Progress Made On My Schedule

Nathaniel Flores

January 4, 2021

In this document, an event on the schedule will be shown with my progress next to it, and notes on how I got there. The schedule of events for this project go as follows:

First phase:

- Due by October 1st: First source in Noodletools, at least 50 lines of code.
- Due by October 26th: 100 lines of code, OS should at least print to screen. Complete research, all sources in Noodletools
- Due by November 2nd: Make sure all missing work is done and make sure the OS boots in a virtual machine. At least 200 lines of code at this point. Buy any other needed books and supplies for the project by this point, with the exeption of a PS/2 keyboard (for reasons too technical to explain fully).

All goals in this section have been met and I have moved onto the next section. Second phase:

- Due by November 16th: Split up code into different sections, ensure that at least 250 lines have been written. Make sure that the code compiles on other operating systems.
- Due by November 30th: Ensure that the code actually boots on real hardware, begin work on PS/2 keyboard driver.
- Due by December 7th: Buy a PS/2 keyboard by this date; test it after purchasing it. Complete shell and some basic apps to reflect what it can do.
- Due by December 21st: Complete PS/2 keyboard input and setup the boot process to read from it. We now have two way input to the user; keyboard drivers are extremely difficult and I do not expect for this to work fully.
- Due by January 4th of 2021: Wrap up development, freeze new commits to the Github repository and begin any work needed for Super Sunday/final presentation.

It is now the end of the project. I have finished most of the work on getting memory mapping and other small details to work, but unfortunately keyboards

proved to be far more difficult than I expect. Due to their complex nature and the fact I'd need to add a lot more to my kernel to write a driver for one, I have decided to hold off on writing one for my project.

What I was able to finish is a barebones C library, various libraries for the kernel and a decent-enough API where someone could create their own basic operating system or protected-mode bootloader off of my code. I have documented as much as I could have and attempted to provide as much explanation as possible so others can use my code for their own learning. Hopefully this proves beneficial to them.

Furthermore, the project is now pathetically easy to build and will be super easy for anyone to try out on any GNU/Linux system. It boots on real hardware and exhibits as many features as I could pack into it over a couple of months. This was a valuable learning experience but I'm excited to return to normal application development where I have a debugger to help me out.

So long and thanks for all the fish!