MINI PROJECT 4

BLOCKING SEMAPHORES AND PRIORITY SCHEDULING

WILLIAM TRACE LACOUR (WTL6C)

PART 1 – EXPERIMENT 1

Watch 1

Name	Value	Type
DisplayCount	2479	unsigned long
ConsumerCount	1	unsigned long
tcbs[1].ArriveTime	2543	unsigned int
tcbs[1].WaitTime	0	unsigned int
tcbs[1].ExecCount	7497	unsigned int
tcbs[3].ArriveTime	2543	unsigned int
tcbs[3].WaitTime	2	unsigned int
tcbs[3].ExecCount	7494	unsigned int
<Enter expression>		

PART 1 – EXPERIMENT 2

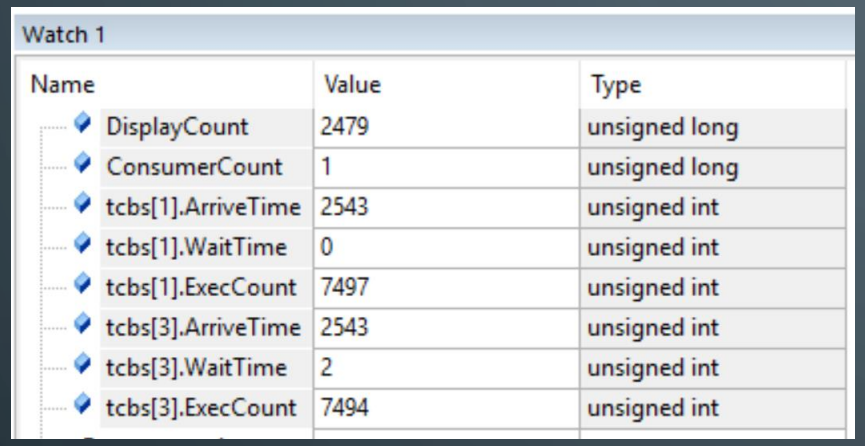
Watch 2

Name	Value	Type
Button1RespTime	13	unsigned long
Button2RespTime	1297	unsigned long
NumCreated	12	unsigned long
<Enter expression>		

PART 1 – QUESTION 1

- Task 2 – top button, quick response just makes a thread
  - SW1 (top button) triggers buttonWork background thread
- Task 3 – consumer, foreground periodic thread that accepts data from producer
  - Priority 1 – writing crosshairs and data to the LCD
- Task 6 – display, foreground periodic thread
  - Priority 3 – writes the psuedoCount to the LCD
- Task 7 – bottom button, slower response must delete each thread and restart
  - SW2 (bottom button) triggers the restart background thread

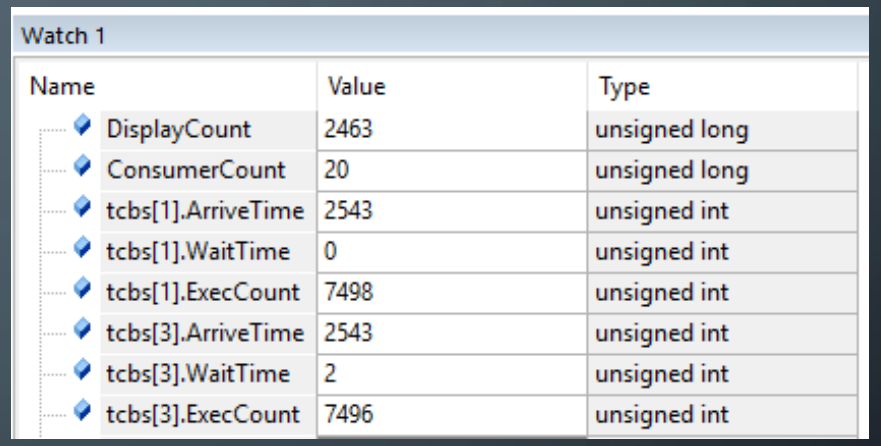
PART 1 – QUESTION 2A



Watch 1

Name	Value	Type
DisplayCount	2479	unsigned long
ConsumerCount	1	unsigned long
tcbs[1].ArriveTime	2543	unsigned int
tcbs[1].WaitTime	0	unsigned int
tcbs[1].ExecCount	7497	unsigned int
tcbs[3].ArriveTime	2543	unsigned int
tcbs[3].WaitTime	2	unsigned int
tcbs[3].ExecCount	7494	unsigned int

BEFORE



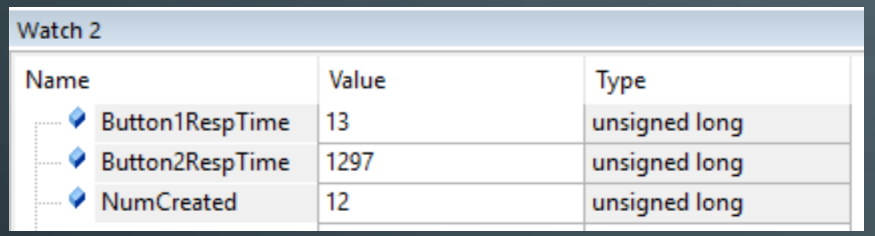
Watch 1

Name	Value	Type
DisplayCount	2463	unsigned long
ConsumerCount	20	unsigned long
tcbs[1].ArriveTime	2543	unsigned int
tcbs[1].WaitTime	0	unsigned int
tcbs[1].ExecCount	7498	unsigned int
tcbs[3].ArriveTime	2543	unsigned int
tcbs[3].WaitTime	2	unsigned int
tcbs[3].ExecCount	7496	unsigned int

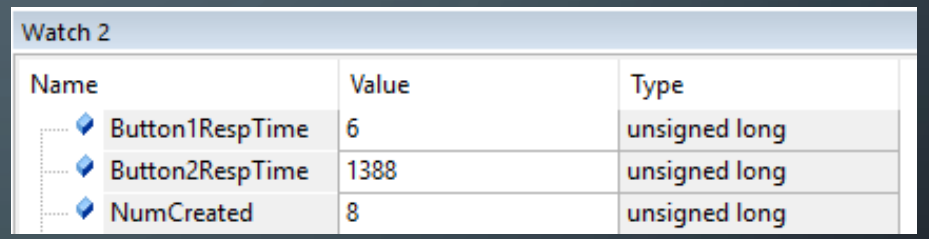
AFTER

I added OS_Sleep(TIME_1MS) right before the OS_Kill() command in the display thread and saw little to no difference. This is most likely because of the location of the sleep command. It will only run once the Pseudo Count gets too 299. So having no difference between either experiment makes sense. The next slide will show the second experiment results. The one difference I did see was the increase in Consumer Count and that was due to the 1 ms sleep.

PART 1 – QUESTION 2B



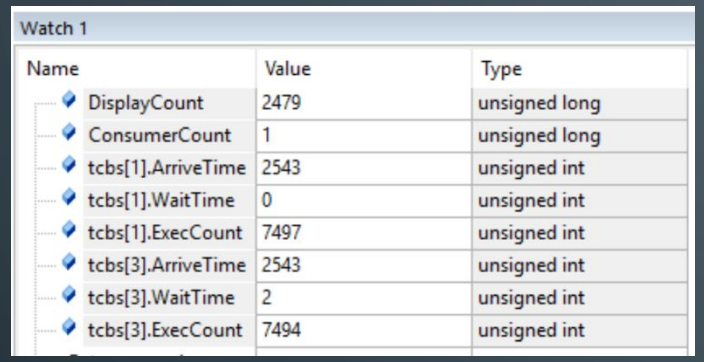
BEFORE



AFTER

The response time of the button1 was about half that of the previous experiment. This was due to the sleeping of 1ms for the display thread.

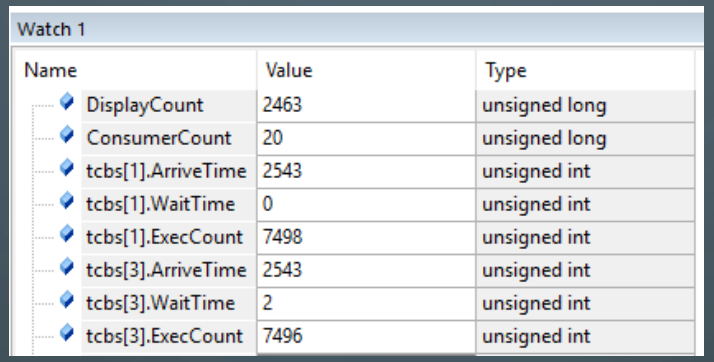
PART 1 – QUESTION 3A



Watch 1

Name	Value	Type
DisplayCount	2479	unsigned long
ConsumerCount	1	unsigned long
tcbs[1].ArriveTime	2543	unsigned int
tcbs[1].WaitTime	0	unsigned int
tcbs[1].ExecCount	7497	unsigned int
tcbs[3].ArriveTime	2543	unsigned int
tcbs[3].WaitTime	2	unsigned int
tcbs[3].ExecCount	7494	unsigned int

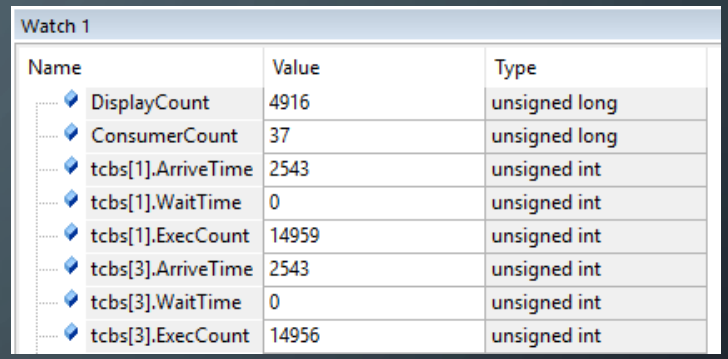
BEFORE



Watch 1

Name	Value	Type
DisplayCount	2463	unsigned long
ConsumerCount	20	unsigned long
tcbs[1].ArriveTime	2543	unsigned int
tcbs[1].WaitTime	0	unsigned int
tcbs[1].ExecCount	7498	unsigned int
tcbs[3].ArriveTime	2543	unsigned int
tcbs[3].WaitTime	2	unsigned int
tcbs[3].ExecCount	7496	unsigned int

Sleep(1ms)



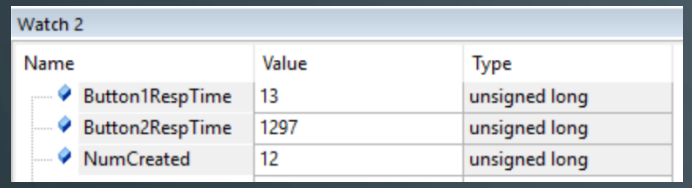
Watch 1

Name	Value	Type
DisplayCount	4916	unsigned long
ConsumerCount	37	unsigned long
tcbs[1].ArriveTime	2543	unsigned int
tcbs[1].WaitTime	0	unsigned int
tcbs[1].ExecCount	14959	unsigned int
tcbs[3].ArriveTime	2543	unsigned int
tcbs[3].WaitTime	0	unsigned int
tcbs[3].ExecCount	14956	unsigned int

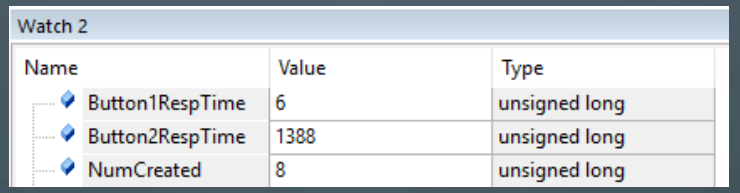
OS_Suspend()

I added OS_Suspend to every foreground thread right before it finished its execution. This made the wait times all go to zero. This also increased the execution count of the foreground threads because of the cooperation from each. This is proof of the cooperative spinlock versus a non cooperative spinlock thread.

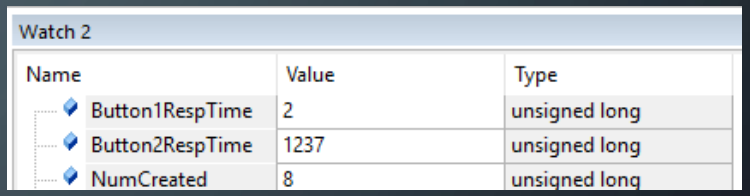
PART 1 – QUESTION 3B



BEFORE



OS_Sleep(1ms)



OS_Suspend()

Once again adding the cooperative part to the spinlock it decreased the response time even more. If a thread can't run it simply just suspends until it can.

PART 2 – EXPERIMENT 1

Watch 2

Name	Value	Type
Button1RespTime	<cannot evaluate>	uchar
Button2RespTime	<cannot evaluate>	uchar
NumCreated	6	unsigned long
SignalCount1	20000	unsigned long
SignalCount2	20000	unsigned long
SignalCount3	1960000	unsigned long
WaitCount1	1983956	unsigned long
WaitCount2	11554	unsigned long
WaitCount3	4490	unsigned long
<Enter expression>		

PART 2 – EXPERIMENT 2

Watch 2

Name	Value	Type
Button1RespTime	<cannot evaluate>	uchar
Button2RespTime	<cannot evaluate>	uchar
NumCreated	6	unsigned long
SignalCount1	20000	unsigned long
SignalCount2	20000	unsigned long
SignalCount3	1960000	unsigned long
WaitCount1	1999889	unsigned long
WaitCount2	50	unsigned long
WaitCount3	61	unsigned long
<Enter expression>		

PART 2 – QUESTION 1

Watch 2

Name	Value	Type
Button1RespTime	<cannot evaluate>	uchar
Button2RespTime	<cannot evaluate>	uchar
NumCreated	6	unsigned long
SignalCount1	20000	unsigned long
SignalCount2	20000	unsigned long
SignalCount3	1960000	unsigned long
WaitCount1	1983956	unsigned long
WaitCount2	11554	unsigned long
WaitCount3	4490	unsigned long
<Enter expression>		

Experiment 1

Watch 2

Name	Value	Type
Button1RespTime	<cannot evaluate>	uchar
Button2RespTime	<cannot evaluate>	uchar
NumCreated	6	unsigned long
SignalCount1	20000	unsigned long
SignalCount2	20000	unsigned long
SignalCount3	1960000	unsigned long
WaitCount1	1999889	unsigned long
WaitCount2	50	unsigned long
WaitCount3	61	unsigned long
<Enter expression>		

Experiment 2

Using cooperative spinlock semaphores in experiment 2 allowed for a dramatically decreased wait time for count 2 & 3 than that of the blocking semaphores in experiment 1. The total for the waits and the total for the signal in both exeriments should both be equal to 2 million.

PART 3 – EXPERIMENT 1

Watch 1

Name	Value	Type
DisplayCount	<cannot evaluate>	uchar
ConsumerCount	<cannot evaluate>	uchar
tcbs[1].ArriveTime	0	unsigned int
tcbs[1].WaitTime	0	unsigned int
tcbs[1].ExecCount	0	unsigned int
tcbs[3].ArriveTime	0	unsigned int
tcbs[3].WaitTime	0	unsigned int
tcbs[3].ExecCount	0	unsigned int
Count1	0x0790035F	unsigned long
Count2	0x00000000	unsigned long
Count3	0x00000000	unsigned long
tcbs[0].ArriveTime	0x00000000	unsigned int
tcbs[2].ArriveTime	0x00000000	unsigned int
tcbs[0].WaitTime	0x00000002	unsigned int
tcbs[2].WaitTime	0x00000000	unsigned int
tcbs[0].ExecCount	0x00005756	unsigned int
tcbs[2].ExecCount	0x00000000	unsigned int
<Enter expression>		

PART 3 – EXPERIMENT 2A

Watch 1

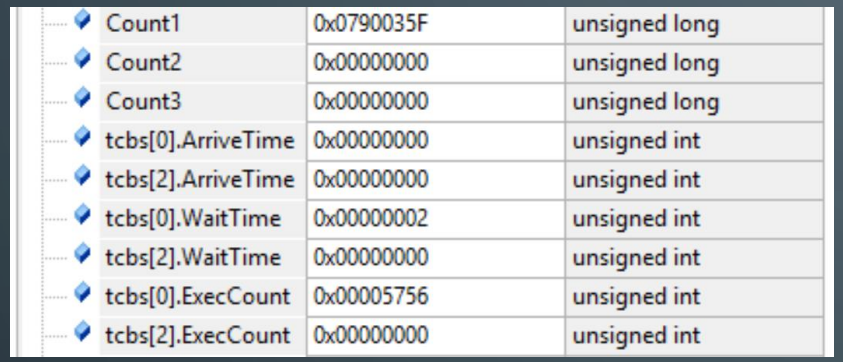
Name	Value	Type
DisplayCount	2049	unsigned long
ConsumerCount	600	unsigned long
tcbs[1].ArriveTime	2545	unsigned int
tcbs[1].WaitTime	0	unsigned int
tcbs[1].ExecCount	2401	unsigned int
tcbs[3].ArriveTime	2545	unsigned int
tcbs[3].WaitTime	2	unsigned int
tcbs[3].ExecCount	6747	unsigned int
Count1	<cannot evaluate>	uchar
Count2	<cannot evaluate>	uchar
Count3	<cannot evaluate>	uchar
tcbs[0].ArriveTime	0x000009F1	unsigned int
tcbs[2].ArriveTime	0x000009F1	unsigned int
tcbs[0].WaitTime	0x00000000	unsigned int
tcbs[2].WaitTime	0x00000000	unsigned int
tcbs[0].ExecCount	0x00000000	unsigned int
tcbs[2].ExecCount	0x00002E69	unsigned int
<Enter expression>		

PART 3 – EXPERIMENT 2B

Watch 2

Name	Value	Type
Button1RespTime	0	unsigned long
Button2RespTime	0	unsigned long
NumCreated	8	unsigned long
SignalCount1	<cannot evaluate>	uchar
SignalCount2	<cannot evaluate>	uchar
SignalCount3	<cannot evaluate>	uchar
WaitCount1	<cannot evaluate>	uchar
WaitCount2	<cannot evaluate>	uchar
WaitCount3	<cannot evaluate>	uchar
<Enter expression>		

PART 3 – QUESTION 1



Count1	0x0790035F	unsigned long
Count2	0x00000000	unsigned long
Count3	0x00000000	unsigned long
tcbs[0].ArriveTime	0x00000000	unsigned int
tcbs[2].ArriveTime	0x00000000	unsigned int
tcbs[0].WaitTime	0x00000002	unsigned int
tcbs[2].WaitTime	0x00000000	unsigned int
tcbs[0].ExecCount	0x00005756	unsigned int
tcbs[2].ExecCount	0x00000000	unsigned int

Experiment 1

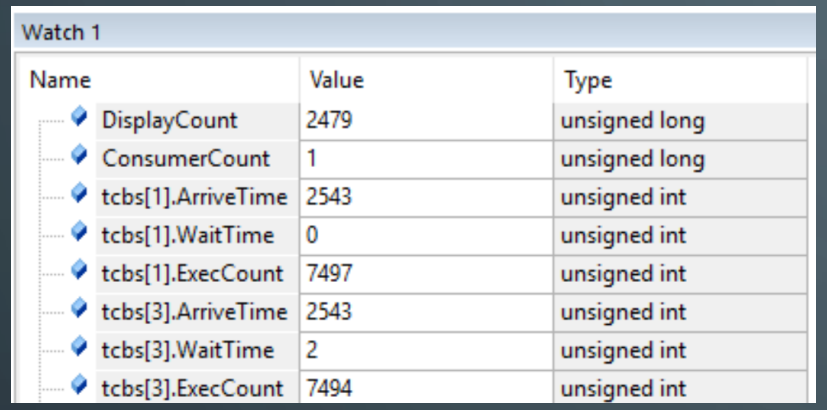
Watch 1

Name	Value	Type
CONTEXTSWI...	0	unsigned long
TIMESLICE	0	unsigned long
count1	0x00051C5A	unsigned long
count2	0x00050DF2	unsigned long
count3	0x0004FE9C	unsigned long
<Enter expression>		

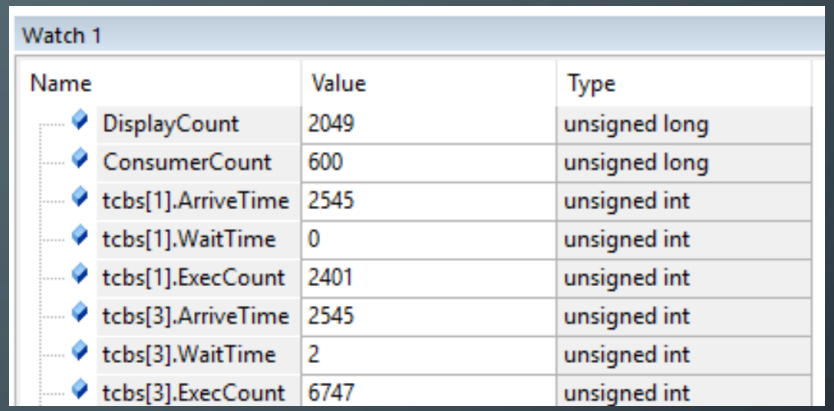
Mini Project 2

The count variable that we saw in mini project two ran very fairly in the fact that they were essentially equal at all times. Using the fixed priority scheduler you only are going to run the highest priority and in this case it is count 1. This is the reason for the zeros for count 2 and count 3.

PART 3 – QUESTION 2A



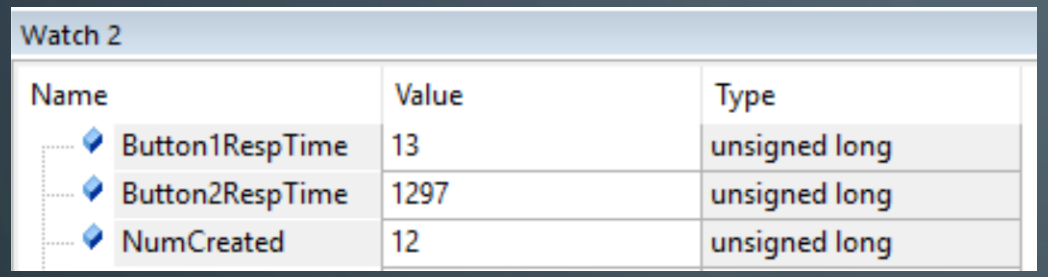
Experiment 1 Part 1



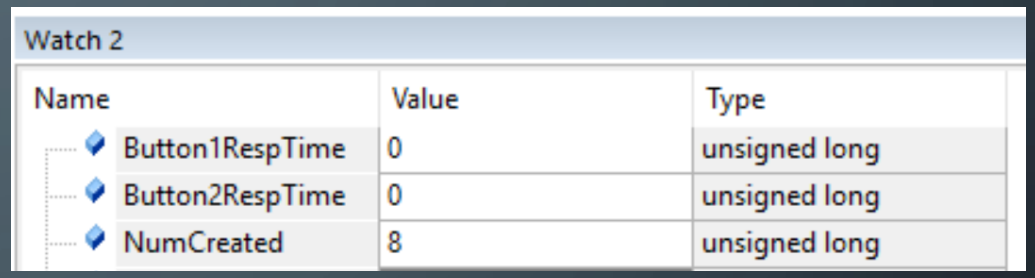
Experiment 2 Part 3

The difference is that consumer is priority one for part 3 and that means it runs a lot more. In part one we used blocking semaphores and this caused the consumer to only run once.

PART 3 – QUESTION 2B



Experiment 2 Part 1



Experiment 2 Part 3

Using the fixed priority scheduler the wait time was nothing for either button response. I believe they are actually running because when I hit the button threads are created. The difference being that the threads are not blocked only running in a priority based fashion.

PART 4 – EXPERIMENT 1

Watch 1

Name	Value	Type
DisplayCount	<cannot evaluate>	uchar
ConsumerCount	<cannot evaluate>	uchar
tcbs[1].ArriveTime	0	unsigned int
tcbs[1].WaitTime	4	unsigned int
tcbs[1].ExecCount	3444	unsigned int
tcbs[3].ArriveTime	0	unsigned int
tcbs[3].WaitTime	0	unsigned int
tcbs[3].ExecCount	0	unsigned int
Count1	19489236	unsigned long
Count2	19486700	unsigned long
Count3	9737416	unsigned long
tcbs[0].ArriveTime	0	unsigned int
tcbs[2].ArriveTime	0	unsigned int
tcbs[0].WaitTime	2	unsigned int
tcbs[2].WaitTime	10	unsigned int
tcbs[0].ExecCount	3444	unsigned int
tcbs[2].ExecCount	1721	unsigned int
<Enter expression>		

PART 4 – EXPERIMENT 2A

Watch 1

Name	Value	Type
DisplayCount	2055	unsigned long
ConsumerCount	600	unsigned long
tcbs[1].ArriveTime	2547	unsigned int
tcbs[1].WaitTime	0	unsigned int
tcbs[1].ExecCount	2401	unsigned int
tcbs[3].ArriveTime	2547	unsigned int
tcbs[3].WaitTime	2	unsigned int
tcbs[3].ExecCount	6765	unsigned int
Count1	<cannot evaluate>	uchar
Count2	<cannot evaluate>	uchar
Count3	<cannot evaluate>	uchar
tcbs[0].ArriveTime	2547	unsigned int
tcbs[2].ArriveTime	2547	unsigned int
tcbs[0].WaitTime	0	unsigned int
tcbs[2].WaitTime	0	unsigned int
tcbs[0].ExecCount	0	unsigned int
tcbs[2].ExecCount	11366	unsigned int
<Enter expression>		

PART 4 – EXPERIMENT 2B

Watch 2

Name	Value	Type
Button1RespTime	30	unsigned long
Button2RespTime	74	unsigned long
NumCreated	8	unsigned long
SignalCount1	<cannot evaluate>	uchar
SignalCount2	<cannot evaluate>	uchar
SignalCount3	<cannot evaluate>	uchar
WaitCount1	<cannot evaluate>	uchar
WaitCount2	<cannot evaluate>	uchar
WaitCount3	<cannot evaluate>	uchar
<Enter expression>		

PART 4 – QUESTION 1

Watch 1

Name	Value	Type
DisplayCount	<cannot evaluate>	uchar
ConsumerCount	<cannot evaluate>	uchar
tcbs[1].ArriveTime	0	unsigned int
tcbs[1].WaitTime	4	unsigned int
tcbs[1].ExecCount	3444	unsigned int
tcbs[3].ArriveTime	0	unsigned int
tcbs[3].WaitTime	0	unsigned int
tcbs[3].ExecCount	0	unsigned int
Count1	19489236	unsigned long
Count2	19486700	unsigned long
Count3	9737416	unsigned long
tcbs[0].ArriveTime	0	unsigned int
tcbs[2].ArriveTime	0	unsigned int
tcbs[0].WaitTime	2	unsigned int
tcbs[2].WaitTime	10	unsigned int
tcbs[0].ExecCount	3444	unsigned int
tcbs[2].ExecCount	1721	unsigned int
<Enter expression>		

Experiment 1 Part 4

Watch 1

Name	Value	Type
DisplayCount	<cannot evaluate>	uchar
ConsumerCount	<cannot evaluate>	uchar
tcbs[1].ArriveTime	0	unsigned int
tcbs[1].WaitTime	0	unsigned int
tcbs[1].ExecCount	0	unsigned int
tcbs[3].ArriveTime	0	unsigned int
tcbs[3].WaitTime	0	unsigned int
tcbs[3].ExecCount	0	unsigned int
Count1	0x0790035F	unsigned long
Count2	0x00000000	unsigned long
Count3	0x00000000	unsigned long
tcbs[0].ArriveTime	0x00000000	unsigned int
tcbs[2].ArriveTime	0x00000000	unsigned int
tcbs[0].WaitTime	0x00000002	unsigned int
tcbs[2].WaitTime	0x00000000	unsigned int
tcbs[0].ExecCount	0x00005756	unsigned int
tcbs[2].ExecCount	0x00000000	unsigned int
<Enter expression>		

Experiment 1 Part 3

I noticed that with the aging scheduler that even though priority 1 should be the only one running, aging the lower priority threads made them run eventually. This is why for part 4 we see that the counts are not zero like those found from part 3.

PART 4 – QUESTION 2A

Watch 1

Name	Value	Type
DisplayCount	2055	unsigned long
ConsumerCount	600	unsigned long
tcbs[1].ArriveTime	2547	unsigned int
tcbs[1].WaitTime	0	unsigned int
tcbs[1].ExecCount	2401	unsigned int
tcbs[3].ArriveTime	2547	unsigned int
tcbs[3].WaitTime	2	unsigned int
tcbs[3].ExecCount	6765	unsigned int

Experiment 2A Part 4

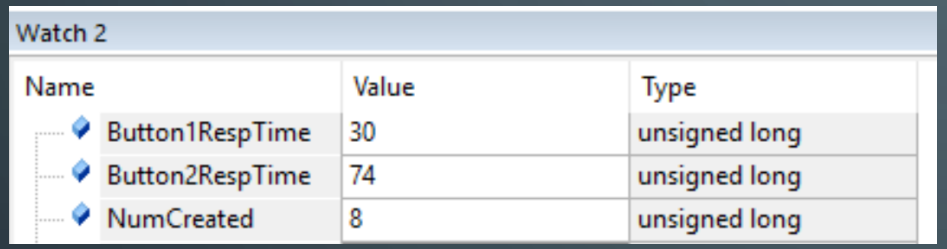
Watch 1

Name	Value	Type
DisplayCount	2463	unsigned long
ConsumerCount	20	unsigned long
tcbs[1].ArriveTime	2543	unsigned int
tcbs[1].WaitTime	0	unsigned int
tcbs[1].ExecCount	7498	unsigned int
tcbs[3].ArriveTime	2543	unsigned int
tcbs[3].WaitTime	2	unsigned int
tcbs[3].ExecCount	7496	unsigned int

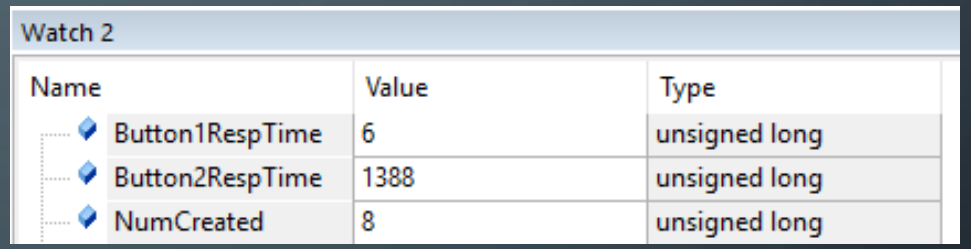
Experiment 2 Part 1

When using the aging scheduler the consumer runs a lot more often than that of the spinlock semaphores used in part 1.

PART 4 – QUESTION 2B



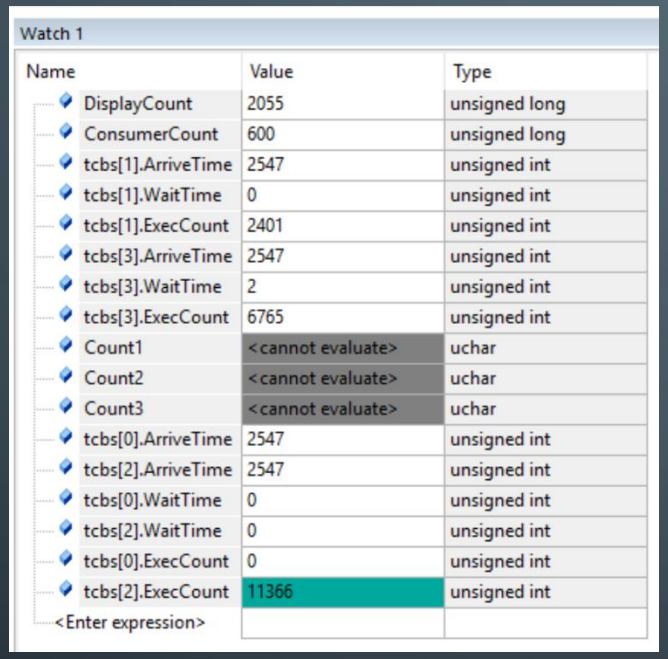
Experiment 2B Part 4



Experiment 1 Part 1

The response time for the buttons were faster using aging if you were to average the button response times. Part 1 response time for button1 was fast but the slow response from button 2 made this a slow process. Part 4 made the response time of button 1 slower but button 2 much faster.

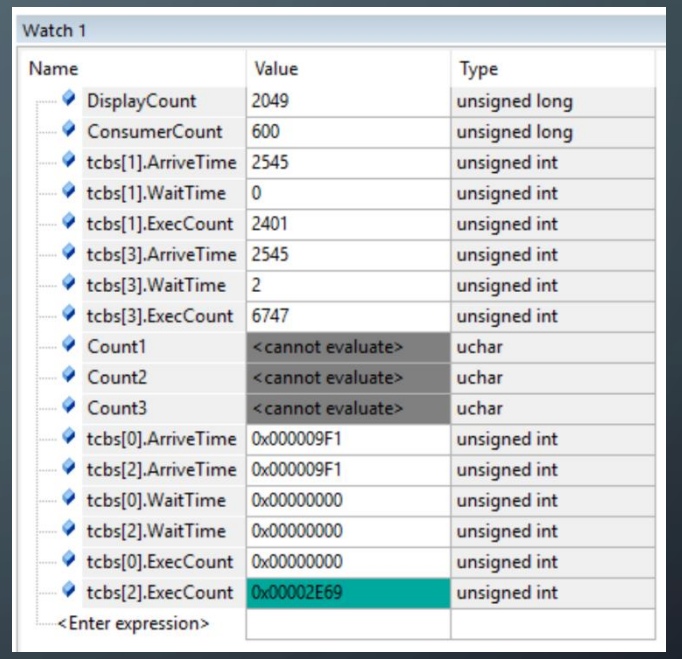
PART 4 – QUESTION 3A



Watch 1

Name	Value	Type
DisplayCount	2055	unsigned long
ConsumerCount	600	unsigned long
tcbs[1].ArriveTime	2547	unsigned int
tcbs[1].WaitTime	0	unsigned int
tcbs[1].ExecCount	2401	unsigned int
tcbs[3].ArriveTime	2547	unsigned int
tcbs[3].WaitTime	2	unsigned int
tcbs[3].ExecCount	6765	unsigned int
Count1	<cannot evaluate>	uchar
Count2	<cannot evaluate>	uchar
Count3	<cannot evaluate>	uchar
tcbs[0].ArriveTime	2547	unsigned int
tcbs[2].ArriveTime	2547	unsigned int
tcbs[0].WaitTime	0	unsigned int
tcbs[2].WaitTime	0	unsigned int
tcbs[0].ExecCount	0	unsigned int
tcbs[2].ExecCount	11366	unsigned int
<Enter expression>		

Experiment 2A Part 4



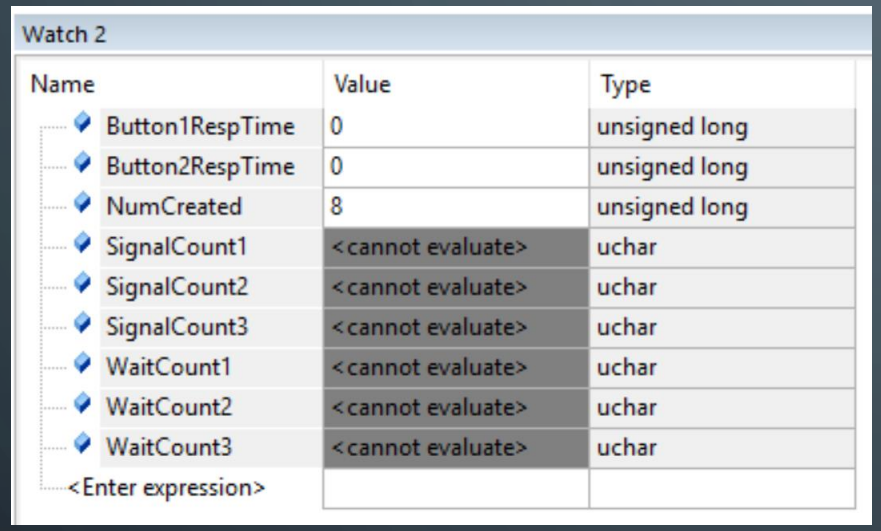
Watch 1

Name	Value	Type
DisplayCount	2049	unsigned long
ConsumerCount	600	unsigned long
tcbs[1].ArriveTime	2545	unsigned int
tcbs[1].WaitTime	0	unsigned int
tcbs[1].ExecCount	2401	unsigned int
tcbs[3].ArriveTime	2545	unsigned int
tcbs[3].WaitTime	2	unsigned int
tcbs[3].ExecCount	6747	unsigned int
Count1	<cannot evaluate>	uchar
Count2	<cannot evaluate>	uchar
Count3	<cannot evaluate>	uchar
tcbs[0].ArriveTime	0x000009F1	unsigned int
tcbs[2].ArriveTime	0x000009F1	unsigned int
tcbs[0].WaitTime	0x00000000	unsigned int
tcbs[2].WaitTime	0x00000000	unsigned int
tcbs[0].ExecCount	0x00000000	unsigned int
tcbs[2].ExecCount	0x00002E69	unsigned int
<Enter expression>		

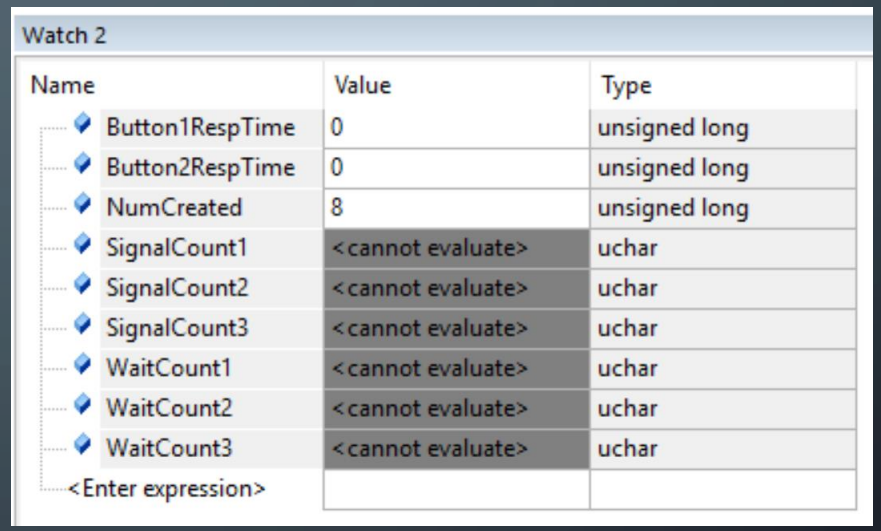
Experiment 2A Part 3

I did not notice any changes as far as the counts and wait times. This seems to be correct.

PART 4 – QUESTION 3B



Experiment 2A Part 4



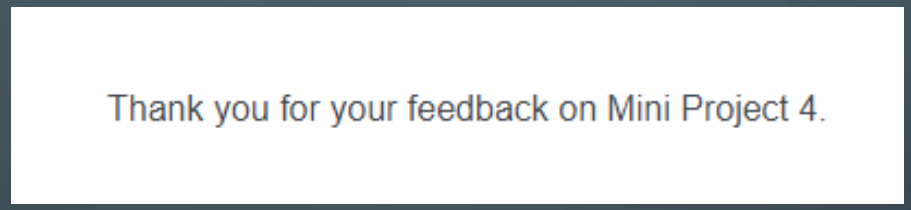
Experiment 2B Part 3

I did not notice any changes within the response times.

PART 4 – EXPERIMENT 3 DATA TABLE

JSFIFOSIZE	TIMESLICE	Spinlock / Round Robin	Blocking / Round Robin	Blocking / Dynamic Priority
8	2	582	0	0
32	2	568	0	0
64	2	349	0	0
64	1	421	0	0
64	10	536	0	0

SURVEY



Thank you for the extension, life got in the way and my procrastination caught up with me. Appreciate it.