Han Wang (Wang2786)
0028451697
Lab5

Part1:

I first created a new C function call sendblk() which behave like a regular send() but implemented with a queue. And set the process that has message to transfer to PR_SNDBLK and enqueue into a queue call sendqueue.

And then I modified regular reveive.c to work with sendblk() function, which is able to empty the message that process that is holding in prmsg, then transfer more msg from send queue to the prmsg buffer and replenish it.

Other small adjustment includes the size of queuetable and prototype and some new variables in process.h.

I wrote a few test to test the fundamental functionalities. And here is the output:

```
(Wang, Han)

wang2786

***Test 1***
Sender(PID:4) is sending message('A') to Reciever(pid:3)
(PID:3):Message recieved: A

***Test 2***
Sender(PID:6) is sending message('B') to Reciever(pid:5)
Se
Making a queue for msg: C

Enqueuing msg(C)
nder(PID:7) is sending message('C') to Reciever(pid:5)
Se
Enqueuing msg(D)
nder(PID:8) is sending message('D') to Reciever(pid:5)
Se
Enqueuing msg(E)
nder(PID:9) is sending message('E') to Reciever(pid:5)
(PID:5):Message recieved: B
(PID:5):Message recieved: C
(PID:5):Message recieved: D
(PID:5):Message recieved: E
```

As you can see, when different processes try to send messages to the same receiver, the receive process will pile up the message into a queue, and FIFO when its receivers turn to print the message.

Sender program: messageSend.c
Receive program: messageRec.c
Testing program: blockTest1.c blockTest2.c

Part2:
Describe your design that allows XINU to implement asynchronous IPC with callback function in Lab5Answers.pdf.
Ans: Since we want to asynchronously handle the inter process communication, I modified sleep() and clkdisp to call a new function I made call jump.c which checks whether the current pid is holding a message in buffer.

If that is the case, I execute the function using function pointer, so that it finds its way back to the original caller! Yay!

Here are some of the reasons why I made these design decisions. Since clkdisp() is called every quantum, it make sense to put the jump function there so that the system can constantly checking for whether current process is waiting and “CAN” receive a message. Another case is process has just woken up from sleep, we need to check whether it has a callback function so that we can run the cb.

Output

```
***PART2: Test 1 sending from multiple process
Sender(PID:11) is sending message('Z') to Reciever(pid:10)
Sender(PID:12) is sending message('Y') to Reciever(pid:10)

Making a queue for msg: Y

Enqueuing msg(Y)
Sender(PID:13) is sending message('P') to Reciever(pid:10)

Enqueuing msg(P)
mbuf = Z
mbuf = Y
mbuf = P

***Waiting PART2: Test 1 to Finish***

***PART2: Test 2 sending from same process with multiple message using send
blk()***
Sender(PID:15) is sending message('1') to Reciever(pid:14)
Sender(PID:15) is sending message('2') to Reciever(pid:14)

Making a queue for msg: 2

Enqueuing msg(2)
Sender(PID:15) is sending message('3') to Reciever(pid:14)

Enqueuing msg(3)
mbuf = 1
mbuf = 3
mbuf = 3

***Waiting PART2: Test 2 to Finish***


You reached the end of the Main().
```

Discuss why your design is compatible with kernels that support isolation/protection even though it is implemented in XINU.

Ans: Since my design is checking if the receiving process is asynchronous and has a callback function or not(and the code is outside of interrupt → meaning not in kernel mode), it limits the all receiver of different getting the message. Hence support the isolation between kernel and user mode. I am only call the jump to call back function iff process was sleep or depleted its time share, it prevents those are not from utilizing the call back jump. In short, all callback function are positioned after “restore(mask)” to ensure only do it in usermode.

Sender: asyMessageSend.c and asyMessageSendMult.c
Receiver: asyMessageRec.c
Test: asyTest1.c and asyTest2.c
Jump: jumpcb.c

Newfile: cbreg.c and mrecv_cb.c
Modification: process.h sleep.c clkdisp.S and so on

Bonus:
Added 2 man pages for receive() and sendblk() in receive.2 and sendblk.2, in proper format.