Group:

Rich Chen
Net ID: rhc57

Jon Getahun
Net ID: jjg224

In this assignment, we first implemented semaphores. To do this, we added the functions sem_init, sem_destroy, sem_wait, and sem_signal to proc.c. Sem_init creates semaphores while sem_destroy destroys them. Meanwhile, sem_wait and sem_signal are called while the semaphores are in use. The most trouble we had with this simply came from understanding how to use everything in our semaphore struct (mainly the spinlock). Once we realized that we can test whether or not something already has the spinlock, the functions became much simpler to implement.

Next, we implemented kernel threads in xv6. The functions added to proc.c were clone, join, and texit. Clone, which was heavily based off of fork, creates a child thread and returns its pid. A great deal of effort went into figuring out how to push the arguments and return value onto the stack. Our troubles mostly stemmed from properly dereferencing and pushing values. Thanh Ngo was very helpful on the forums for helping us understand how to properly set up the stack. His explanations about how the return values shared between texit and join were supposed to work were also extremely informative. The function texit is called at the end of the thread function, and stores the return value in a place that is accessible by the parent process. Join waits on a thread to finish and collects the return value from that space. A problem that we had was that we did not know where to put said return value. Eventually, we realized that we could just stick it in the eax of the thread, and reference that from the parent process.

While the main coding chunks were located in proc.c, we also had to make changes to various other files so that the functions could be recognized in xv6. We made changes to the Makefile, defs.h, syscall.c, syscall.h, sysproc.c, user.h, and usys.S to allow us to use our functions. For the most part, we simply mimicked the code that was already in the files to complete this part of the project. In the end, most of the time spent on this project was for understanding how exactly the functions (and related processes) were supposed to work, rather than actually coding the project.