Report Checkpoint 4

Jonathan Sutedjo 鄭安良 111006207

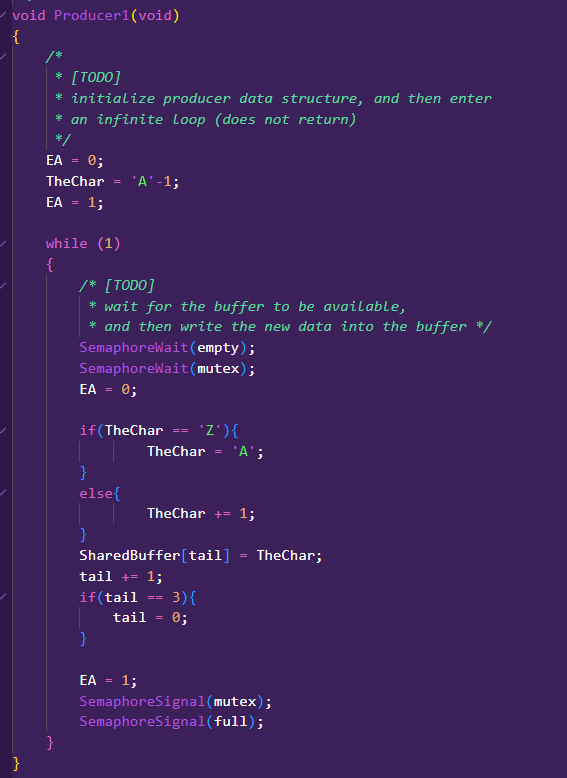
A screenshot of a computer screen

Description automatically generated

A screenshot of a computer program

Description automatically generated

For the test3threads.c file, I have an additional variable to store the temporary value of the numbers from producer2, the name is “TheInt”.



This is my producer1, where it is the same with checkpoint 3, producing out A-Z alphabet to the shared buffer.

A screenshot of a computer program

Description automatically generated

This is my producer2, where it is similar to producer1 but instead of alphabet it produces number from 0 to 9 and stored it to the shared buffer.

For my consumer, it stays the same as from checkpoint 3.

A screenshot of a computer program

Description automatically generated

In my main, I write the ThreadCreate in that sequence, and that would result in the printing of alphabets for the first 3, then the next 3 are numbers from 0 to 2, and then it change to alphabets again and so on. This change happen for the fairness. If I change the sequence such that Producer2 is first in the line of code, then it will print out the number from 0 to 2 first then comes the 3 alphabets a to c, and so on.

A screenshot of a computer

Description automatically generated

From the result of the Rx here, producer1 will fill in the first 3 of the so called shared buffer with their products and then the consumer will print it out or display it. After that, the other producer fill in the shared buffer and then the consumer will then print out the result of producer2. With this, it can be seen that it is being printed alternatively between producer 1 and 2 by switching every 3 characters.

A screenshot of a computer program

Description automatically generated A screenshot of a computer program

Description automatically generated

This is my code for the MyTimer0Handler, where I implemented another thing so there can be fairness there. In the preemptive.c file, I added another variable which is the “ProdCountID”. In the myTimer0Handler, it will first check wether the curThreadID or the current thread ID is 0 or not. In this case, if it is 0, it means it is for consumer, and the curThreadId will be assigned to the ProdThreadID that will then be loop between 1 to 3 every time, so that the next producer will not be the last one. If the current thread ID is not 0, then the curThreadID will be assign 0 so that the consumer will free the shared buffer.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Here, for we to see when the Producer1 is running, we can see the mutex(0x25), full(0x26), and empty(0x27). First, the empty will decrease by 1 and then the mutex and full increased by 1 to become 1-1-2. Then the empty will be decreased by 1 and loop till the empty is 0 and the full becomes 3 in this case.

A screenshot of a computer

Description automatically generated

After producer1 has finished printed out by consumer, the mutex full and empty will go back to 1-0-3 and here comes the producer2.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

For producer2, now we can see similar behaviour with when producer1 is running, we can see the mutex, full, and empty, where it first from 1-0-3 goes to 1-3-0, that is the full producer2 running.

When it is an unfair UART, then only one producer gets to produce the characters, either alphabet or numbers.

Here is an example of the unfair UART:

A screenshot of a computer program

Description automatically generated

Above shown the condition where only the producer1 gets to produce and so only alphabets gets printed out.

Here is an example of the fair UART:

A screenshot of a computer error

Description automatically generated

Here we can see that each producer produce 3 characters to the share buffer and then gets printed out by the consumer alternately.