Min Kyaw - Section 05 - 018182136

Donovan Lee - Section 04 - 016741645

Lab 1 Report

In the header, we have many libraries included in order for the code to work. We have 3 global variables, one of which is a shared variable. As the threads get created, the shared variable will be the one getting accessed across the multiple threads. We also have a lock of a pthread mutex type and a barrier of a pthread barrier type, in which both are used in the SimpleThread method. In the SimpThread method, an integer value is being passed in which is the number of threads to be created in the main function. In our implementation of the method after the loop which was given to us, a mutex lock is created and the reference of the lock global variable is passed in to ensure that the other threads are not interrupting the shared variable for running our code with synchronization. After, we increment the shared variable into the variable val. The mutex lock is unlocked and the pthread_barrier_wait is created to pass in the global barrier variable in which it waits for the other threads to finish. If you run this code without the -DPTHREAD_SYNC, the lock and the barrier will not be implemented and the shared variable will be interrupted in which the final value will not be the same.

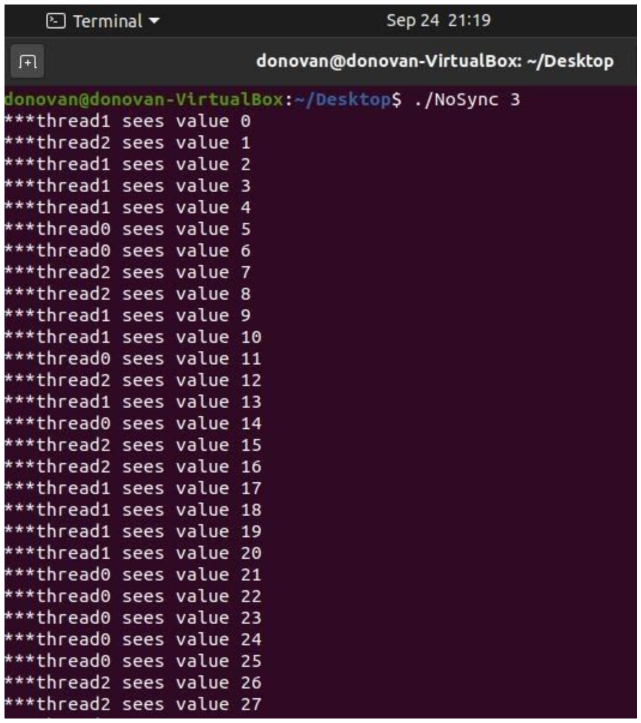
For our ThreadID function, we take in a void parameter because our pthread_create method creates an entry point for threads to run in. We cast our parameter as a long so we can run it in our SimpleThread function.

We implemented our ThreadID this way because pthread_create needed a void function to take in so that it can read an i value that was double casted as a void and a long.

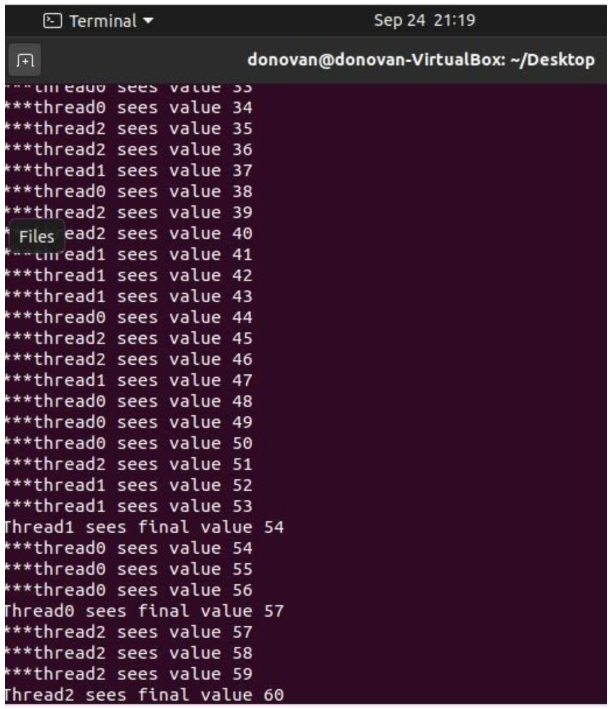
For the main function, it takes in a parameter and is checked with validations to see if there was only one parameter given and if the parameter is a positive number. After the parameter is checked we put it into a variable and initialize the first barrier to run the threads. We created 2 for loops, one for loop creates the threads and the other for loop stops the next thread's execution so the first thread can finish.

We implemented the main like this because we have to check if the parameter the user gives us is a good parameter or else the file won't run. Then we initialize a barrier so threads can be separated and let in one by one. With our pthread_join, it makes our thread run one by one so nothing is mixed allowing the ending threads to be synced.

Pthreads without synchronization

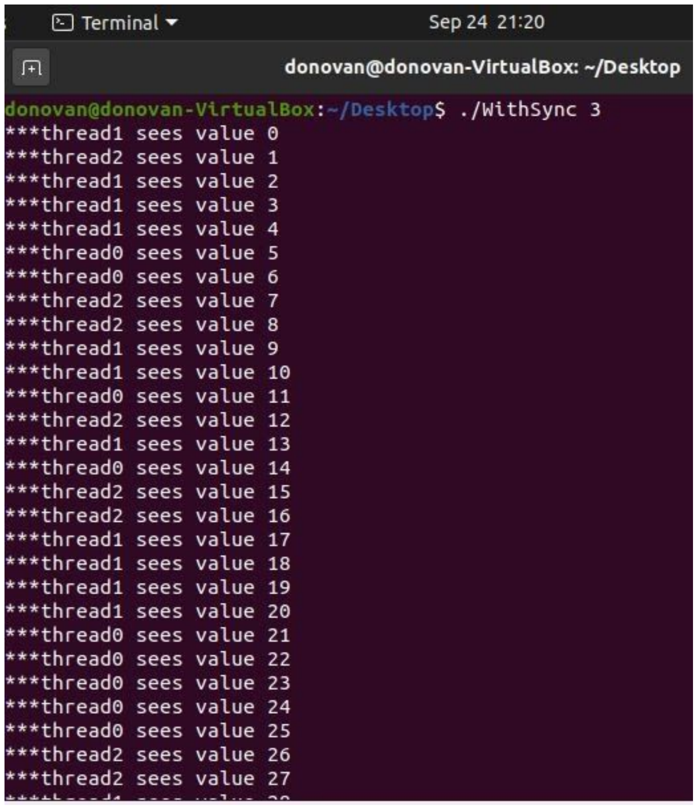


```
donovan@donovan-VirtualBox:~/Desktop$ ./NoSync 3
***thread1 sees value 0
***thread2 sees value 1
***thread1 sees value 2
***thread1 sees value 3
***thread1 sees value 4
***thread0 sees value 5
***thread0 sees value 6
***thread2 sees value 7
***thread2 sees value 8
***thread1 sees value 9
***thread1 sees value 10
***thread0 sees value 11
***thread2 sees value 12
***thread1 sees value 13
***thread0 sees value 14
***thread2 sees value 15
***thread2 sees value 16
***thread1 sees value 17
***thread1 sees value 18
***thread1 sees value 19
***thread1 sees value 20
***thread0 sees value 21
***thread0 sees value 22
***thread0 sees value 23
***thread0 sees value 24
***thread0 sees value 25
***thread2 sees value 26
***thread2 sees value 27
```

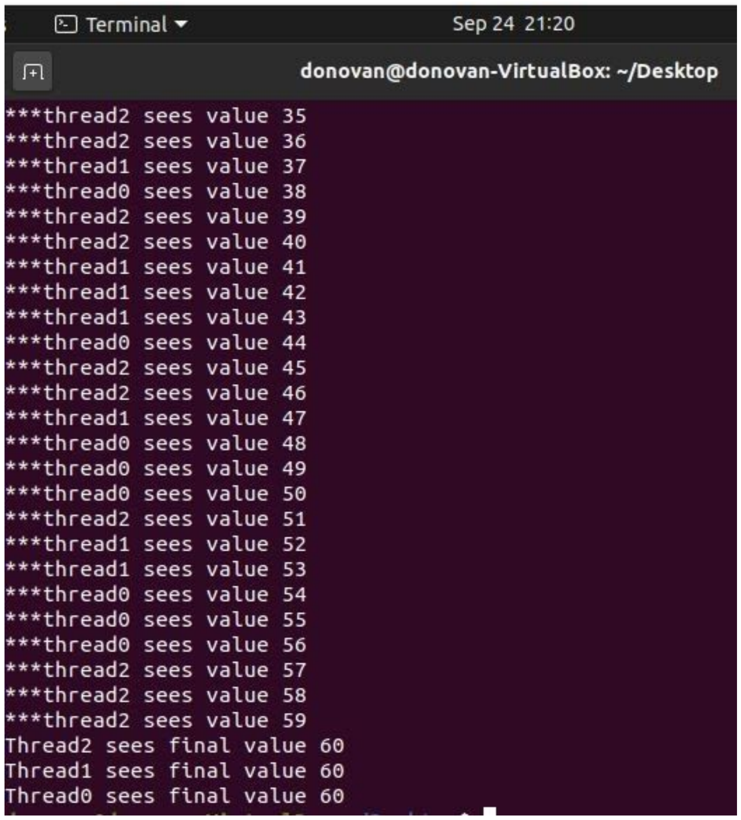


```
***thread0 sees value 33
***thread0 sees value 34
***thread2 sees value 35
***thread2 sees value 36
***thread1 sees value 37
***thread0 sees value 38
***thread2 sees value 39
*      ead2 sees value 40
***thread1 sees value 41
***thread1 sees value 42
***thread1 sees value 43
***thread0 sees value 44
***thread2 sees value 45
***thread2 sees value 46
***thread1 sees value 47
***thread0 sees value 48
***thread0 sees value 49
***thread0 sees value 50
***thread2 sees value 51
***thread1 sees value 52
***thread1 sees value 53
Thread1 sees final value 54
***thread0 sees value 54
***thread0 sees value 55
***thread0 sees value 56
Thread0 sees final value 57
***thread2 sees value 57
***thread2 sees value 58
***thread2 sees value 59
Thread2 sees final value 60
```

Pthreads with synchronization



```
donovan@donovan-VirtualBox:~/Desktop$ ./WithSync 3
***thread1 sees value 0
***thread2 sees value 1
***thread1 sees value 2
***thread1 sees value 3
***thread1 sees value 4
***thread0 sees value 5
***thread0 sees value 6
***thread2 sees value 7
***thread2 sees value 8
***thread1 sees value 9
***thread1 sees value 10
***thread0 sees value 11
***thread2 sees value 12
***thread1 sees value 13
***thread0 sees value 14
***thread2 sees value 15
***thread2 sees value 16
***thread1 sees value 17
***thread1 sees value 18
***thread1 sees value 19
***thread1 sees value 20
***thread0 sees value 21
***thread0 sees value 22
***thread0 sees value 23
***thread0 sees value 24
***thread0 sees value 25
***thread2 sees value 26
***thread2 sees value 27
```



```
***thread2 sees value 35
***thread2 sees value 36
***thread1 sees value 37
***thread0 sees value 38
***thread2 sees value 39
***thread2 sees value 40
***thread1 sees value 41
***thread1 sees value 42
***thread1 sees value 43
***thread0 sees value 44
***thread2 sees value 45
***thread2 sees value 46
***thread1 sees value 47
***thread0 sees value 48
***thread0 sees value 49
***thread0 sees value 50
***thread2 sees value 51
***thread1 sees value 52
***thread1 sees value 53
***thread0 sees value 54
***thread0 sees value 55
***thread0 sees value 56
***thread2 sees value 57
***thread2 sees value 58
***thread2 sees value 59
Thread2 sees final value 60
Thread1 sees final value 60
Thread0 sees final value 60
```

The difference between the pthread with no synchronization and pthread with synchronization is that when you don't enter in the -DPTHREAD_SYNC to run the terminal, the program will not have synchronization as the lock/unlock and the pthread_barrier_wait method will not be implemented. The reason being is that these blocks of code are specific to a scope of synchronization, with #ifdef PTHREAD_SYNC and #endif, meaning it will only run synchronized once you enter in the right command in the terminal. This is why the shared variable gets interrupted by other threads and the final value is not the same without synchronization.

Contributions

We contributed the same amount of work because we were doing it at the same time over a communication platform, giving each other inputs on how to approach certain things in the lab.