CSC-335 Final Status Report group CDN

Hope Cristafi, Zach Dubinsky, & Tyler Nass

- Subsystems
  - Scheduler
    - Tyler
    - Fully implemented and fully works. Implemented as a simple priority queue that gives higher priority to processes the longer they spend waiting to be completed. This is done by adding an integer instance variable to the threadlistnode struct in threadlist.h and interacting with it in the schedule() method in thread.c.
  - Process Table
    - Hope
    - Proctable is fully functional and passes all the tests. I have fully integrated the feedback provided by Matt after project 3. Proc_destroy deallocates all of what has been allocated, and CVs were added so it works with exit and waitpid. The only thing that isn't deallocated is the file table. I initially called file_table_destory(), but had to comment it out because it caused a synchronization issue.
  - File Table
    - Zach
    - File is at ⅘ functionality. It is integrated and works with open, read, write, and dup2 tests. Improvements made since deliverable three are debugging, albeit to an unsuccessful end, and more synchronization checking. There is still a problem with file_table_destroy which causes a synchronization error when called from proc_table_destroy().
- Syscalls
  - getpid
    - Tyler
    - Fully implemented and fully works. Returns the pid of a process.
  - fork
    - Hope
    - Once again, I incorporated all of the feedback from project 3. Fork now makes a copy of the filetable for the child (and does not destroy it), and a new address space is not created for the child. The main issues I was seeing with fork was with the entrypoint helper function call; I got synchronization errors, then I had a stack issue with the child trapframe, that required me to copy it locally to resolve it. Fork passes the testbin/forktest.
  - execv
    - Tyler
    - Partially implemented. Does not fully work. Correctly checks for error cases and should be robust for all cases, including too many arguments, bad pointers, and no memory remaining errors. (I think) doesn't correctly interact with userspace. Destroys the addrspace. Destroys the old file table and replaces it with a new one. Enters the new process.

- waitpid
    - Tyler
    - Implemented with mutex locks and condition variables stored in the process table. Communicates with _exit. Does not fully synchronize with _exit, as the changes which add a second set of mutex locks and condition variables to ensure the exiting process doesn't delete itself before the waiting process collects its information was not pushed to the main branch. It is implemented in branch synch-attempt, along with the necessary changes to _exit.
- _exit
    - Hope
    - Exit passes the badcall test. I added the CV from the proctable so that it works with waitpid. For a while I was running into an issue that had a non-specific error message, but I found that I was calling thread_exit twice, once in exit and once in proc_destory. I fixed this and it did not crash.
- open
    - Zach
    - Passes some badcall tests, not integrated with read/write permissions flags. Passes open_test and I think f_test. Logically functional.
- read
    - Zach
    - Passes all but two badcalls, readwritetest, and palin. Made one change to update the seek_pos variable stored in the file struct on Thursday so the lseek syscall would work. Someone overwrote this. No other changes were made.
- write
    - Zach
    - Passes all but two badcalls, readwritetest, and palin. Made one change to update the seek_pos variable stored in the file struct on Thursday so the lseek syscall would work. Someone overwrote this. No other changes were made.
- lseek
    - Hope
    - Lseek passes 6 of the testbin/badcall tests. It fails when it tests "stdin when open on file." I added a check to make sure that it would not try and execute on these invalid standard input files, but it didn't work and it still fails the test. But the other 6 still pass.
- Close
    - Zach
    - Passes all badcall tests. Passes f_test I think, but does not pass closest. This is a semantics issue that I did not address for open or dup2 in the file table. As of now, I don't allow STD files (fd = 0,1,2) to be meddled with, they are static and stay in their place.
- dup2
    - Tyler
    - Implemented. Copies the file handle to the new file descriptor. Checks for invalid file descriptors.

- chdir
    - Tyler
    - Mostly implemented. Changes to use userptr were not completed.
- __getcwd
    - Hope
    - This passes all 3 tests. It gets the cwd by creating a temporary buffer, copying the string of the cwd from kernel space to the temporary buffer, and then copying out the contents of the buffer to user space using copyoutstr.