Deadlock Report

Instruction:

Take some code that you have already written, for this class or some other class, and modify it so that it can deadlock.

Submit a brief report with your code and evidence that it actually does deadlock.

Explanation:

I used the previous assignment's code (concurrency beyond locks) in order to complete this assignment. The modification I made is on condition for putChar while loop. I changed it so that it will go into while-loop, which will cause to wait() while the other is waiting as well. The code satisfies the Conditions for Deadlock, which are listed below with an explanation.

Condition 1: Circular Wait

In the code, two conditional variables work as a lock. One we do wait() in the while-loop, and the other will do the same. As a result, both of them will have the ability to release the lock; however, both of them are in wait(). Thus, it is a circular wait(). One is waiting for the other.

Condition 2: Hold and Wait  
 In the code, there are two conditional variables, and one variable (lock with conditional variable) is holding the resources required for the other process to proceed. For example, charAdded holds the recourses that enable to release charRemoved's processes.

Condition 3: No Preemption

There is no priority in executing any of the code in the code and cannot forcibly remove or add any type of processes threads that are holding them to wait.

Condition 4: Mutual Exclusion

In the code, two conditional variables work as a lock. Each variable has control over which part, which method to run since if one is running, it will run a piece of code that the other conditional variable has no control. Thus, each lock (conditional variable) has exclusive control of resources that are required.

Result:

In the code, I have added four print statements in order to see where visually what is taking place in the code. I added print statements right after entering into while-loop in putChar() and getChar(). Other two print statements are right after the while-loops in putChar() and getChar(). As figure 1 (placed below) shows, only two print statements that I added were displayed. Both of them are that was inside of while-loop, meaning it never broke neither while-loop. It shows that both locks were never unlocked and stayed at the wait, and it will be at wait forever since no one will notify (unlock). On the other hand, as you can see in Figure 2 (placed below), it repeatedly enters to while-loop, goes outside of the while-loop, and retrieves the character. This example is when it is not deadlocked.

Code:

deadlock.py (Modified):

1. #!/usr/bin/env python3
2. # pump.py Nigel Ward, 2021. Based on John Osterhout's Producer/Consumer code.
3. import sys, threading, time
4. from threading import Condition
5. global count, putIndex, getIndex, cbuffer, bufLock
6. def pumpProducer():
7. print('starting Producer')
8. arpaciQuote ='" Yeats famously said "Education is not the filling of a pail but the lighting of a fire." He was right but wrong at the same time. You do have to "fill the pail" a bit, and these notes are certainly here to help with that part of your education; after all, when you go to interview at Google, and they ask you a trick question about how to use semaphores, it might be good to actually know what a semaphore is, right? But Yeats's larger point is obviously on the mark: the real point of education is to get you interested in something, to learn something more about the subject matter on your own and not just what you have to digest to get a good grade in some class. As one of our fathers (Remzi's dad, Vedat Arpaci) used to say, "Learn beyond the classroom".'
9. for charToSend in arpaciQuote:
10. putChar(charToSend)
11. putChar(0) # the solution!!
13. def putChar(character):
14. global count, putIndex, bufLock, charRemoved, charAdded
15. bufLock.acquire()
16. while count == 0:
17. print('removed')
18. charRemoved.wait()
19. print('out of removed')
20. count += 1
21. cbuffer[putIndex] = character
22. putIndex += 1
23. if putIndex == bufsize:
24. putIndex = 0
25. charAdded.notify()
26. bufLock.release()
27. def pumpConsumer():
28. print('starting Consumer')
29. while (1):
30. for i in range(100):
31. c = getChar()
32. if c == 0:
33. return
34. print(c,end="")
35. print("") # newline
36. def getChar():
37. global count, getIndex, bufLock, charRemoved, charAdded
38. bufLock.acquire()
39. while (count == 0):
40. print('added')
41. charAdded.wait()
42. print('out of added')
43. count -= 1
44. c = cbuffer[getIndex]
45. getIndex += 1
46. if (getIndex == bufsize):
47. getIndex = 0
48. charRemoved.notify()
49. bufLock.release()
50. return c
51. ### main ###
52. if len(sys.argv) != 2:
53. print(len(sys.argv))
54. print("usage: pump bufferSize")
55. exit(1)
56. bufsize = int(sys.argv[1])
57. cbuffer = ['x'] \* bufsize # circular buffer; x means uninitialized
58. count = putIndex = getIndex = 0
59. bufLock = threading.Lock()
60. charAdded = Condition(bufLock)
61. charRemoved = Condition(bufLock)
62. consumer = threading.Thread(target=pumpConsumer)
63. consumer.start()
64. producer = threading.Thread(target=pumpProducer)
65. producer.start()
66. producer.join()

pump-fixed.py (original):

1. #!/usr/bin/env python3
2. # pump.py Nigel Ward, 2021. Based on John Osterhout's Producer/Consumer code.
3. import sys, threading, time
4. from threading import Condition
5. global count, putIndex, getIndex, cbuffer, bufLock
6. def pumpProducer():
7. print('starting Producer')
8. arpaciQuote ='" Yeats famously said "Education is not the filling of a pail but the lighting of a fire." He was right but wrong at the same time. You do have to "fill the pail" a bit, and these notes are certainly here to help with that part of your education; after all, when you go to interview at Google, and they ask you a trick question about how to use semaphores, it might be good to actually know what a semaphore is, right? But Yeats's larger point is obviously on the mark: the real point of education is to get you interested in something, to learn something more about the subject matter on your own and not just what you have to digest to get a good grade in some class. As one of our fathers (Remzi's dad, Vedat Arpaci) used to say, "Learn beyond the classroom".'
9. for charToSend in arpaciQuote:
10. putChar(charToSend)
11. putChar(0) # the solution!!
13. def putChar(character):
14. global count, putIndex, bufLock, charRemoved, charAdded
15. bufLock.acquire()
16. while count >= bufsize:
17. charRemoved.wait()
18. count += 1
19. cbuffer[putIndex] = character
20. putIndex += 1
21. if putIndex == bufsize:
22. putIndex = 0
23. charAdded.notify()
24. bufLock.release()
25. def pumpConsumer():
26. print('starting Consumer')
27. while (1):
28. for i in range(100):
29. c = getChar()
30. if c == 0:
31. return
32. print(c,end="")
33. print("") # newline
34. def getChar():
35. global count, getIndex, bufLock, charRemoved, charAdded
36. bufLock.acquire()
37. while (count == 0):
38. charAdded.wait()
39. count -= 1
40. c = cbuffer[getIndex]
41. getIndex += 1
42. if (getIndex == bufsize):
43. getIndex = 0
44. charRemoved.notify()
45. bufLock.release()
46. return c
47. ### main ###
48. if len(sys.argv) != 2:
49. print(len(sys.argv))
50. print("usage: pump bufferSize")
51. exit(1)
52. bufsize = int(sys.argv[1])
53. cbuffer = ['x'] \* bufsize # circular buffer; x means uninitialized
54. count = putIndex = getIndex = 0
55. bufLock = threading.Lock()
56. charAdded = Condition(bufLock)
57. charRemoved = Condition(bufLock)
58. consumer = threading.Thread(target=pumpConsumer)
59. consumer.start()
60. producer = threading.Thread(target=pumpProducer)
61. producer.start()
62. producer.join()

Output:

Text

Description automatically generated   
 Figure 1:Code output Example

Command: time python3 pump-fixed.py 10

Text

Description automatically generated

Text

Description automatically generated with medium confidence

Figure 2: A piece of output running pump-fixed.py with four additional print statement included in the deadlock.py