# **Lab 6**

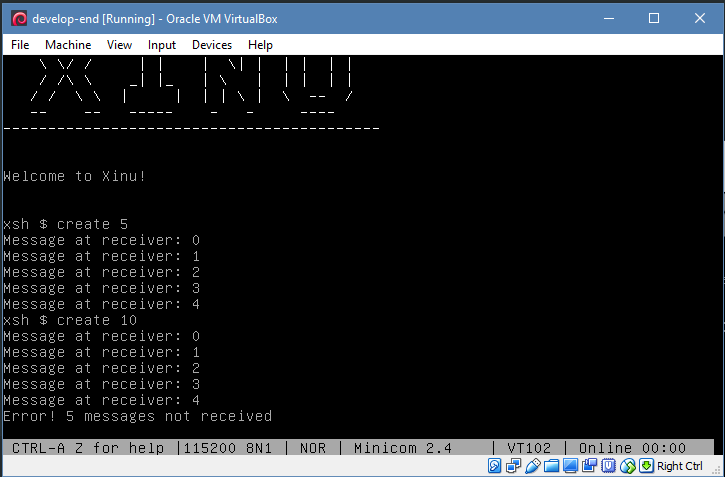
To create a system that can record up to K messages per process, we implemented the port method as described in the textbook. Basically, there is a port data structure setup in the file ports.h that allows for a message buffer, but also checks to make sure there are free ports available. We didn’t use the send or receive function, instead opting to use the built-in functions that deal with the port structure (specifically ptinit, ptsend, ptrecv, and ptcreate). These functions allow for the creation and initialization of a given number of ports, as well as access to the global message buffer which used to send the messages.

The process flow was to take an argument from the user to define the number of messages to send. We default the number of ports to 5 (as defined in our single shell function xsh\_create). Then, to get the messaging system set up, we first initialize all the ports with ptinit. This sets **all** the ports in Xinu (as defined by the global constant NPORTS which we did not edit) to be in the state PT\_FREE. Then, we created the same number of ports using ptcreate. Now, for all the ports to be set to PT\_ALLOC, which allows for them to actually receive a message.

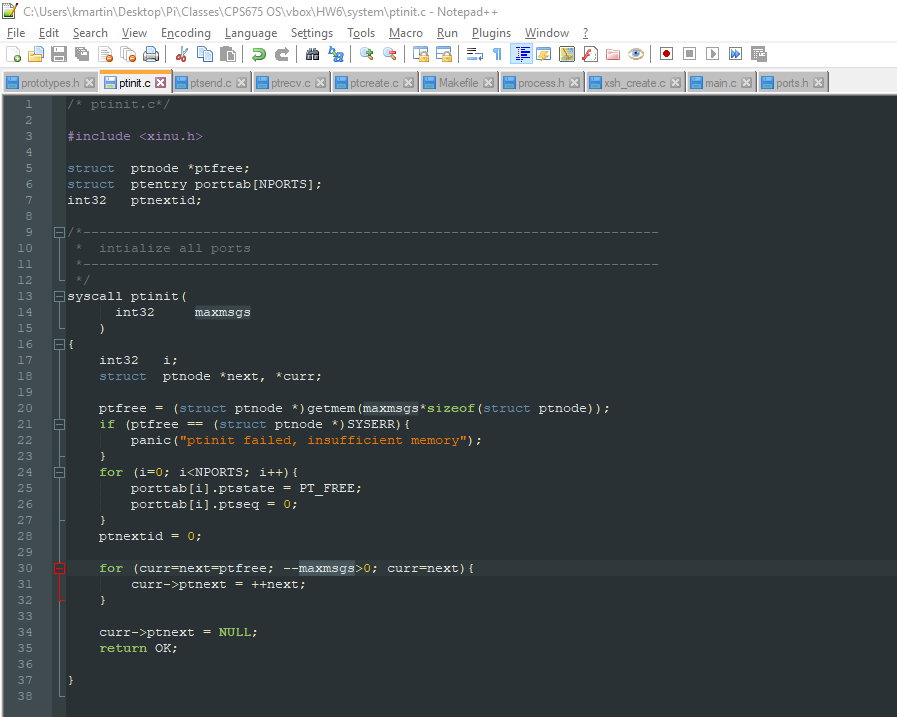
At this point, we needed to create a process to receive the message. At its core, all the process does is call ptrecv on the port we send the messages to (we just the first port, port 0). Next we added a loop to call ptrecv for as many messages as we sent. Finally, we built in checks to ensure that the number of messages would not be greater than the number of ports available.

To send the messages, we used the system call ptsend. Each call of ptsend sends one message, so we set a loop for the total messages as defined previously by the user. The loop sends the messages, and the receiver function gathers them and prints them out. In our case, we just the counter integers as our “messages”.

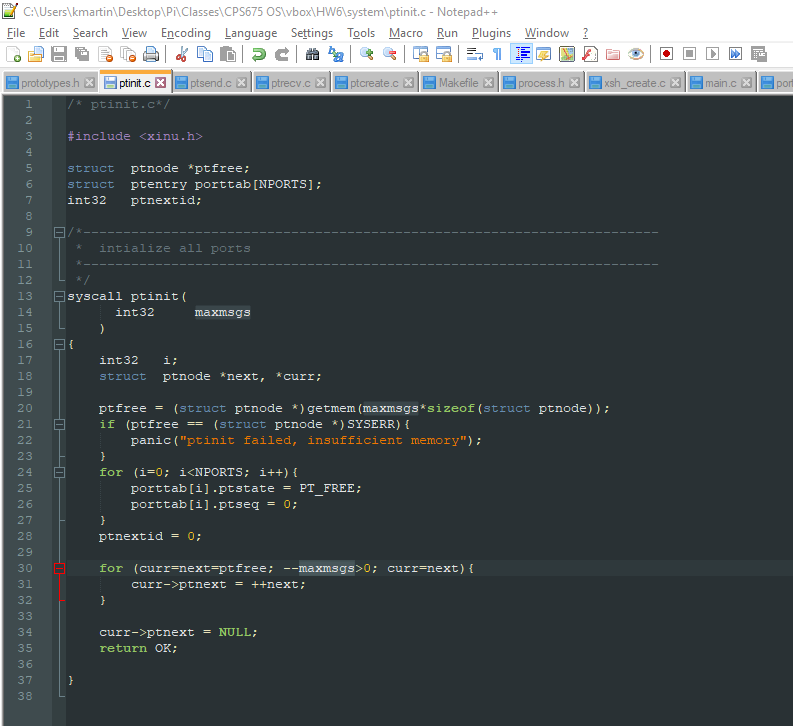
Below the screenshot shows first the user entering “create 5” which begins the process and sends 5 messages. Because the number of ports is equal to 5, all 5 messages are sent without error. Next the user enters “create 10” which causes 10 messages to be sent. This time, only 5 are received, and an error calculating the number of undelivered messages is displayed:



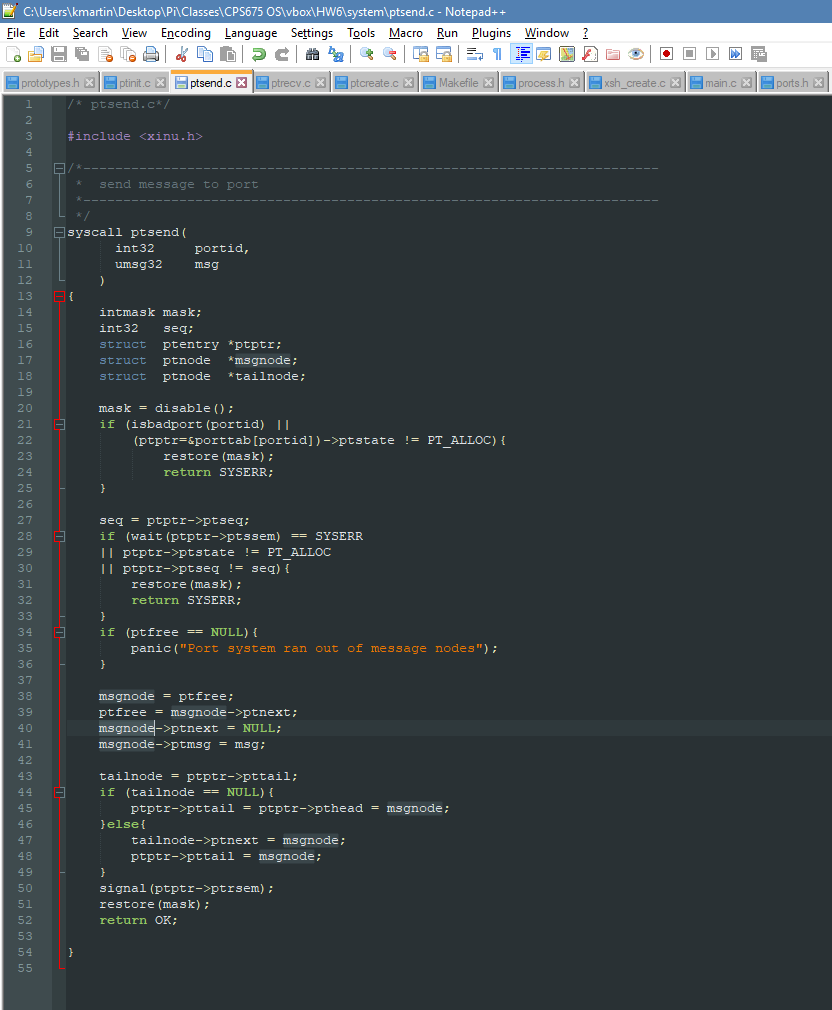
For the relevant screenshots, first the port related functions. These are directly from Xinu, but were not present in our Xinu folders, so we manually added:  
  
ptinit.c



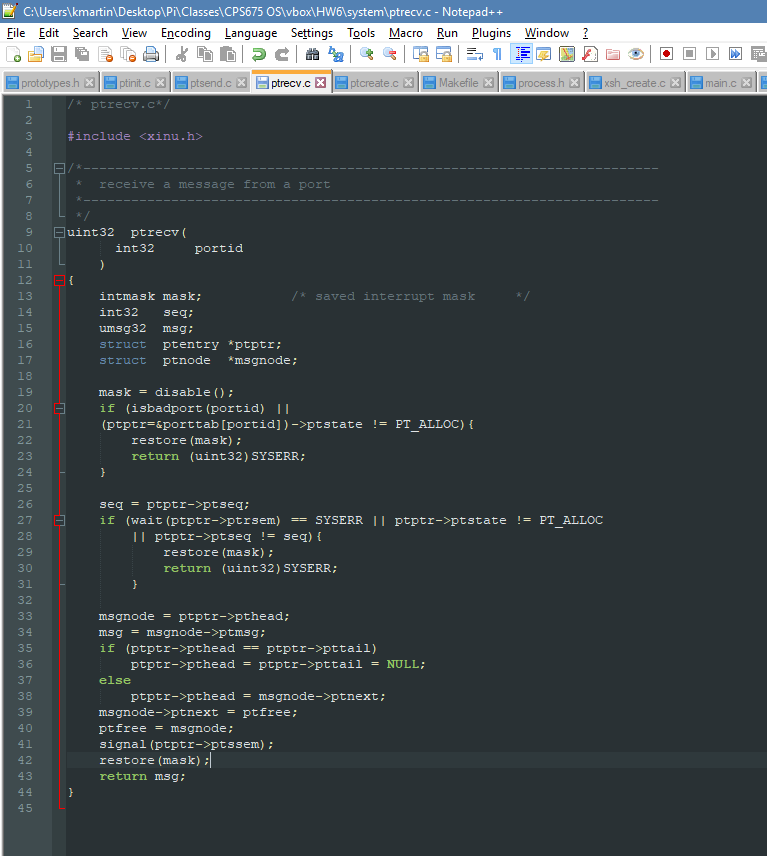
ptsend.c



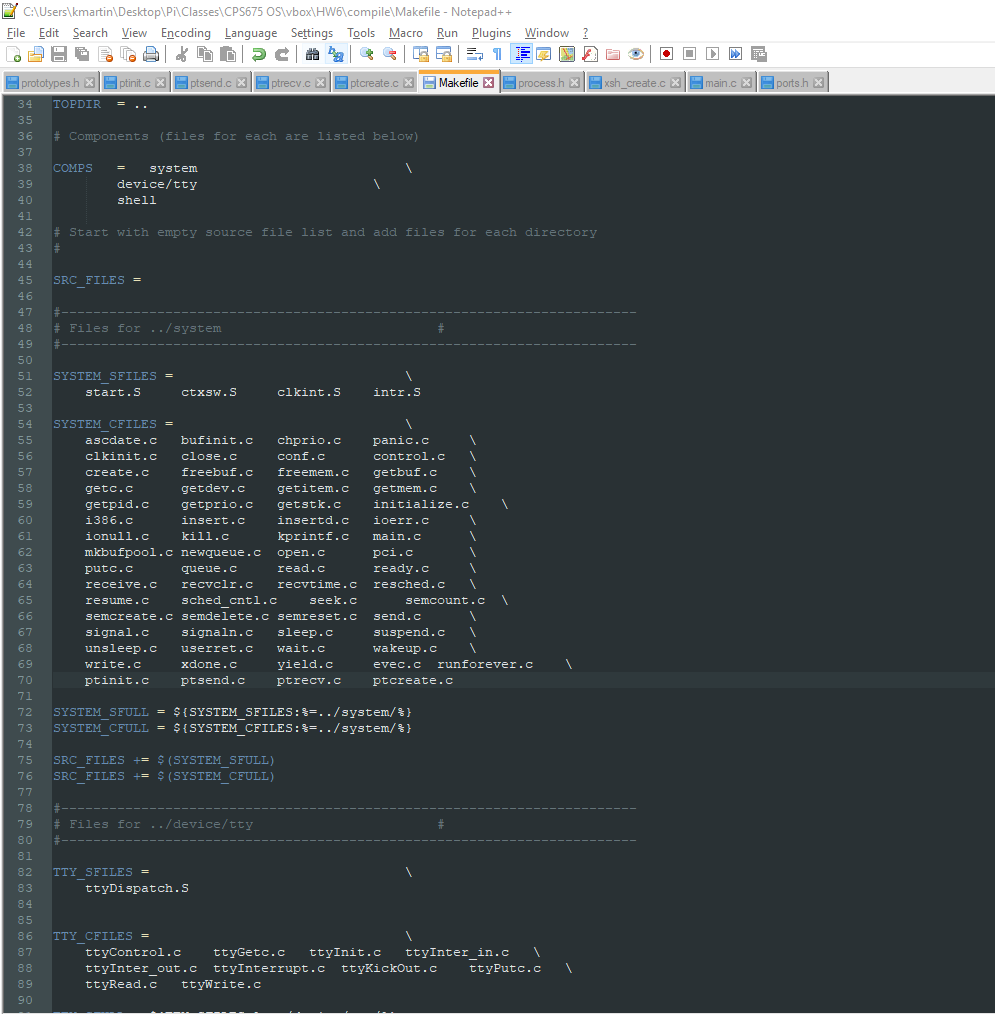
ptsend.c



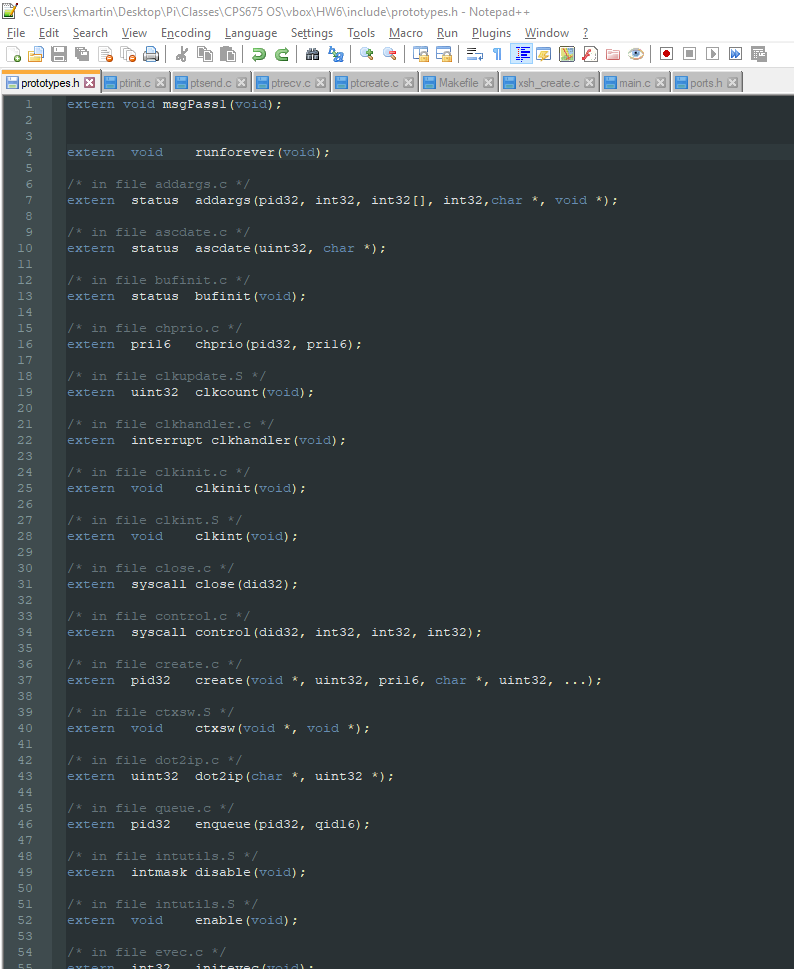
ptrecv.c



Makefile



Single function added to prototypes.h (to get the message passing function in our shell command going):



Finally, the shell command xsh\_create.c (we did not have to modify main.c)

