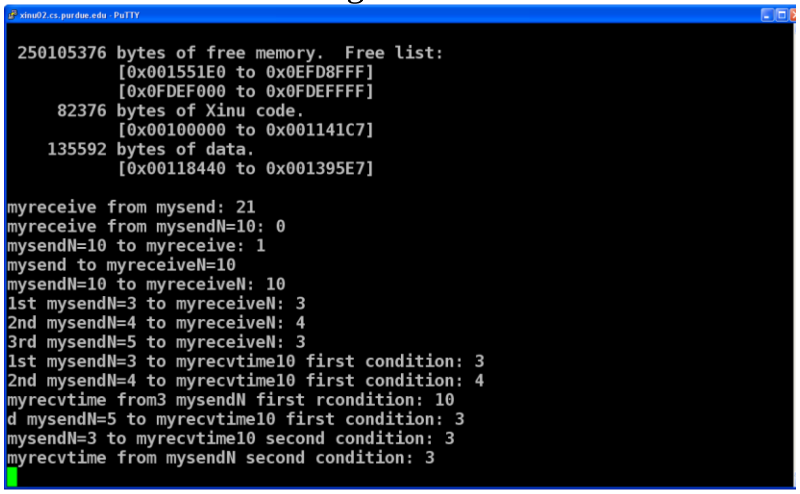
Qiaomu Yao CS 503 Lab 3

- The details behind your implementation. As part of this discussion write answers to the following questions:
    - How does your solution guarantee messages are received in the order in which they were sent?

      I used a circular linked list of size 20 for storing the messages. When I push the message on the linked list top, and pop the message from the bottom of the linked list. Just like a queue.

    - Describe your test cases. How to they ensure that the system calls correctly meet the requirements?

      Here is the Final running results below:



```
 250105376 bytes of free memory.  Free list:
           [0x001551E0 to 0x0EFD8FFF]
           [0x0FDEF000 to 0x0FDEFFFF]
     82376 bytes of Xinu code.
           [0x00100000 to 0x001141C7]
    135592 bytes of data.
           [0x00118440 to 0x001395E7]

myreceive from mysend: 21
myreceive from mysendN=10: 0
mysendN=10 to myreceive: 1
mysend to myreceiveN=10
mysendN=10 to myreceiveN: 10
1st mysendN=3 to myreceiveN: 3
2nd mysendN=4 to myreceiveN: 4
3rd mysendN=5 to myreceiveN: 3
1st mysendN=3 to myrecvtime10 first condition: 3
2nd mysendN=4 to myrecvtime10 first condition: 4
myrecvtime from3 mysendN first rcondition: 10
d mysendN=5 to myrecvtime10 first condition: 3
mysendN=3 to myrecvtime10 second condition: 3
myrecvtime from mysendN second condition: 3
```

There are several testing cases:

1. Send and receive with both single message and the message is 21;
2. Sendn 10 messages starting from 0 to 9, and receive single message, and return the message received which is 0, and then return the number of messages sendn has sent which is 1.
3. Ready to receiveN 10 messages, but only send one message.
4. Ready to receiveN 10 messages, and sendN 10 messages, return the number of messages of sendN which is 10.
5. Ready to receiveN 10 messages, and start to sendN 3 , sendN 4, and sendN 5, return the number of messages been sent of sendN which is 3, 4, 3. Since the first 2 send instruction don't exceed the limit so they can send all of their messages, but the last one cannot

send all of the 5 messages and instead send 3 which make the total number of messages satisfy the receiveN requirement.

6. There are two satisfaction condition for myrecvtime function: first is receiving enough messages and second is timeout. For the first condition I sendN 3 sendN 4 and sendN 5 continuously and myrecvtime function which can have 10 messages and lasts for 1000ms return the total number of messages it received which is 10; (the order of sendN 5 is below return value of myrecvtime function)
7. For the second time out condition, I only sendN 3 which obviously will time out for myrecvtime function with 10 messages limit and 1000ms time delay. It returns the number of messages the myrecvtime function received.

- What modifications would need to be made to allow for a truly unlimited number of messages to be sent to a target process? Are these modifications practical?

  It is better to use a queue to store the messages if considering have unlimited number of messages. I used Circular linked list which has a finite size and it will constrains the number of messages, thus queue is a better data structure for this. However, even though the receive function can receive unlimited number of messages, it still needs to return to the process, restore the interrupt, and deal with all the messages. Then call the receive function again to receive more messages.

  Also, if want to send unlimited messages, you have to have unlimited number of receive instructions. Receiving and sending are working together. Even though you sometimes can have more sending instruction than receiving instructions, receiving instructions always come ahead of sending. Only if we call unlimited receive or large enough receive that we are able to send unlimited messages.

  This could be possible by adding a while loop in my program and allocate more memory and changing the data structure from circular linked list to queue. But I don’t think it’s practical since when have limited memory and storage for messages and it’s better to have some restrictions to get rid of the overflow of memory.