Han Wang (Wang2786)
0028451697
Lab5

For debugging purposes, I have moved the original print statement from receiver to receive.c, which will sometime print out other empty messages that communicated by idle processes. But the mechanic does not change, you can still determine where the message coming from and going to base on the (PID).

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 used cbreg() for checking whether the current has a call back function or not. I made changes to my sleep.c and clkhandler.c to handle the cases of “waking up from sleeping” and “consume all QUANTUM” status of a process. And if the function is not null and there is a message after context switch, I make a jump to the callback function, in my case, I worte the jumpcb function in assembly based on clkdisp.S without the register pushing.
I made two test cases, one is multiple process sending message and test 2 is single process sending multiple messages. They all make use sendblk() function I implemented in the first part as well as regular send().
Output

```
***PART2: Test 1 sending from multiple process
Sender(PID:11) is sending message('Z') to Reciever(pid:10)
(PID:2):Message recieved:

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)
(PID:10):Message recieved: Z
mbuf is 'Z'
(PID:10):Message recieved: Y
mbuf is 'Y'
(PID:10):Message recieved: P
mbuf is 'P'
(PID:2):Message recieved:


***Waiting PART2: Test 1 to Finish***
(PID:2):Message recieved:


***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)
(PID:14):Message recieved: 1
Sender(PID:15) is sending message('3') to Reciever(pid:14)

Enqueuing msg(3)
mbuf is '1'
(PID:14):Message recieved: 2
mbuf is '2'
(PID:14):Message recieved: 3
mbuf is '3'
(PID:2):Message recieved:

***Waiting PART2: Test 2 to Finish***
(PID:2):Message recieved:


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, 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.

Bonus:

Added 2 man pages for receive() and sendblk() in receive.2 and sendblk.2.