Lab 4: Timed Blocking Send and Asynchronous IPC with Callback Function

Changes to XINU

Cloned XINU's queue implementation for the implementation of timed blocking message send

Added these to the process table

Umsg32 sndmsg /* holds the 1 word message */
Bool8 sndflag /* if sndmsg is valid, then flag is true */
Qid16 pq /* blocked message queue id for the process */
Bool8 isSleeping /* flag to know whether or not process is sleeping */
Bool8 isAsynchronous /* flag to determine if process is asynchronous message receive */
Void *cbPointer /* pointer to the callback function */
Bool8 hasCBFunction /* flag to determine if there is a callback function */

Added msgglob to process

Defined PR_SEND constant

Added the clktimefine implementation

Modified recvtime.c and recvclr.c

1)

To start off, I basically cloned XINU's queue implementation as to separate XINU's queue from my implementation of blocked message sending's queue(s). From then on, each process created will have a process table entry called "pq" which holds the id of the queue of blocked messages for that specific process.

In sendbt() when I handle the blocking of messages, I will check if the receiver has any waiting messages. If so, then I know that I need to add the sending process to the receiver's queue of messages that are waiting to be received.

Before I add the message to the receiver's pq, I set the sender's sndmsg process table entry to the contents of the message passed into the sendbt() function and I set the sndflag to true to indicate that there is a valid message.

I then need to determine whether or not I need to insert the process into sleepq and the receiver's pq or just the receiver's pq. If in addition to enqueuing the process in the receiver's pq, I also have to insert the process into the sleepq because the maxwait is greater than 0, I set a flag in the process table of the sender process to indicate that the sender is currently in the sleepq (isSleeping).

I set the state of the sending process to PR_SEND to indicate that the process is in the waiting state and then call resched() to context switch the current (sending) process out. Right after that, I check again whether or not the receiver has any messages. If it doesn't then that means

that receive() has successfully reset its buffer. If it does have a message, then that means that the sending process has been woken up by sleep because its maxwait time is up. The message gets dropped and we just remove the sending process from the receiver's pq. Then lastly, sendbt() will just set the receiver's prmsg to the sending process's msg content and set prhasmsg to true, check the receiver's state, and ready the receiving process.

Every time receive() is called, I check if the receiver's pq is empty. If not, that means that there are sending processes that are waiting to be received so I unblock one message in FIFO order. I check if it is also in the sleepq (if maxwait initially was greater than 0) and set the isSleeping flag to false and then remove it from the queue. I then ready the unblocked sending process and receive() returns whatever message was stored in the receiver's process table's prmsg entry.

I also made the appropriate changes in recvtime.c and recvclr.c

2)

I did not use the TS scheduler.

First off, I added three new entries in the process table.

Bool8 isAsynchronous /* flag to determine if process is asynchronous message receive */

Void *cbPointer /* pointer to the callback function */

Bool8 hasCBFunction /* flag to determine if there is a callback function */

When I first run main(), I call registercb() (same as lab handout) which will set the hasCBFunction flag to true and also set the cbPointer to the address of myrecvhandler which is the callback function. When send is called, I set the isAsynchronous flag to true and then send makes the process ready. Whenever context switching is happening, I check if the receiver process isAsynchronous and if the receiver process hasCBFunction. If so, I will run the callback function which will call receive and receive the message.

Bonus)

Added the man page