Lab6
Han Wang
wang2786(0028451697)

Since I used print statement to test my garbage collection in kill(), running all the test all together with all #define might make output a little crowdy. It does not effect functionality at all.
I put my print statement for garbage collection in #ifdef part4. So if you would like a clearer test output, disable #define 4 in main.c kill.c and freemem.c will make other test output look cleaner.:)

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.

———————————————

Part 4
For memory garbage collection, I have utilized original memblk structure and initialized “mylist” to book-keeping for all the memory allocated by user, and free it when process is about to terminate(kill). I modified getmem.c to add length and next.address into mylist with offset of 8(extraspace), in freemem I only have to remove it from mylist, and let freemem do the work for me. In kill, I loop through mylist and free them using freemem. The key part is I also modify memory.h since freestk() is also using freemem, I did the proper adjustment for freestk() to work with my modification. Free testing I allocate memory multiple times and interleave with multiple voluntary freemem. The correct output should match the exact same byte as started. Output as below:

```
***PART2: Test 1 Memory Garbage Collection
 250018296 bytes of free memory.  Free list:
           [0x0015A208 to 0x0EFC8FFF]
           [0x0FDEF000 to 0x0FDEFBFF]
creating first element(split)
user is freeing 256, at 0x0015A210, extraspace:8
creating first element(split)
creating non-first element(split)
user is freeing 512, at 0x0015A418, extraspace:8
creating non-first element(split)
user is freeing 512, at 0x0015A210, extraspace:8
user is freeing 512, at 0x0015A418, extraspace:8
 250018296 bytes of free memory.  Free list:
           [0x0015A208 to 0x0EFC8FFF]
           [0x0FDEF000 to 0x0FDEFBFF]
user is freeing 1016, at 0x0FDEFC08, extraspace:8

***Waiting PART2: Test 1 to Finish***


You reached the end of the Main().
```

As you can see, the memory start from xxx18296 and ended 18296, you are welcome to check memory in between to see the allocation, I removed some of the print statement for this snip for shorter screenshot. The print statement are still in the code

BONUS
With all the knowledge I have learnt from this course, I'd like to use this bonus as a opportunity to circle back to what we did for the first lab, to make it a full circle, in a sense. For our very first lab, the first thing we did is to give our XINU our name to print out every time we boot up. I would like to mimic that at the end of the semester by using the signal handling to get my XINU name so that we can always get the owner of the XINU by using this signal.

If user register the function, when process is about to resume, it checks for XSIGMY, if there is a my custom signal, its going to print my name! YAY!!

```
***BONUS: Test 1 MY SIGNIAL
XSIGMY

(Wang, Han)

wang2786

***Waiting BONUS: Test 1 to Finish***
```

Thanks for the amazing semester. I thoroughly enjoyed it. THANKS.