Project 5

Powell Fendley

This project was to simulate inter-proccess communication via port communication in OSP2.

First we call a Message.java method with super(length) in order to simulate a message being passed of a certain size. This method is used throughout the PortCB.java file, which is our next method.

First in PortCB.java we call super() and initialize and variable we may need. Next there is the do\_create method. This creates a new PortCB object and assigns it to the current task. Then this uses the setTask method to set the owner of the PortCB to the task it was assigned to. Next we set the port status to PortLive to show the port is active. Then we use a variable to keep track of how much message is left in our port’s buffer by using PortBufferLength as our port’s buffer size. Next is the do\_destroy method, which will destroy our PortCB object. First this method sets the status of the port to PortDestroyed. Next this method notifies all threads waiting on the port that it has been destroyed. Finally this method removes the port from the task’s table and nullifies the port’s owner. The next method is do\_send. This method will send the message our port has out. It does this by making sure the message is well formed, and then creating a systemEvent to handle the sent message. This then suspends the current thread, and checks if there is room in our port buffer to send the message. In addition, this checks two further complications that are: 1) if the thread is Killed we have to return FAILURE and 2) if the thread is not PortLive then we have to return FAILURE because these both mean the port has died. Finally we insert the message into the port buffer and notify threads that we have a message. Then we keep track of our buffer space and notifythreads through our system event, then return SUCCESS. The final method is the do\_receive method. This method does the opposite of the do\_send method by receiving and processing an incoming message. First the current thread is attained and then a systemEvent is started. Then we check to see if our buffer isEmpty(), and if it is then we suspend the thread. If not, we process the message. This also deals with the two complications listed above in do\_send because we have to check if the thread is ThreadKill or if the status is not PortLive again. To process the message we remove it from the buffer and assign it to a variable so we can return it. Next we update our buffer length to reflect the new space we have. Finally we notify all threads that we have received and taken the message and then notify all systemEvent threads of the same thing. At the end of this we return the message we received.

These are the steps to simulate port communication via messages in OSP2.