MARMARA UNIVERSITY

FACULTY OF ENGINEERING
COMPUTER SCIENCE & ENGINEERING DEPARTMENT

CSE3033
OPERATING SYSTEMS
Report of Programming Assignment #3

A.Tunahan Cinsoy – 150117062
Enver Aslan – 150115851
Cem Güleç – 150117828

Implementation Details

Implementation of our program starts with appropriate struct definitions which will be the key elements of the execution process. Respectively, publisher, packager, book, buffer (and also thread_arguments with thread_arguments_2 for pthread_create function call) structs are the main data structures.

In main function, we declare all of our structures with appropriate types. For example:

- $publisher\ allPublishers[publisherTypeCount][publisherThreadCount];$

declares a 2-D array to store publishers of our system. If there are two types of publishers and 5 publishers for each of types, this line will be:

  - $publisher\ allPublishers[2][5];$

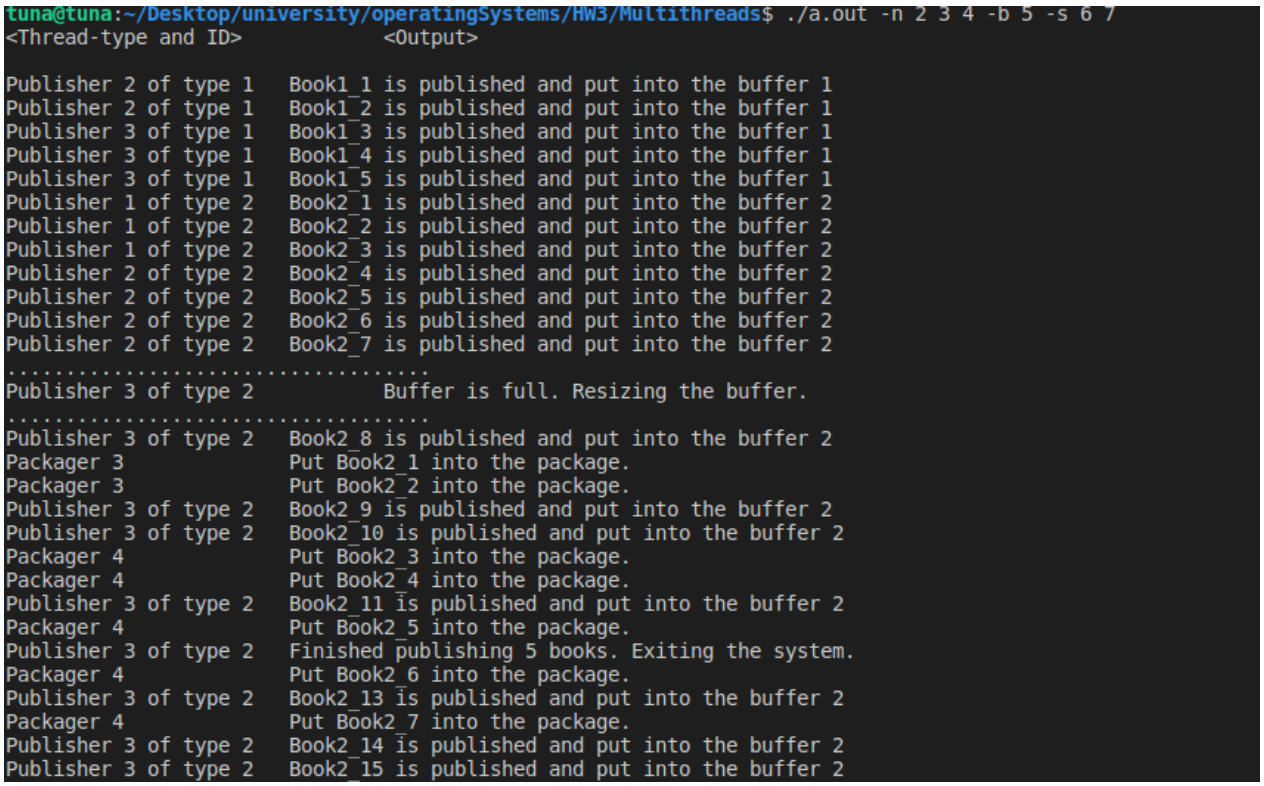
While we are declaring the buffers, we've implemented our architecture with "Different buffer types have their own buffers, so all buffers have to have their own semaphores and mutexes." approach.

After these declarations, we start multithreading process with pthread_create() built-in function for both publishers and packagers. Each of them has their own input argument which will be passed into their functions as input argument. (Referring to the first paragraph, thread_arguments and thread_arguments_2 will be input parameters for them, respectively.)

Below main function, we can see that there are various functions that have different jobs for creating a better separation of concern hierarchy. All of them are clearly commented out, so it might be better to understand them while looking at code.

Down below, we will state the problems and bugs of our execution with appropriate screenshots of our executions.

Screenshots and Explanations



```
tuna@tuna:~/Desktop/university/operatingSystems/HW3/Multithreads$ ./a.out -n 2 3 4 -b 5 -s 6 7
<Thread-type and ID>            <Output>

Publisher 2 of type 1   Book1_1 is published and put into the buffer 1
Publisher 2 of type 1   Book1_2 is published and put into the buffer 1
Publisher 3 of type 1   Book1_3 is published and put into the buffer 1
Publisher 3 of type 1   Book1_4 is published and put into the buffer 1
Publisher 3 of type 1   Book1_5 is published and put into the buffer 1
Publisher 1 of type 2   Book2_1 is published and put into the buffer 2
Publisher 1 of type 2   Book2_2 is published and put into the buffer 2
Publisher 1 of type 2   Book2_3 is published and put into the buffer 2
Publisher 2 of type 2   Book2_4 is published and put into the buffer 2
Publisher 2 of type 2   Book2_5 is published and put into the buffer 2
Publisher 2 of type 2   Book2_6 is published and put into the buffer 2
Publisher 2 of type 2   Book2_7 is published and put into the buffer 2
.................................
Publisher 3 of type 2           Buffer is full. Resizing the buffer.
.................................
Publisher 3 of type 2   Book2_8 is published and put into the buffer 2
Packager 3              Put Book2_1 into the package.
Packager 3              Put Book2_2 into the package.
Publisher 3 of type 2   Book2_9 is published and put into the buffer 2
Publisher 3 of type 2   Book2_10 is published and put into the buffer 2
Packager 4              Put Book2_3 into the package.
Packager 4              Put Book2_4 into the package.
Publisher 3 of type 2   Book2_11 is published and put into the buffer 2
Packager 4              Put Book2_5 into the package.
Publisher 3 of type 2   Finished publishing 5 books. Exiting the system.
Packager 4              Put Book2_6 into the package.
Publisher 3 of type 2   Book2_13 is published and put into the buffer 2
Packager 4              Put Book2_7 into the package.
Publisher 3 of type 2   Book2_14 is published and put into the buffer 2
Publisher 3 of type 2   Book2_15 is published and put into the buffer 2
```

- As seen above, we've implemented the multithreading functionality of the program. Books are getting published by several publishers, packagers are packaging books, when the buffer size is full, buffer gets resized by doubling its previous size. However, at some point, code gets stuck and does not continue. It might be happening because of a deadlock which is an issue that we couldn't solve unfortunately.

```
tuna@tuna:~/Desktop/university/operatingSystems/HW3/Multithreads$ ./a.out -n 2 3 4 -b 5 -s 6 7
<Thread-type and ID>            <Output>

Publisher 3 of type 1   Book1_1 is published and put into the buffer 1
Publisher 1 of type 2   Book2_1 is published and put into the buffer 2
Publisher 3 of type 2   Book2_2 is published and put into the buffer 2
Publisher 3 of type 2   Book2_3 is published and put into the buffer 2
Publisher 3 of type 2   Book2_4 is published and put into the buffer 2
Packager 3              Put Book2_1 into the package.
Packager 4              Put Book2_2 into the package.
Packager 4              Put Book2_3 into the package.
Publisher 3 of type 2   Book2_5 is published and put into the buffer 2
.................................
Publisher 3 of type 2   Finished publishing 5 books. Exiting the system.
.................................
Publisher 3 of type 2   Book2_7 is published and put into the buffer 2
Publisher 3 of type 2   Book2_8 is published and put into the buffer 2
Publisher 3 of type 2   Book2_9 is published and put into the buffer 2
Packager 4              Put Book2_4 into the package.
Packager 4              Put Book2_5 into the package.
Publisher 3 of type 2   Book2_10 is published and put into the buffer 2
Publisher 3 of type 2   Book2_11 is published and put into the buffer 2
Publisher 3 of type 2   Book2_12 is published and put into the buffer 2
Packager 4              Put Book2_6 into the package.
Publisher 3 of type 2   Book2_13 is published and put into the buffer 2
Packager 4              Put Book2_7 into the package.
Packager 4              Finished preparing one package. The package contains:
                        Book2_2 Book2_3 Book2_4 Book2_5 Book2_6 Book2_7
Publisher 3 of type 2   Book2_14 is published and put into the buffer 2
```

- In the above execution output, we can see that once a publisher publishes number of books that should be published, that publisher exits the program. However, as seen above, “Publisher 3 of type 2” does not leave the program, and continues to publish books and preventing other publishers to publish their own books. This is another issue of our system.

```
tuna@tuna:~/Desktop/university/operatingSystems/HW3/Multithreads$ ./a.out -n 2 3 4 -b 5 -s 6 7
<Thread-type and ID>            <Output>

Publisher 2 of type 1   Book1_1 is published and put into the buffer 1
Publisher 3 of type 1   Book1_2 is published and put into the buffer 1
Publisher 1 of type 2   Book2_1 is published and put into the buffer 2
Publisher 2 of type 2   Book2_2 is published and put into the buffer 2
Publisher 3 of type 2   Book2_3 is published and put into the buffer 2
Publisher 3 of type 2   Book2_4 is published and put into the buffer 2
Packager 2              Put Book2_1 into the package.
Packager 4              Put Book2_2 into the package.
Packager 4              Put Book2_3 into the package.
Publisher 3 of type 2   Book2_5 is published and put into the buffer 2
Publisher 3 of type 2   Book2_6 is published and put into the buffer 2
................................
Publisher 3 of type 2   Finished publishing 5 books. Exiting the system.
................................
Publisher 3 of type 2   Book2_8 is published and put into the buffer 2
Publisher 3 of type 2   Book2_9 is published and put into the buffer 2
Publisher 3 of type 2   Book2_10 is published and put into the buffer 2
Packager 4              Put Book2_4 into the package.
Packager 4              Put Book2_5 into the package.
Publisher 3 of type 2   Book2_11 is published and put into the buffer 2
Publisher 3 of type 2   Book2_12 is published and put into the buffer 2
Packager 4              Put Book2_6 into the package.
Publisher 3 of type 2   Book2_13 is published and put into the buffer 2
Packager 4              Put Book2_7 into the package.
Packager 4              Finished preparing one package. The package contains:
                        Book2_2 Book2_3 Book2_4 Book2_5 Book2_6 Book2_7
Publisher 3 of type 2   Book2_14 is published and put into the buffer 2
```

- When a package gets finished, packager prints out the books that package contains and continues to pack even more books. So, we assumed that all packagers only have one package, once it gets full, packagers use their own packages after taking out all books.

```
tuna@tuna:~/Desktop/university/operatingSystems/HW3/Multithreads$ ./a.out -n 2 3 4 -b 5 -s 6 7
<Thread-type and ID>            <Output>

Publisher 2 of type 1   Book1_1 is published and put into the buffer 1
Publisher 3 of type 1   Book1_2 is published and put into the buffer 1
Publisher 1 of type 2   Book2_1 is published and put into the buffer 2
Publisher 2 of type 2   Book2_2 is published and put into the buffer 2
Publisher 3 of type 2   Book2_3 is published and put into the buffer 2
Publisher 3 of type 2   Book2_4 is published and put into the buffer 2
Packager 3              Put Book2_1 into the package.
Packager 4              Put Book1_1 into the package.
Packager 4              Put Book2_2 into the package.
Packager 4              Put Book2_3 into the package.
Publisher 3 of type 2   Book2_5 is published and put into the buffer 2
Publisher 3 of type 2   Book2_6 is published and put into the buffer 2
................................
Publisher 3 of type 2   Finished publishing 5 books. Exiting the system.
................................
Publisher 3 of type 2   Book2_8 is published and put into the buffer 2
Publisher 3 of type 2   Book2_9 is published and put into the buffer 2
Publisher 3 of type 2   Book2_10 is published and put into the buffer 2
Packager 4              Put Book2_4 into the package.
Packager 4              Put Book2_5 into the package.
Packager 4              Put Book1_2 into the package.
Packager 4              Finished preparing one package. The package contains:
                        Book1_1 Book2_2 Book2_3 Book2_4 Book2_5 Book1_2
Publisher 3 of type 2   Book2_11 is published and put into the buffer 2
Publisher 3 of type 2   Book2_12 is published and put into the buffer 2
Packager 4              Put Book2_6 into the package.
Packager 4              Put Book2_7 into the package.
Publisher 3 of type 2   Book2_13 is published and put into the buffer 2
Publisher 3 of type 2   Book2_14 is published and put into the buffer 2
Packager 4              Put Book2_8 into the package.
Publisher 3 of type 2   Book2_15 is published and put into the buffer 2
Packager 4              Put Book2_9 into the package.
Publisher 3 of type 2   Book2_16 is published and put into the buffer 2
Packager 4              Put Book2_10 into the package.
Publisher 3 of type 2   Book2_17 is published and put into the buffer 2
```

- Above execution simply summarizes all of our execution outputs with the problems that have been stated.

References

- https://medium.com/@charlesdobson/how-to-implement-a-simple-circular-buffer-in-c-34b7e945d30e
- https://www.sitesbay.com/cprogramming/c-buffer-concept
- https://stackoverflow.com/questions/9369873/sem-init-what-is-the-value-parameter-for
- https://man7.org/linux/man-pages/man3/pthread_create.3.html