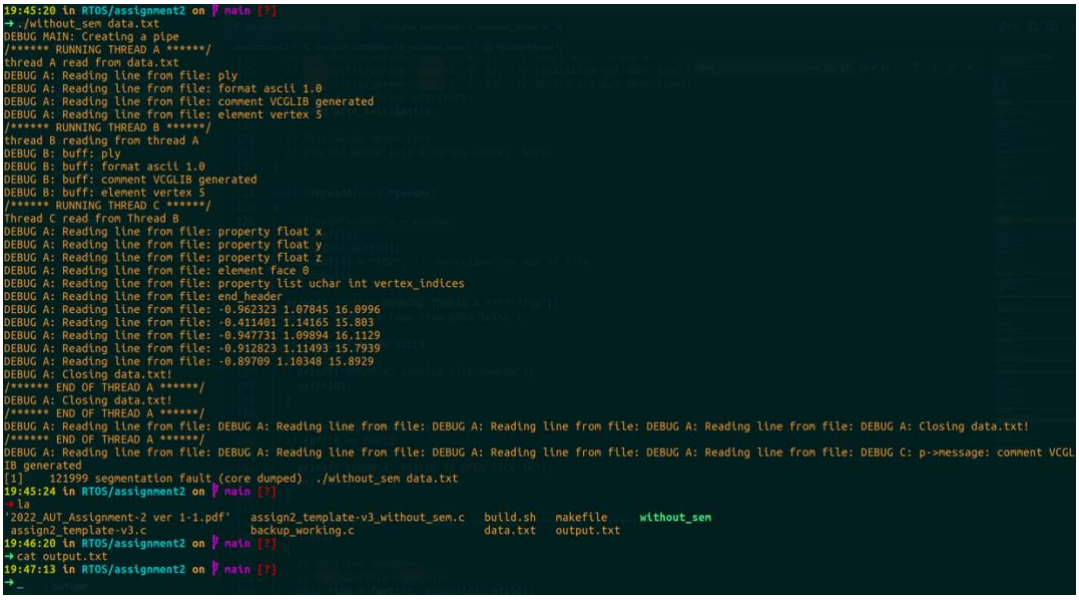

Figure V.2a) Experimental results without semaphores

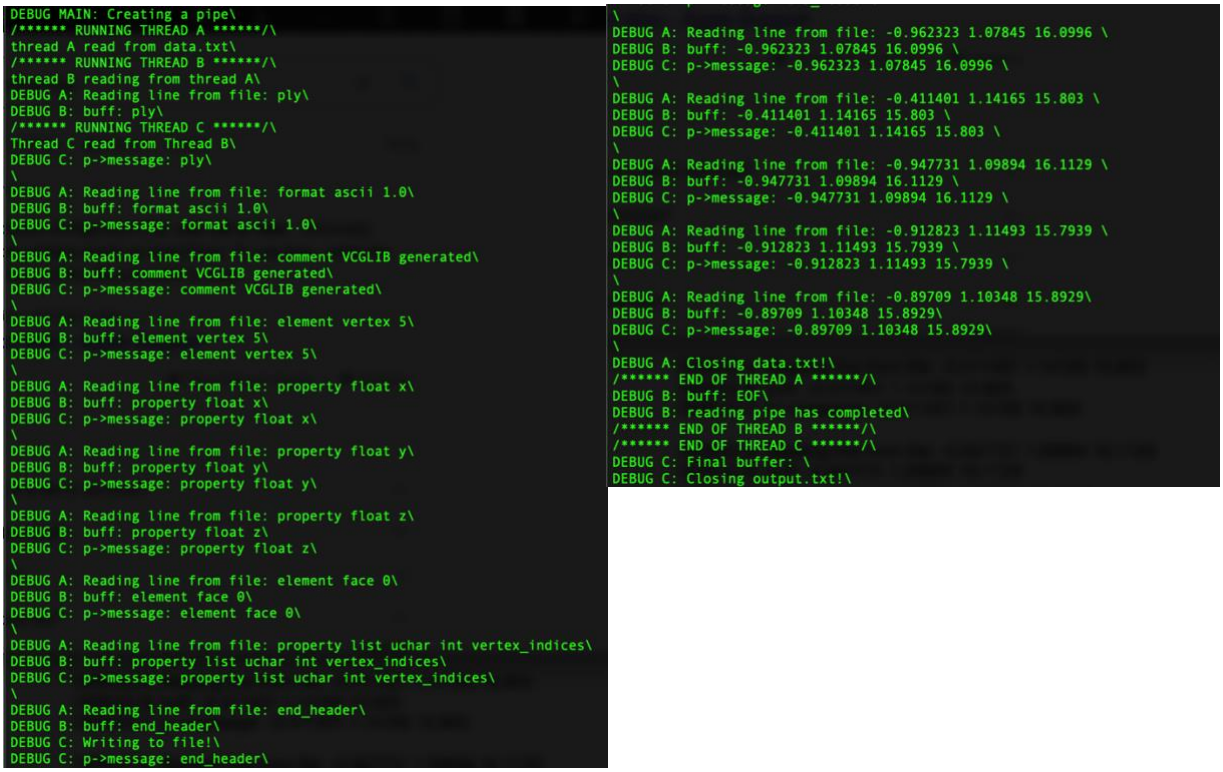

Figure V.2b) Debug results with semaphore.

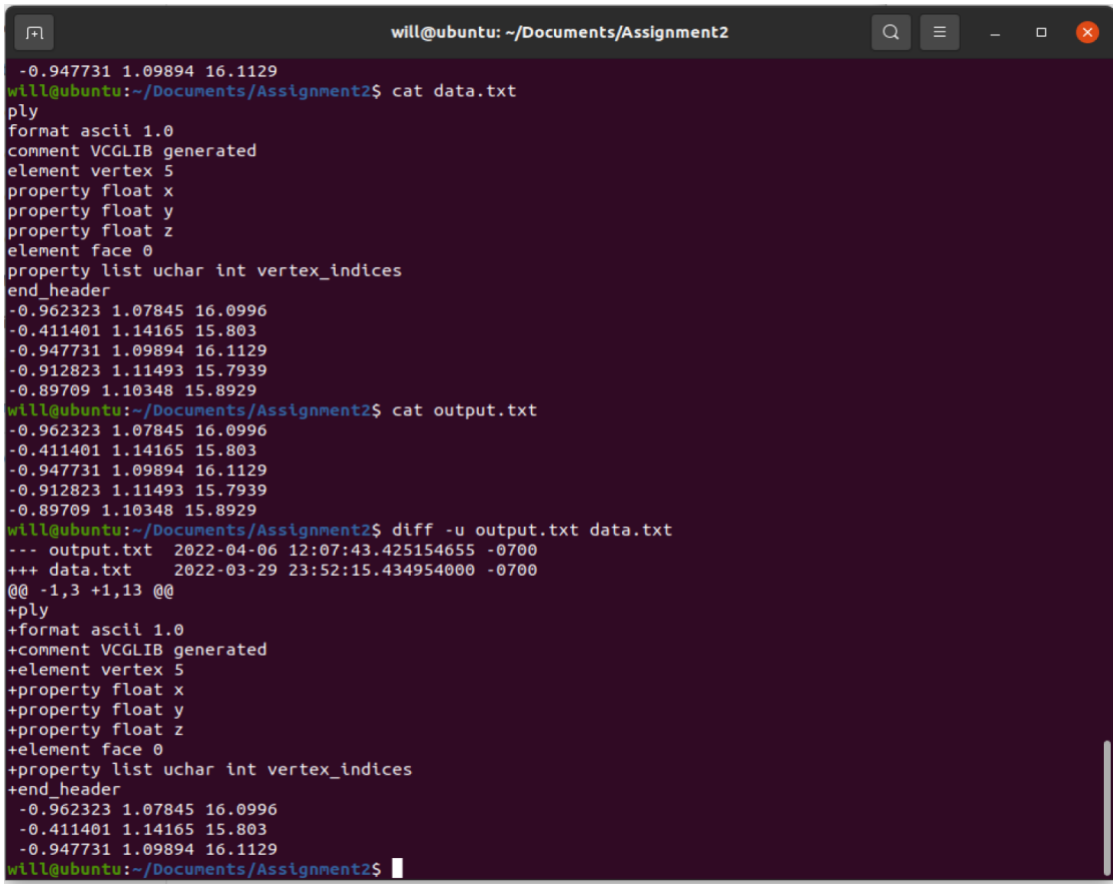

Figure V.2c) Experimental results with semaphore.

VI. Conclusion on result analysis

Without Semaphores

Based on the results shown above, when the program runs without the aid or use of semaphores, it does not successfully write to the output.txt file. As when we run the “cat output.txt” command, the text file reveals no data as shown in figure V.2a. This is due to the logic behind the structure of the program, as it does not sequentially flow between threads, rather is executing without awaiting the data to be properly written or stored.

With Semaphores

When the program includes the use of semaphores between the threads, there is a sequential flow of data through the pipe between the threads to output the contents correctly into the output.txt file as shown in figure V.2c). This is via the use of semaphores signals between the threads to buffer the pipe for thread C to output the data lines onto the “output.txt” file.

Based on the experiment, the use of semaphores is critical in a multi-threaded environment to ensure the processes are running in synchronous order.

VII. References

A. Silberschatz, P. B. Galvin & G. Gagne, 2012, Operating System Concepts, 9 th edn, John Wiley & Sons, New York.