**Lab Activity 16: IPC and Synchronisation II**

1. Brief description of what the producer/consumer problem is.

[Brief description here]

The producer/consumer problem, also known as bounded-buffer problem describes two processes, they both share a common fixed size buffer used as a queue. The producers job is to generate data, put it to into the buffer and then start the process again. The consumer at the same time is consuming data such as removing it from the buffer, one piece at a time. The problem is that the to make sure the producer does not add data into the buffer if the buffer is full and that the consumer does not try to delete or remove data from an empty buffer. Therefore, the producer only has two choices, either to go to sleep or discard data if it finds that the buffer is full, so when the consumer tries to remove an item from the buffer next time, the producer is notified about the activity and instantly start to fill the buffer again so that the buffer is not empty. Similarly, the consumer goes to sleep if the buffer is empty therefore when the producer adds data into the buffer, the consumer will be notified and starts to remove data from the buffer.

<https://en.wikipedia.org/wiki/Producer%E2%80%93consumer_problem>

<http://www.dcs.ed.ac.uk/home/adamd/essays/ex1.html>

<https://docs.oracle.com/cd/E19455-01/806-5257/sync-31/index.html>

1. Modified commented code that creates a ring/circular buffer, ensures data is not corrupted and works with different buffer sizes.

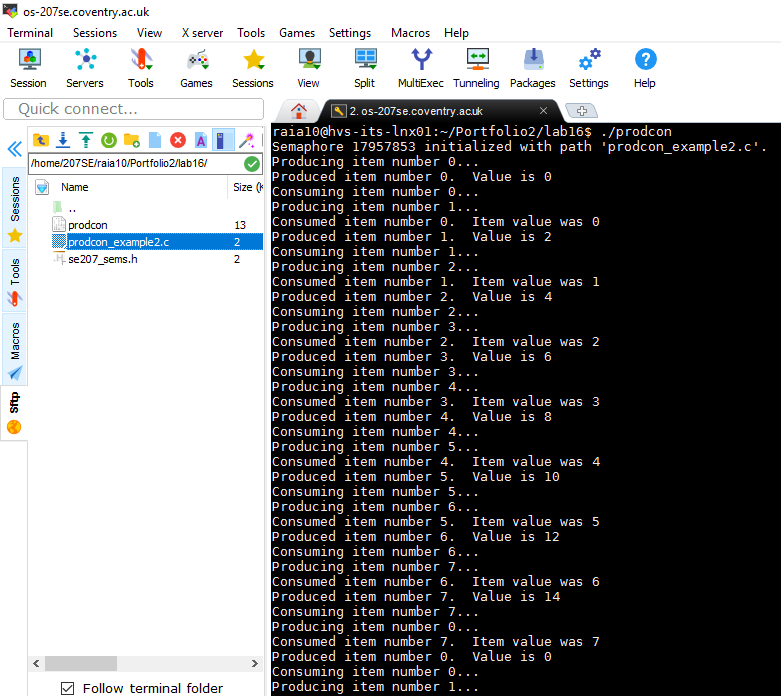
[Commented code here]

1. #include <sys/ipc.h>
2. #include <sys/sem.h>
3. #include <sys/shm.h>
4. #include <stdio.h>
5. #include <stdlib.h>
6. #include "se207\_sems.h"
8. /\* Remember to try reversing the timings...\*/
10. **int** bufferlength=8; //Limited buffer length
11. //what could we do about this?
13. **int** main(**int** argc, **char** argv[]){
15. //Create shared memory segment
16. **int** shm\_id=shmget(ftok("prodcon\_example2.c",2),bufferlength,
17. 0666|IPC\_CREAT);
19. **int** consumerpos = bufferlength + 1;
20. **int** producerpos = bufferlength + 2;
22. //Use our source file as the "key"
23. **int** id=se207\_semget("prodcon\_example2.c",0);
25. **char**\* data; //For our pointer to shared memory...
27. **int** pid=fork();
28. **if**(pid){
29. //P1 - CONSUMER
30. shm\_id=shmget(ftok("prodcon\_example2.c",2),0,006);
32. //Attach the shared buffer
33. data = shmat(shm\_id, (**void** \*)0, 0);
35. data[consumerpos] = 0;
36. data[producerpos] = 0;
38. **while**(1){
39. se207\_wait(id);
40. **while** (data[consumerpos] == data[producerpos]);
41. printf("Consuming item number %d...\n",data[consumerpos]);
42. sleep(1);
44. **char** item=data[consumerpos];
46. printf("Consumed item number %d.  Item value was %d\n",
47. data[consumerpos],item);
48. data[consumerpos] =(data[consumerpos] + 1) % bufferlength;
49. }
51. //Detatch
52. shmdt(data);
53. printf("All done consuming.\n");
55. wait(); //For child process so that we can
57. //Delete the shared memory
58. printf("Child ended, removing shm\n");
59. shmctl(shm\_id, IPC\_RMID, NULL);
60. }**else**{
61. //P2
62. shm\_id=shmget(ftok("prodcon\_example2.c",2),0,006);
63. //Attach the shared buffer
64. data = shmat(shm\_id, (**void** \*)0, 0);
66. data[consumerpos] = 0;
67. data[producerpos] = 0;
69. **while**(1){ //changed to true so loop runs forever
70. **while** (((data[producerpos] + 1)%bufferlength) == data[consumerpos]);
71. printf("Producing item number %d...\n",data[producerpos]);
72. sleep(2);
73. data[data[producerpos]]=data[producerpos]\*2; //Simple data, easy to check.
74. printf("Produced item number %d.  Value is %d\n",
75. data[producerpos],data[data[producerpos]]);
76. data[producerpos] = (data[producerpos] + 1)%bufferlength;
77. se207\_signal(id);
78. }
79. //Detatch
80. shmdt(data);
81. printf("Producer finished.");
82. }
83. }

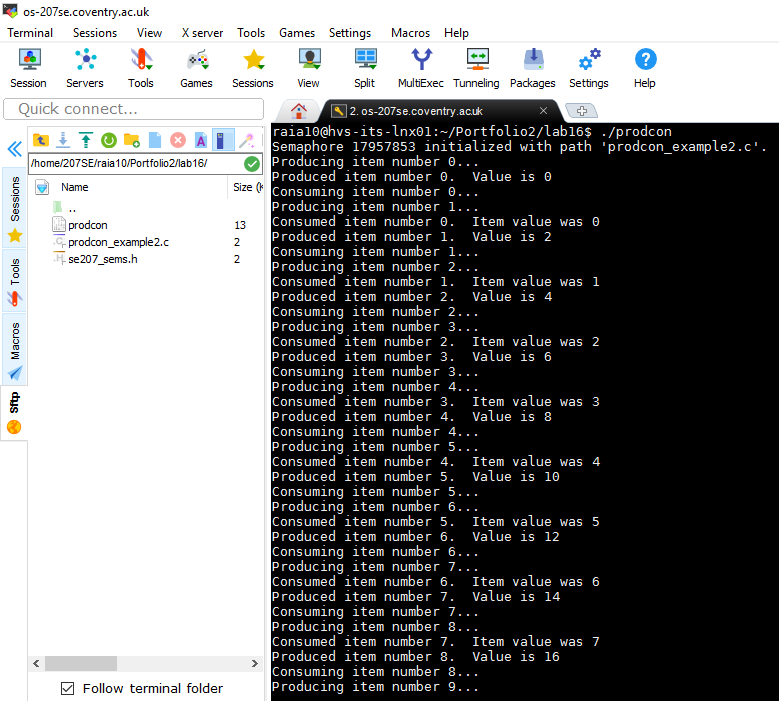
* Ring/Circular buffer with different buffer sizes

[Screenshot(s) showing code working as a circular buffer, with different buffer sizes]

**BUFFE SIZE = 8**



**BUFFER SIZE = 10**

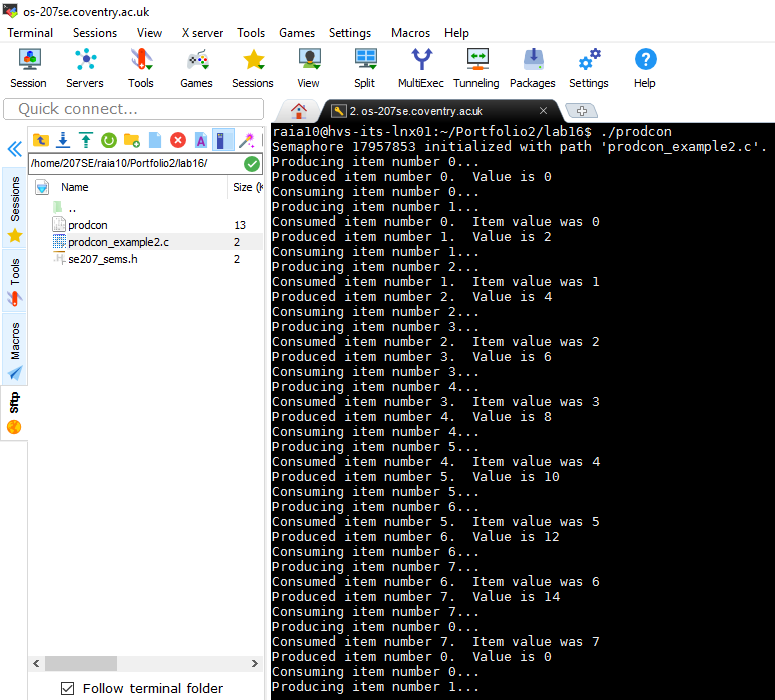


* Producer/Consumer working at different speeds

[Screenshot(s) showing the producer and consumer working at different speeds]

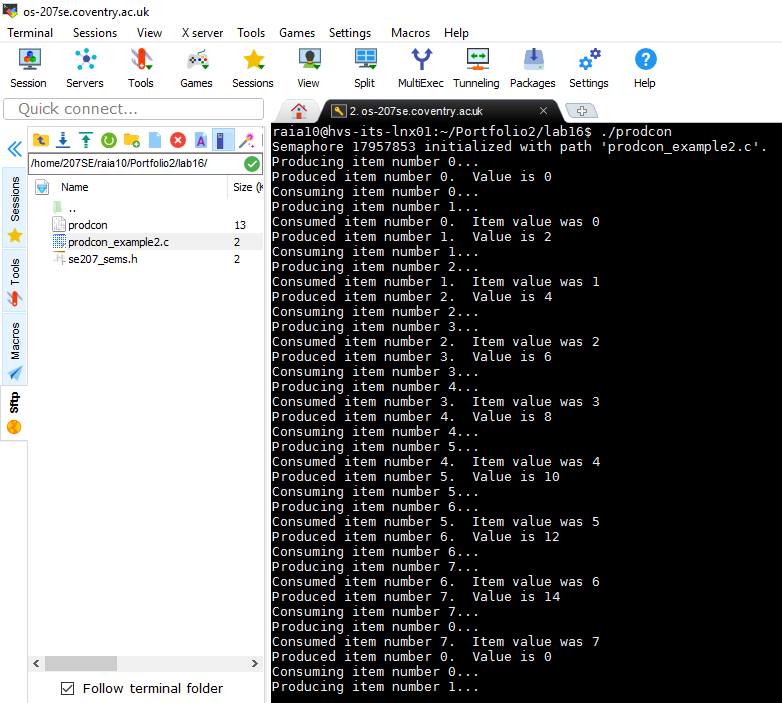
**CONSUMER SPEED – 1**

**PRODUCER SPEED – 2**



**CONSUMER SPEED – 1**

**PRODUCER SPEED – 3**



Evidence

Include the description of the producer/consumer problem. The Commented code to complete the task. Output from the working program.