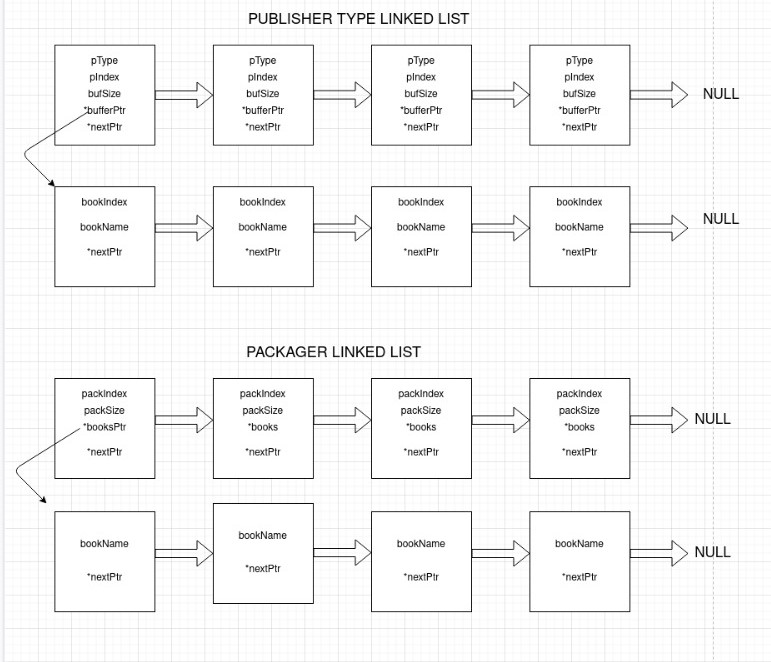
**PROJECT-3 REPORT**

Introduction :

In this Project we are expected to design a system which consists of two different type of threads. One of them is Publisher Type and another one is Packager Type. Publisher thread will publish books and put them into its own type buffer whereas packager threads will package that books by selecting buffer type randomly. Preserving consistency and preventing deadlock are major issues in this project.

Structure :



This is the structure that we used to implement hierarchy between threads. Each publisher type is hold as node of linkedlist and their buffer is added as a another linkedlist. Packager implementation is same as publisher. This is just the draft. That means , variables could be different.

Implementation Details :

First of all , we checked arguments by using “*getAndCheckArguments*” function. In this function , we handled all type of errors we can encounter. Order of arguments is the major issue in that function. Also we got parameters that we will use for creating threads.

Then we initialized thread arrays which are “publishers” and “packagers”. Their size is given via command line. After that , according to the parameters , thread count – thread type etc. , we created our linkedlists to hold thread types , buffers and packagers. All the nodes are created at the beginning of the program including buffer nodes. Start pointer of these linkedlists are global variables which means they can be used by all type of threads.

Publisher threads are assigned to the “publisher” function and packager thread are assigned to the “packager” function”. Struct is used to send parameters to these functions.

In the publisher , we determined the correct node of publisher type linkedlist by using type variable coming from parameter of that function. Then , we just allowed one publisher thread to work on buffer of this publisher type by using semaphore. We initiliazed semaphore as 1. That means it is same as mutex. We also used another semaphore not to allow any packager to work on buffer of this type. Then we checked the completeness of that buffer. If it is full , we doubled its size by using “resizeBuffer” function and publish new book by using “publishBook” function. If the thread published all books that it should do , then it leaves the system by unlocking the semaphores. We used “threadCount” variable in the publisher linkedlist . Thanks to this variable , we determined the number of thread belonging to this type.

In the packager , firstly we got the index which shows us the correct node of packager linkedlist. We created a infinite while loop to package all the books in the system . The most important part here is that only one packager should be able to reach one type of buffer. We put semaphore into the publisher struct type . Thanks to that we can lock this type while a packager is working on that buffer. If there is any book in that buffer , we just packaged it and at that point we checked the size of the packager package . If it is full , we resets it by using “printAndResetBuffer” function. If there is not any book in the corresponding buffer , we checked whether there is any publisher thread of this type or not by using “threadCount” variable located in the publisher struct. If there is , we waited for that thread by using “pthread\_join” and then we packaged book . If there is not any books and any threads , then we just unlocked that critical section and allow packager thread to leave program by using “pthread\_exit”.

The Most Critical Parts :

* Each publisher node has its own semaphore . Thanks to that , we locked the buffer that we want to lock easily. Semaphores are initialized as 1 , which means they are as same as mutex.
* We are needed to wait for corresponding thread when it is necessary . We put thread instances into the publisher struct as an array. Thanks to that we reached threads and waited for them easily.

What Is Missing ?

Because packager thread are working concurrently , when they need to print their info messages, there can be little bit disorder situation . Packager prints its items and new line(“\n”) in different printf() functions. Meanwhile there can be another packager thread that prints its own information . At that point you can see some irregularity in output.

Yunus Emre Ertunç – 150117064

Muhammed Enes Aktürk - 150117036