Lab 2

Kevin Martin
CIS657 Monday @ 10:00pm EST
Syracuse University

January 31, 2020

1 Introduction

Our team shared code, what worked, and what didn't. I found that moving the "resume" function below the newly created "create" functions yielded the proper resault. Without doing so, the "kill" function did not seem to get invoked. We also noted that including an additional "i" variable in the kprintf statement of the signaler function seemed to help it print better. I think it might be because the kprintf statement expects two arguments.

As far as the code itself, we modified the entire main function (highlighted in yellow). I kept the orignal main code commented out just in case. The "semcreate(20) gets us the first 21 numbers (as we need the semaphore to get to -1 before it blocks). The rest of the wait functions print out the requsted 5.

For the rest of the submissions, I'm just including the main.h file, the video, and the tar file of "xinu-x86-vm" (not the entire root directory).

2 Code

```
/*  main.c  - main */

#include <xinu.h>
void waiter();
void signaler();
sid32 sem;
pid32 wpid, spid;
void main(void) {
  sem = semcreate(20);
  wpid = create(waiter, 1024, 40, "w", 0);
  spid = create(signaler, 1024, 20, "s", 0);
  resume(wpid);
  resume(spid);

  return OK;
}

void waiter() {
  int32 i;
  for (i = 1; i <= 2000; i++) {
    kprintf("%d ", i);
    wait(sem);
  }
  kill(spid);
}


void signaler() {
  while (1) {
    kprintf("signaler is running \n");
    signaln(sem, 5);
  }
}
```

```
/*  main.c  - main

\#include <stdio.h>
\#include <xinu.h>

int main(int argc, char **argv)
{
        uint32 retval;
```

```
        resume(create(shell, 8192, 50, "shell", 1, CONSOLE));

        Wait for shell to exit and recreate it

        recvclr();
        while (TRUE) {
                retval = receive();
                kprintf("\n\n\rMain process recreating shell\n\n\r");
                resume(create(shell, 4096, 1, "shell", 1, CONSOLE));
        }
        while (1);

        return OK;
}*/
```