Lab6
Han Wang
wang2786(0028451697)

How to manage back-to-back invocation of the same signal?
Ans: The easiest way I can think of to avoid signal back-to-back trafficking is using the queue just like what we did for the sendblk(). Using XINU's original queue structure, we can easily implement a signal handling queue for each process. Then we dequeue them one by one using FIFO.

Modification implementation and Results?
Ans:

1. For part 3 signal RCV, I did not have to do a lot of chance since my lab5 part 4 was almost cover what this do, I added checking for Null pointer for function pointer and removed the checking statement in register function.

```
(Wang, Han)

wang2786

***PART3: Test for XSIGRCV***
Sender(PID:4) is sending message('A') to Reciever(pid:3)
Sender(PID:5) is sending message('B') to Reciever(pid:3)

Making a queue for msg: B

Enqueuing msg(B)
Sender(PID:6) is sending message('C') to Reciever(pid:3)

Enqueuing msg(C)
mbuf = A
mbuf = B
mbuf = C
```

2. For signal CHD, I implemented childwait() to handle difference cases of the childwait() call, added field in the process table that store returning child pid to keep track, and also another field to track how many running children I currently have in order to support those cases. Within Kill() I also check for callback function and try to handle the signal by unregister it from the process.

```
***PART3: Test1 for XSIGCHL***
parent(7) is running
child(8) is ruCHLDEBUG:childwait 4
nning(5)
child(8) is running(4)
child(8) is running(3)
child(8) is running(2)
child(8) is running(1)
child(8) is finished
This is Callback function chl_cb, a child process is about to end

***Waiting PART3: Test for XSIGCHL to Finish***
```

3. For signal XTM, it was quite simple, since time is handled by function's like clkhandler, which constantly gets called and changing the system time, I put my callback in there after I check the difference between wall time and process running time(clktime-start time), then we call the callback. I implemented unregister to unregister the process so that callback only gets call once.

```
***Waiting PART3: Test for XSIGCHL to Finish***

***PART3: Test for XSIGXTM***
clktime:11 seconds
XTM
walltime: 3
Wall time exceeded! XTMcallback(pid9) called at 14 seconds

***Waiting PART3: Test for XSIGXTM to Finish***
```

Each utility function and test functions are consolidated into LAB6_XXX_UTIL.c for easier access.