ASSIGNMENT 2 REPORT

Multi-process and multi-threaded print server

HARITA AGARWAL

04.05.2023
CMSC 312

WORKING PORTIONS OF THE CODE:

LIFO

- The program stores data in shared memory segments and manages producers and consumers through structs.
- Producers are created as separate processes, whereas consumers are created as threads.
- The program uses shared memory semaphores to synchronize data access(buffer_index, full, empty, buffer_mutex, buffer) and a signal handler to gracefully terminate threads(CTRL-C).
- Threads will not be terminated until all producers have finished their tasks. (All forks must exit before any threads exit.)
- Command line parameters provide the total number of producers and consumers.
- The program deallocates shared memory and semaphores once all jobs have been completed.
- The program also displays the total time required for execution and average wait time for threads and processes.

FIFO

- The program stores data in shared memory segments and manages producers and consumers through structs.
- Producers are created as separate processes, whereas consumers are created as threads.
- The program uses shared memory semaphores to synchronize data access(read, write, full, empty, buffer_mutex, buffer, buffer_index) and a signal handler to gracefully terminate threads(CTRL-C).
- Threads will not be terminated until all producers have finished their tasks. (All forks must exit before any threads exit.)
- Command line parameters provide the total number of producers and consumers.
- The program deallocates shared memory and semaphores once all jobs have been completed.
- The program also displays the total time required for execution and average wait time for threads and processes.

NON WORKING PORTIONS

LIFO

- none

FIFO

- none

LOGIC USED TO IDENTIFY THE TERMINATING CONDITION

1. The PRODUCER_LOOPS constant, which is set to 2, specifies the termination condition for the producer threads. Each producer thread creates a random number of jobs to add to the buffer, with a randomly generated size for each work. The producer threads add jobs to the buffer until they have created the amount of loops given, at which point they exit.
2. The consumer's termination conditions might be any situation in which the consumer no longer requires or desires the produced items, or when the system is no longer capable of sending the generated items to the consumer.

HOW WERE THE SEMAPHORES AND OTHER BOOK KEEPING VARIABLES SHARED BETWEEN PROCESSES AND THREADS

➔ The program uses the shared memory to construct shared memory segments and allocate variables to them. The shmget function is used to create shared memory regions with a given key and size, while shmat is used to attach to a process's virtual address space. Pointers are used to access variables in shared memory. For example, in this program's code, the buffer, buffer_index, total_jobs, and shmid_value variables are all allocated in shared memory. Furthermore, the program utilizes semaphores, which are likewise allocated in shared memory using shmget and shmat. The sem_t type is used to declare the semaphores, which are subsequently initialized using sem_init. Semaphores are used to synchronize access to shared resources between processes and threads. It is important to note that appropriate synchronization of semaphores and shared memory use is critical in order to avoid race situations and other problems. The sem_wait and

sem_post functions are used in this application to verify that only one process or thread may access a shared resource at a time, and that access is properly synchronized.

WHAT WAS DONE IN THE SIGNAL HANDLER FOR GRACEFUL HANDLER FOR GRACEFUL TERMINATION

1. The given code includes two signal handlers: sigint_handler and sigkill_handler.
    a. When the program receives a SIGINT signal, which is frequently generated when the user hits Ctrl+C in the terminal, the sigint_handler function is run. When this signal is received, the handler function loops through all of the producer processes that happened and issues a SIGKILL signal to each of them using the kill system call. This terminates all producers and releases the system resources that had been assigned to them.
    b. When the program gets a SIGKILL signal, which is commonly generated when the program is terminated suddenly or violently, the sigkill_handler function is invoked. When this signal is received, the handler function detaches any shared memory segments and semaphores that the program generated with the shmdt system call and then quits the program using the exit system call. This guarantees that any system resources allocated by the program's execution are released before the program exits.

HOW DID THE LIFO AND FIFO IMPLEMENTATIONS DIFFER IN TERMS OF YOUR USAGE EITHER THE BUFFER_INDEX VARIABLE OR IN/OUT POINTERS (or some other method that you may use for the FIFO queue)

LIFO

The code employs a FIFO (First In First Out) implementation, in which jobs are added to the buffer's end and deleted from its beginning. An array of buffer_t structures is used to implement the buffer, and the buffer_index variable keeps track of the current index where a new task can be inserted. Insert Buffer adds a new task to the buffer at the current index, whereas dequeuebuffer removes a job from the beginning of the buffer by decrementing the index.

The producer function creates print jobs of varying sizes and inserts them into the buffer using insertbuffer while signaling the full_sem and empty_sem semaphores to

synchronize with the consumers. The consumer function calls dequeuebuffer to remove tasks from the buffer and signals the semaphores to synchronize with the producers.

Overall, in a LIFO implementation, the index would need to be modified to accommodate tasks being added and deleted from the buffer's beginning.

FIFO

The code implements FIFO and LIFO queues using the insert buffer() and dequeuebuffer() methods. Insert Buffer() adds a new buffer element to the buffer's end, whereas dequeuebuffer() removes the most recent element from the beginning.

If we wanted to build a LIFO queue, we'd need to update the insert buffer() and dequeuebuffer() functions.(). In particular, insert buffer() would have to insert the new buffer element at the beginning rather than the end of the buffer, and dequeuebuffer() would have to remove the most recently entered element from the end rather than the beginning of the buffer.

The FIFO implementation uses the buffer_index variable or in/out pointers to keep track of the buffer's place. The “in” pointer indicates the next available place in the buffer where a new element may be added, whereas the “out” pointer indicates the oldest element in the buffer that must be discarded. The insert buffer() method inserts the new element and increments in at the place indicated by “in ", whereas the dequeuebuffer() function removes the element indicated by out and increments out.

SAMPLE RUN FOR LIFO

```
^Cagarwalhp@egr-v-cmsc312-1:~/assignment2$ gcc LIFO.c -o a.out -g
agarwalhp@egr-v-cmsc312-1:~/assignment2$ ./a.out 4 4
Producer 1829835 added 66 to buffer
Consumer 139958584411712 dequeue <1829835, 66> from buffer
Producer 1829837 added 77 to buffer
Producer 1829838 added 2 to buffer
Consumer 139958576019008 dequeue <1829838, 2> from buffer
Consumer 139958567626304 dequeue <1829837, 77> from buffer
Producer 1829836 added 87 to buffer
Consumer 139958559233600 dequeue <1829836, 87> from buffer
Producer 1829835 added 66 to buffer
Consumer 139958584411712 dequeue <1829835, 66> from buffer
Producer 1829837 added 77 to buffer
Consumer 139958576019008 dequeue <1829837, 77> from buffer
Producer 1829838 added 2 to buffer
Consumer 139958567626304 dequeue <1829838, 2> from buffer
Producer 1829835 added 66 to buffer
Consumer 139958559233600 dequeue <1829835, 66> from buffer
Producer 1829836 added 87 to buffer
Consumer 139958584411712 dequeue <1829836, 87> from buffer
Producer 1829837 added 77 to buffer
Consumer 139958576019008 dequeue <1829837, 77> from buffer
Producer 1829838 added 2 to buffer
Consumer 139958567626304 dequeue <1829838, 2> from buffer
Producer 1829835 added 66 to buffer
Consumer 139958559233600 dequeue <1829835, 66> from buffer
Producer 1829837 added 77 to buffer
Consumer 139958584411712 dequeue <1829837, 77> from buffer
Producer 1829838 added 2 to buffer
Consumer 139958576019008 dequeue <1829838, 2> from buffer
Producer 1829836 added 87 to buffer
Consumer 139958567626304 dequeue <1829836, 87> from buffer
Producer 1829836 added 87 to buffer
Consumer 139958559233600 dequeue <1829836, 87> from buffer
The producer produced 16
The consumer consumed 16
Total execution time: 28581555649.396545 seconds
Average waiting time: 1681267979.376267 microseconds
```

```
agarwalhp@egr-v-cmsc312-1:~/assignment2$ gcc LIFO.c -o a.out -g
agarwalhp@egr-v-cmsc312-1:~/assignment2$ ./a.out 3 2
Producer 1830558 added 70 to buffer
Producer 1830559 added 91 to buffer
Consumer 140320623466048 dequeue <1830559, 91> from buffer
Consumer 140320615073344 dequeue <1830558, 70> from buffer
Producer 1830560 added 90 to buffer
Consumer 140320623466048 dequeue <1830560, 90> from buffer
^CThe producer produced 3
The consumer consumed 3
Total execution time: 3362536215.542439 seconds
Average waiting time: 1681268107.771219 microseconds
agarwalhp@egr-v-cmsc312-1:~/assignment2$
```

SAMPLE RUN FOR FIFO

```
agarwalhp@egr-v-cmsc312-1:~/assignment2$ gcc FIFO.c -o a.out -g
agarwalhp@egr-v-cmsc312-1:~/assignment2$ ./a.out 5 3
Producer 1827657 added 24 to buffer
Producer 1827658 added 62 to buffer
Consumer 139749142296128 dequeue <1827657, 24> from buffer
Consumer 139749133903424 dequeue <1827658, 62> from buffer
Producer 1827659 added 25 to buffer
Consumer 139749125510720 dequeue <1827659, 25> from buffer
Producer 1827661 added 90 to buffer
Producer 1827660 added 55 to buffer
Consumer 139749142296128 dequeue <1827661, 90> from buffer
Consumer 139749133903424 dequeue <1827660, 55> from buffer
Producer 1827657 added 24 to buffer
Producer 1827658 added 62 to buffer
Consumer 139749125510720 dequeue <1827657, 24> from buffer
Producer 1827659 added 25 to buffer
Consumer 139749142296128 dequeue <1827658, 62> from buffer
Consumer 139749133903424 dequeue <1827659, 25> from buffer
Producer 1827661 added 90 to buffer
Consumer 139749125510720 dequeue <1827661, 90> from buffer
Producer 1827660 added 55 to buffer
Producer 1827657 added 24 to buffer
Producer 1827658 added 62 to buffer
Producer 1827659 added 25 to buffer
Consumer 139749142296128 dequeue <1827660, 55> from buffer
Consumer 139749133903424 dequeue <1827657, 24> from buffer
Consumer 139749125510720 dequeue <1827658, 62> from buffer
Producer 1827657 added 24 to buffer
Producer 1827658 added 62 to buffer
Consumer 139749142296128 dequeue <1827659, 25> from buffer
Consumer 139749133903424 dequeue <1827657, 24> from buffer
Producer 1827661 added 90 to buffer
Producer 1827660 added 55 to buffer
Consumer 139749125510720 dequeue <1827658, 62> from buffer
Producer 1827659 added 25 to buffer
Consumer 139749133903424 dequeue <1827661, 90> from buffer
Consumer 139749142296128 dequeue <1827660, 55> from buffer
Consumer 139749125510720 dequeue <1827659, 25> from buffer
Producer 1827661 added 90 to buffer
```

```
Producer 1827660 added 55 to buffer
Consumer 139749125510720 dequeue <1827658, 62> from buffer
Producer 1827659 added 25 to buffer
Consumer 139749133903424 dequeue <1827661, 90> from buffer
Consumer 139749142296128 dequeue <1827660, 55> from buffer
Consumer 139749125510720 dequeue <1827659, 25> from buffer
Producer 1827661 added 90 to buffer
Producer 1827660 added 55 to buffer
Consumer 139749133903424 dequeue <1827661, 90> from buffer
Consumer 139749142296128 dequeue <1827660, 55> from buffer
The producer produced 20
The consumer consumed 20
Total execution time: 28581549040.537472 seconds
Average waiting time: 1681267590.619851 microseconds
```

```
agarwalhp@egr-v-cmsc312-1:~/assignment2$ ./a.out 3 2
Producer 1829142 added 32 to buffer
Consumer 140186356004416 dequeue <1829142, 32> from buffer
Producer 1829141 added 27 to buffer
Consumer 140186347611712 dequeue <1829141, 27> from buffer
Producer 1829143 added 11 to buffer
Consumer 140186356004416 dequeue <1829143, 11> from buffer
^Cagarwalhp@egr-v-cmsc312-1:~/assignment2$
```

EXECUTION TIME AND AVERAGE WAITING TIME PLOTS FOR LIFO FOR DIFFERENT VALUES OF NUMBER OF PRODUCER PROCESSES AND NUMBER OF CONSUMER TREADS.

PRODUCER	CONSUMERS	TOTAL EXECUTION TIME	AVERAGE WAITING TIME
2	2	6724047804.057915 seconds	2241349268.019305 microseconds
2	4	6724048696.846519 seconds	2241349565.615507 microseconds

2	6	6724049095.525388 seconds	1681012273.881347 microseconds
2	8	6724049423.156750 seconds	2241349807.718916 microseconds
2	10	6724049917.223559 seconds	1681012479.305890 microseconds
4	2	16810125977.965452 seconds	1681012597.796545 microseconds
4	4	13448101788.577734 seconds	1681012723.572217 microseconds
4	6	13448102682.996227 seconds	1681012835.374528 microseconds
4	8	13448103313.801155 seconds	1681012914.225144 microseconds
4	10	13448104151.897709 seconds	1681013018.987214 microseconds
6	2	26896210600.717922 seconds	1681013162.544870 microseconds
6	4	23534187904.323944 seconds	1681013421.737425 microseconds
6	6	20172162688.232559 seconds	1833832971.657505 microseconds
6	8	20172163920.846352 seconds	1681013660.070529 microseconds
6	10	20172165317.906090 seconds	1681013776.492174 microseconds
8	2	36983159300.154030 seconds	1681052695.461547 microseconds
8	4	33621056387.127880 seconds	1681052819.356394 microseconds
8	6	33621058205.634415 seconds	1681052910.281721 microseconds

8	8	26896848123.165138 seconds	1681053007.697821 microseconds
8	10	25215796396.258686 seconds	1681053093.083912 microseconds
10	2	47069490485.610085 seconds	1681053231.628932 microseconds
10	4	47069498086.794296 seconds	1681053503.099796 microseconds
10	6	40345287312.922981 seconds	1681053638.038458 microseconds
10	8	43707398017.892593 seconds	1748295920.715704 microseconds
10	10	33621077331.259956 seconds	1681053866.562998 microseconds

EXECUTION TIME AND AVERAGE WAITING TIME PLOTS FOR FIFO FOR DIFFERENT VALUES OF NUMBER OF PRODUCER PROCESSES AND NUMBER OF CONSUMER TREADS.

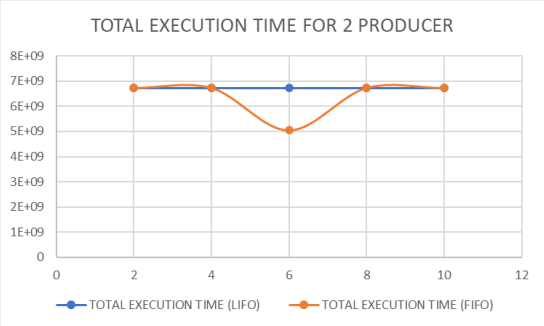
PRODUCER	CONSUMERS	TOTAL EXECUTION TIME	AVERAGE WAITING TIME
2	2	6724292255.337212 seconds	1681073063.834303 microseconds
2	4	6724292905.768911 seconds	1681073226.442228 microseconds
2	6	5043220158.999344 seconds	1681073386.333115 microseconds
2	8	6724293971.807667 seconds	2241431323.935889 microseconds
2	10	6724294343.152670 seconds	1681073585.788167 microseconds
4	2	10086445581.553217 seconds	1681074263.592203 microseconds
4	4	13448594742.938211 seconds	1681074342.867276 microseconds

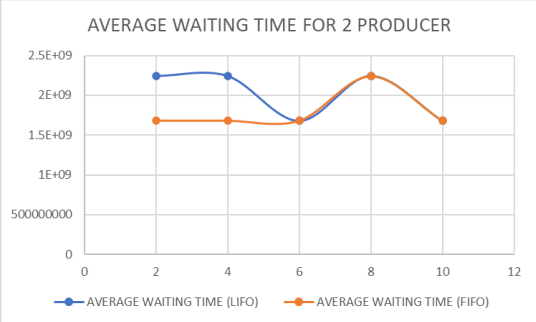
4	6	13448102682.996227 seconds	1681012835.374528 microseconds
4	8	13448597143.675413 seconds	1681074642.959427 microseconds
4	10	13448599693.956633 seconds	1681074961.744579 microseconds
6	2	10086450253.806261 seconds	1681075042.301044 microseconds
6	4	13448604116.492222 seconds	1681075514.561528 microseconds
6	6	20172911505.130623 seconds	1681075958.760885 microseconds
6	8	20172912461.679359 seconds	1681076038.473280 microseconds
6	10	20172918332.854980 seconds	1681076527.737915 microseconds
8	2	50437366885.877937 seconds	1681245562.862598 microseconds
8	4	47074883463.685966 seconds	1681245837.988785 microseconds
8	6	45393655172.235573 seconds	1681246487.860577 microseconds
8	8	47074907856.153763 seconds	1681246487.860577 microseconds
8	10	47074920246.286270 seconds	1681247151.653081 microseconds
10	2	63887398774.340958 seconds	1681247336.166867 microseconds
10	4	60524911813.689507 seconds	1681247550.380264 microseconds

10	6	60524919809.408943 seconds	1681247772.483582 microseconds
10	8	60524923784.034744 seconds	1681247882.889854 microseconds
10	10	60524933323.927055 seconds	1681248147.886863 microseconds

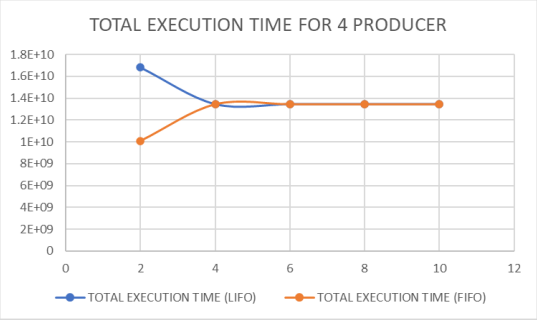
2-D graphs: 5 different graphs are needed for #producers = 2, 4, 6, 8, 10 respectively

- ### 2 PRODUCERS

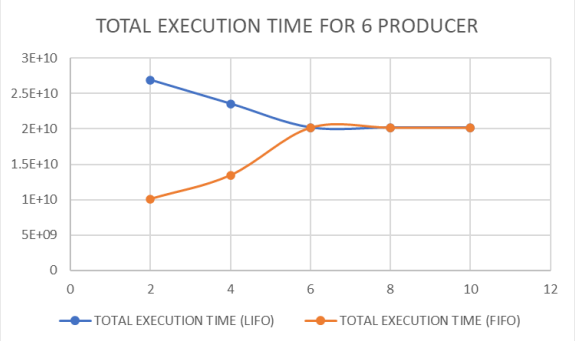


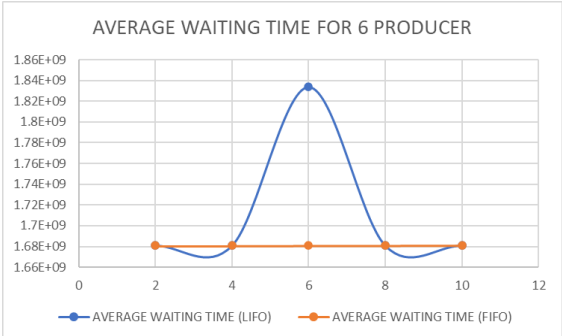


- ### 4 PRODUCERS

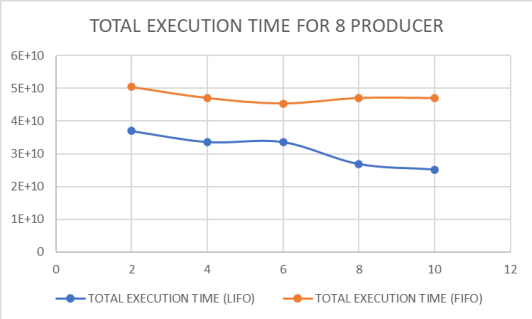


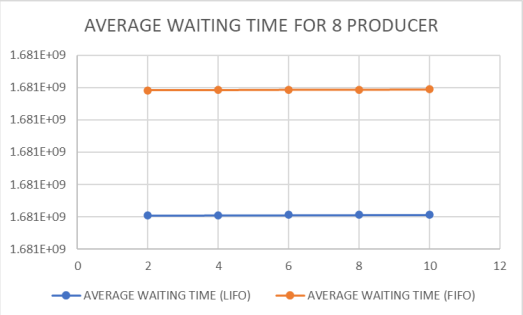
- 6 PRODUCER



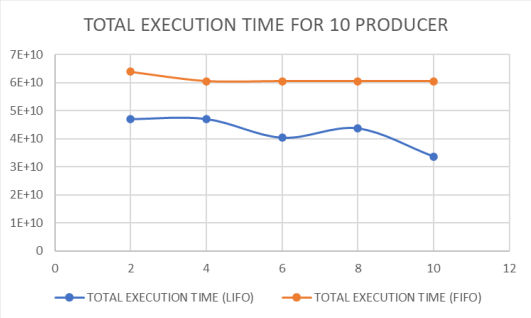


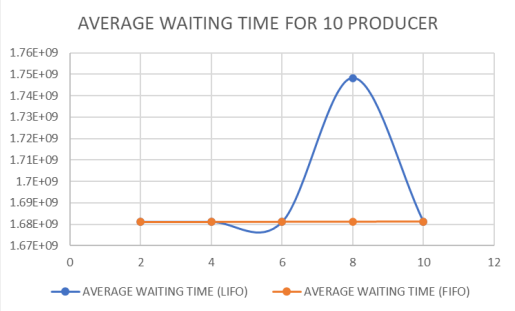
- 8 PRODUCER





- 10 PRODUCER


TOTAL EXECUTION TIME FOR 10 PRODUCER
7E+10
6E+10
5E+10
4E+10
3E+10
2E+10
1E+10
0
0
2
4
6
8
10
12
TOTAL EXECUTION TIME (LIFO)
TOTAL EXECUTION TIME (FIFO)


AVERAGE WAITING TIME FOR 10 PRODUCER
1.76E+09
1.75E+09
1.74E+09
1.73E+09
1.72E+09
1.71E+09
1.7E+09
1.69E+09
1.68E+09
1.67E+09
0
2
4
6
8
10
12
AVERAGE WAITING TIME (LIFO)
AVERAGE WAITING TIME (FIFO)