CSE3033 - Operating Systems Assignment 3 Report

Fall 2020

Ali Reza Ibrahimzada
Student ID: 150119870

Huzeyfe Ayaz
Student ID: 150119700

Sena Dilara Yangoz
Student ID: 150119706

Introduction

The goal of this project is, simulating a multi-threaded library system. We have written our codes in the Linux operating system using the C programming language. Pthread library, which is a built-in library in C, have been used for this program. Threads divided into 2 groups namely publishers and packagers. Also, each publisher has its own type as type 1, type 2, etc. The aim of publishers is creating books for their own type and the goal of packagers is taking published books to their own package. Since it is a multi-threaded program, the main goal is to let the threads work concurrently and preserve consistency.

Frequently-Used Functions

We have written 3 functions which we use frequently. Some global variables are used in these functions to perform necessary tasks.

Mutexes and Responsibilities

We have created 2 mutex in order to supply synchronization between threads. One of them is "current_publisher_mutex" and the other one is called "packager_mutex". Each type of publisher has their own mutex, so that one publisher type does not lock the other publisher type and they can work simultaneously since they do not have a common buffer. Additionally, packagers are using corresponding mutexes while taking a book from a randomly chosen publisher type. Packager mutex is unique for packagers to prevent changing the common variables at the same time.

Semaphores and Responsibilities

Currently, we have 2 counting semaphores namely full_semaphores and empty_semaphores. Full semaphores are supposed to dynamically hold the book sizes inside the buffer, so that threads can track the buffer size and take an action based on that. They are initially 0 and we are incrementing them via sem_post function when a publisher thread creates a book and we are decrementing them using sem_wait function when a packager takes a published book. Besides, empty semaphores start with current buffer size since there is no published book initially. Then, we are decrementing it when a book is published and put inside a buffer. Using empty semaphores we can stop packager buffers, and make them wait for publishers to publish a book if the value of empty semaphore reaches zero.

Publish

Publish is a custom function which is being run by all publisher threads. Upon successful creation of publisher threads, first they will calculate their IDs, types, ... and retrieve their corresponding semaphores and mutexes. Moreover, the first publisher thread of a specific type scheduled by the system is responsible to create a buffer for all threads of that type, and hence subsequent threads of the same type will not initialize any buffer.

The main functionality of publish starts in a while loop which continues iterating until the total number of books published by a specific thread reaches the required number of books. When a publisher thread wins the lock for publishing a book, it first checks if the buffer is full. If so, it will double the buffer size and then continue publishing. In case the buffer is not full, the thread will just continue publishing a book and placing it into the buffer. Necessary operations are being done over semaphores to keep track of the process.

Pack

Pack is a custom function which is being run by all packager threads. Upon successful creation of packager threads, first they will get their IDs, and corresponding semaphores and mutexes. The first packager thread scheduled by the system will be allocating memory to the package buffer. Then, the packager threads will continue working until there are no books left in the publisher buffers and no publisher threads are available from all publisher types.

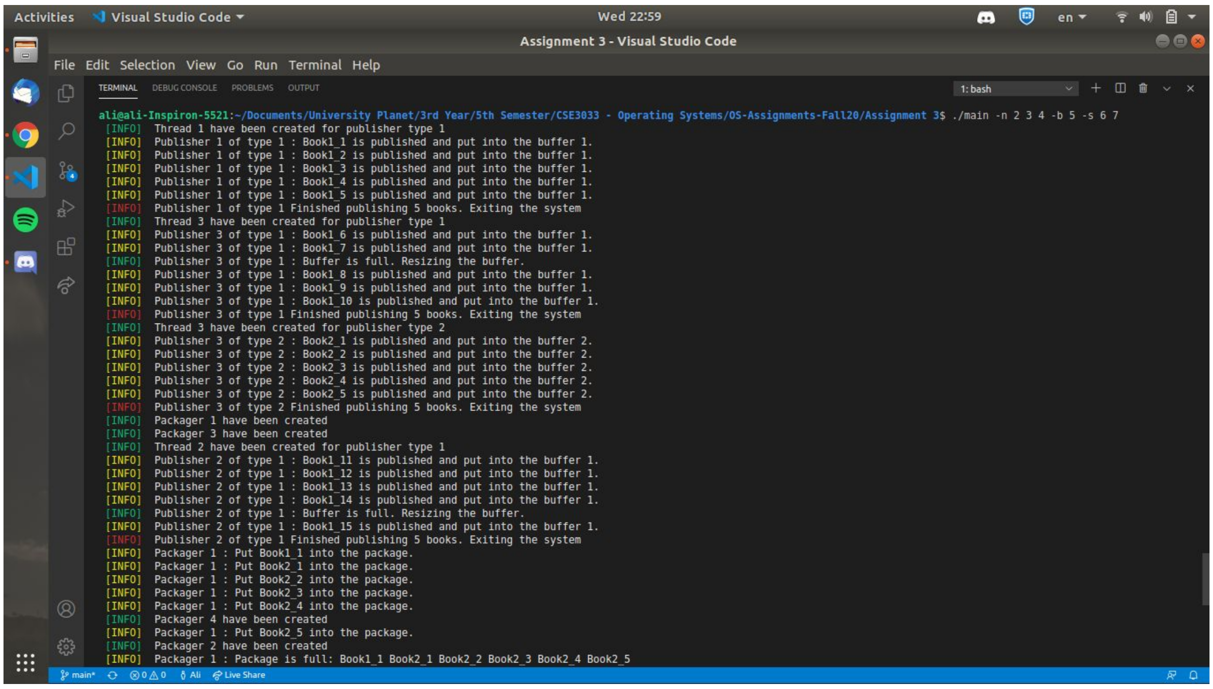
In case there exists at least one book in the buffer of the randomly selected publisher type, we first lock the mutex of that publisher type so that while packing a book, no thread of that type could place a new book in the buffer. Then, we pop a book from the beginning of the buffer, shift other books 1 to the left and pack the popped book. Before placing the packed book into the package, we first check if the package is full or not. If it's full, we print a message that the package is full along with its contents. Then we clear the content of the package buffer and put the next packed book to the beginning of the package buffer. Necessary operations are being applied on semaphores in order to keep the correct state of the program.

Execution Order

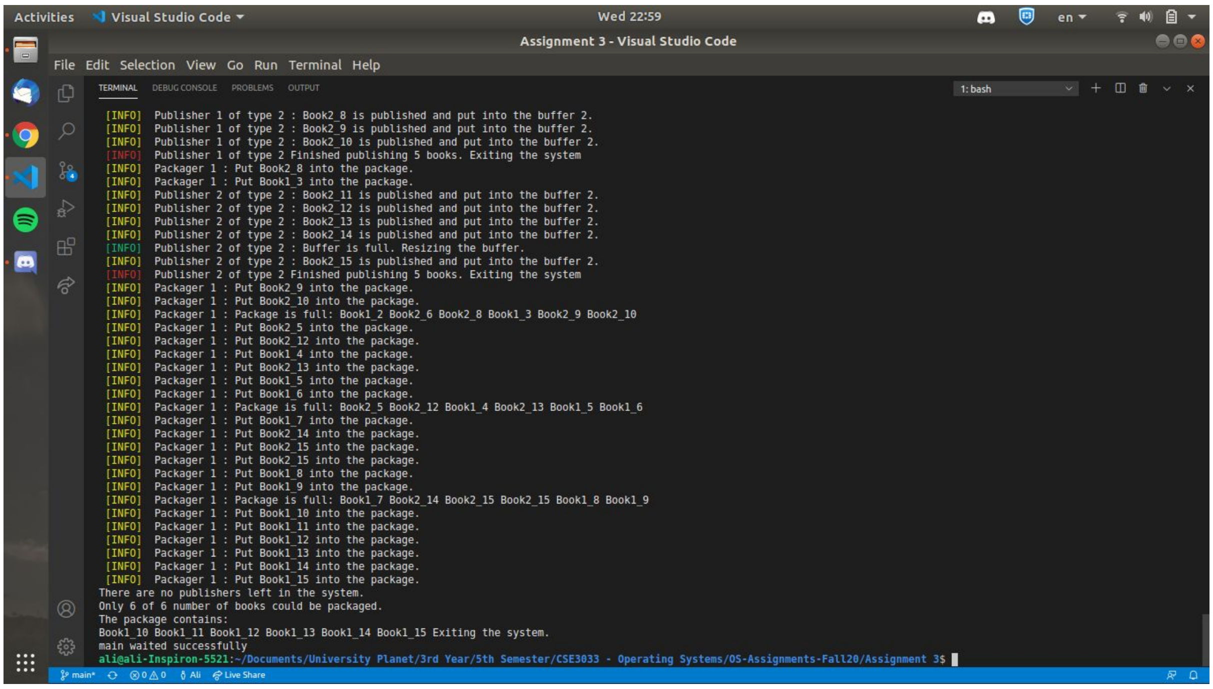
Initially the program starts from the main function. There, we first make a necessary check if the user has provided the correct number of command line arguments. Then using a loop, we parse the command line arguments and store the given values in specific variables. Then based on the given values, we dynamically allocate memory for our data structures. Since dynamic memory allocation can be unsuccessful sometimes, we make sure that we have the necessary memory, otherwise we simply exit from the program. In order to preserve consistency and make sure that threads can work concurrently, we create some semaphores and mutexes, and then initialize them.

Once all those allocations have been done, then it's time to create publisher and packager threads. The main thread will create the required number of publisher and packager threads, and then wait for their executions to finish. Finally, the main thread will deallocate the allocated memories from heap and terminate.

Screenshots


Activities
Visual Studio Code
Wed 22:59
Assignment 3 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT
1: bash
main* ⊗0 ⚠0 Ali Live Share

```
ali@ali-Inspiron-5521:~/Documents/University Planet/3rd Year/5th Semester/CSE3033 - Operating Systems/OS-Assignments-Fall20/Assignment 3$ ./main -n 2 3 4 -b 5 -s 6 7
 [INFO]  Thread 1 have been created for publisher type 1
 [INFO]  Publisher 1 of type 1 : Book1_1 is published and put into the buffer 1.
 [INFO]  Publisher 1 of type 1 : Book1_2 is published and put into the buffer 1.
 [INFO]  Publisher 1 of type 1 : Book1_3 is published and put into the buffer 1.
 [INFO]  Publisher 1 of type 1 : Book1_4 is published and put into the buffer 1.
 [INFO]  Publisher 1 of type 1 : Book1_5 is published and put into the buffer 1.
 [INFO]  Publisher 1 of type 1 Finished publishing 5 books. Exiting the system
 [INFO]  Thread 3 have been created for publisher type 1
 [INFO]  Publisher 3 of type 1 : Book1_6 is published and put into the buffer 1.
 [INFO]  Publisher 3 of type 1 : Book1_7 is published and put into the buffer 1.
 [INFO]  Publisher 3 of type 1 : Buffer is full. Resizing the buffer.
 [INFO]  Publisher 3 of type 1 : Book1_8 is published and put into the buffer 1.
 [INFO]  Publisher 3 of type 1 : Book1_9 is published and put into the buffer 1.
 [INFO]  Publisher 3 of type 1 : Book1_10 is published and put into the buffer 1.
 [INFO]  Publisher 3 of type 1 Finished publishing 5 books. Exiting the system
 [INFO]  Thread 3 have been created for publisher type 2
 [INFO]  Publisher 3 of type 2 : Book2_1 is published and put into the buffer 2.
 [INFO]  Publisher 3 of type 2 : Book2_2 is published and put into the buffer 2.
 [INFO]  Publisher 3 of type 2 : Book2_3 is published and put into the buffer 2.
 [INFO]  Publisher 3 of type 2 : Book2_4 is published and put into the buffer 2.
 [INFO]  Publisher 3 of type 2 : Book2_5 is published and put into the buffer 2.
 [INFO]  Publisher 3 of type 2 Finished publishing 5 books. Exiting the system
 [INFO]  Packager 1 have been created
 [INFO]  Packager 3 have been created
 [INFO]  Thread 2 have been created for publisher type 1
 [INFO]  Publisher 2 of type 1 : Book1_11 is published and put into the buffer 1.
 [INFO]  Publisher 2 of type 1 : Book1_12 is published and put into the buffer 1.
 [INFO]  Publisher 2 of type 1 : Book1_13 is published and put into the buffer 1.
 [INFO]  Publisher 2 of type 1 : Book1_14 is published and put into the buffer 1.
 [INFO]  Publisher 2 of type 1 : Buffer is full. Resizing the buffer.
 [INFO]  Publisher 2 of type 1 : Book1_15 is published and put into the buffer 1.
 [INFO]  Publisher 2 of type 1 Finished publishing 5 books. Exiting the system
 [INFO]  Packager 1 : Put Book1_1 into the package.
 [INFO]  Packager 1 : Put Book2_1 into the package.
 [INFO]  Packager 1 : Put Book2_2 into the package.
 [INFO]  Packager 1 : Put Book2_3 into the package.
 [INFO]  Packager 1 : Put Book2_4 into the package.
 [INFO]  Packager 4 have been created
 [INFO]  Packager 1 : Put Book2_5 into the package.
 [INFO]  Packager 2 have been created
 [INFO]  Packager 1 : Package is full: Book1_1 Book2_1 Book2_2 Book2_3 Book2_4 Book2_5
```


Activities
Visual Studio Code
Wed 22:59
Assignment 3 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT
1: bash
main* ⊗0 ⚠0 Ali Live Share

```
 [INFO]  Publisher 1 of type 2 : Book2_8 is published and put into the buffer 2.
 [INFO]  Publisher 1 of type 2 : Book2_9 is published and put into the buffer 2.
 [INFO]  Publisher 1 of type 2 : Book2_10 is published and put into the buffer 2.
 [INFO]  Publisher 1 of type 2 Finished publishing 5 books. Exiting the system
 [INFO]  Packager 1 : Put Book2_8 into the package.
 [INFO]  Packager 1 : Put Book1_3 into the package.
 [INFO]  Publisher 2 of type 2 : Book2_11 is published and put into the buffer 2.
 [INFO]  Publisher 2 of type 2 : Book2_12 is published and put into the buffer 2.
 [INFO]  Publisher 2 of type 2 : Book2_13 is published and put into the buffer 2.
 [INFO]  Publisher 2 of type 2 : Book2_14 is published and put into the buffer 2.
 [INFO]  Publisher 2 of type 2 : Buffer is full. Resizing the buffer.
 [INFO]  Publisher 2 of type 2 : Book2_15 is published and put into the buffer 2.
 [INFO]  Publisher 2 of type 2 Finished publishing 5 books. Exiting the system
 [INFO]  Packager 1 : Put Book2_9 into the package.
 [INFO]  Packager 1 : Put Book2_10 into the package.
 [INFO]  Packager 1 : Package is full: Book1_2 Book2_6 Book2_8 Book1_3 Book2_9 Book2_10
 [INFO]  Packager 1 : Put Book2_5 into the package.
 [INFO]  Packager 1 : Put Book2_12 into the package.
 [INFO]  Packager 1 : Put Book1_4 into the package.
 [INFO]  Packager 1 : Put Book2_13 into the package.
 [INFO]  Packager 1 : Put Book1_5 into the package.
 [INFO]  Packager 1 : Put Book1_6 into the package.
 [INFO]  Packager 1 : Package is full: Book2_5 Book2_12 Book1_4 Book2_13 Book1_5 Book1_6
 [INFO]  Packager 1 : Put Book1_7 into the package.
 [INFO]  Packager 1 : Put Book2_14 into the package.
 [INFO]  Packager 1 : Put Book2_15 into the package.
 [INFO]  Packager 1 : Put Book2_15 into the package.
 [INFO]  Packager 1 : Put Book1_8 into the package.
 [INFO]  Packager 1 : Put Book1_9 into the package.
 [INFO]  Packager 1 : Package is full: Book1_7 Book2_14 Book2_15 Book2_15 Book1_8 Book1_9
 [INFO]  Packager 1 : Put Book1_10 into the package.
 [INFO]  Packager 1 : Put Book1_11 into the package.
 [INFO]  Packager 1 : Put Book1_12 into the package.
 [INFO]  Packager 1 : Put Book1_13 into the package.
 [INFO]  Packager 1 : Put Book1_14 into the package.
 [INFO]  Packager 1 : Put Book1_15 into the package.
There are no publishers left in the system.
Only 6 of 6 number of books could be packaged.
The package contains:
Book1_10 Book1_11 Book1_12 Book1_13 Book1_14 Book1_15 Exiting the system.
main waited successfully
ali@ali-Inspiron-5521:~/Documents/University Planet/3rd Year/5th Semester/CSE3033 - Operating Systems/OS-Assignments-Fall20/Assignment 3$
```