**Name:**

* Sarah Depew

**Notes:**

* My log is found in log.txt
* I ran the Ziting tests and took care of the deadlocking cases by returning errors for these cases; I didn't want to deadlock, so I addressed the issues and simply did not permit the joins to occur
* I added a few misuse test cases Wednesday morning

**How to Run:**

* Put whatever code you want to run in test.c
* Run 'make'
* Link program with library as described in Appendix A
* Run ./test executable

**Known Bugs/Notes: (Some may be outdated, based on test, but wanted to record everything I had run into)**

* My yielding is not working 100% correctly for threads; I ended up having to sigprocmask too much of my code to allow successful interleaving to occur (I think I fixed this by modifying my masking locations)
* I have a ton of valgrind errors (these were fixed by the valgrind inclusion with the stack for ucontext\_t)
* Odd error relating to priority scheduling where I run for shorter time than expected, but thread function adequately runs and reaches the end of the run for the ZPW\_big\_job\_priority\_test.c
* There was one time I ran valgrind and got a segfault. I tried to recreate the error to no avail. I am wondering if there is a minor race condition that occurred. I knew about this and attempted to fix it, but was unable to recreate the issue. So, I am recording it here. This was running in Priority scheduling mode.
* Weird bug about printing and swapping for Priority on one test (Rachel was sent a screen shot) (same program as in above comment)
* I was unable to figure out how to break my code in userthread.c into multiple .h files. If I could have done so, I would have done this to make more .h's and make the code more readable.
* I am not 100% certain the ratio on my scheduler is correct. I followed your advice, but after running Eileen's scheduling test I am not sure I have the proper ratio. I think I do, but I wanted to document the possibility it was not 100% correct.
* I kept getting a persistent error with jiaping's wjp\_priority\_yield\_same test and wjp\_test\_fifo\_basic where I had a Conditional jump or move depending upon an uninitialized value error
* I decided on different behavior than jiaping for wjp\_test\_invalid\_join where an attempted join was valid on a thread that had already terminated that was attempting to be joined; I decided to return success if another join is attempted on a prior created, but finished, thread, so this test doesn’t fail as it does for them.
* When a failure occurs during a test, such as in wjp\_test\_invalid\_join2 and wjp\_test\_invalid\_priority, the exit happens immediately, so there is a guaranteed, and occurring memory leak as threadlib\_term was is not called. Similar behavior occurs in ziting’s tests with memory leaks due to not calling terminate. I thought this was expected behavior.
* wjp\_test\_misuse1.c also showed invalid reads, writes, and jumps; I was able to fix the segfault
* There was/could still be a free():invalid pointer and Aborted (core dumped) error on wjp\_test\_misuse1
* Not sure if this is pertinent, but I changed the type of the foo functions in kyu's tests to get the files to compile.
* I have an error on the kyu misuse test (I think I fixed this)
* I removed the valgrind information from my code, since it made it messier. (I can also submit the valgrind enabled version, if needed)
* I think there is at least one segfault that I missed...
* I could not figure out how to break the code into a series of .h files (see comment above)
* I am not 100% sure if freeUcontext(scheduler); on 309 and 274 is causing an issue
* Please note that some of the information may be outdated, but it is comprehensive to tracking the progression of the project.
* fifo\_null\_func.c has a memory leak, but that's because we exit on a failure and never call the terminate function for the library
* wjp\_test\_invalid\_join.c prints Hello World! three times, since I return a different thing/have different behavior for that type of failure (see note above)
* I wasn't able to add as many tests as I had hoped, but I did get some simple tests to ensure memory leaks didn't occur upon creation, joining, and yielding
* I added three more misuse tests the morning of Wednesday, March 21st

**Resources Used:**

* Rachel (she was amazing!)
* TA sessions
* Fellow classmates Mark and Kellie (to discuss design ideas with)
* GitHub for code examples (whenever I copied code, I listed the source fully in the source code for userthread.c)