Daniel Martinez
CS-3600
Rajan, Ranjidha
4/24/21

The first thing I wanted to look for in Xv6 was to see a certain locking/semaphore mechanism. In the second page of the main (pg. 10 in PDF), they initialize something that looks familiar for locking. The method they use in this code is called spinlocks. This is a different way to lock critical sections specifically for multi-threading according to it's documentation. In multiple sections, you can see the spinlock being used to protect critical sections. The only difference from what we've learned in class is the name of the class. Otherwise, it's used the same way in almost every scenario it's mentioned in the code. Page 72 of the PDF and lines 6521-6526 demonstrate a clear example of how it's being used.

Page 30 of the PDF demonstrates multi-threading with selection of the CPUs. The comment above the initialization of it (line 2329) describes how similar it is to Linux Pthread. The CPU is initialized as a number and a structure. The number is called "ncpu", which will be the tool they will be used to tell which cpu they are on. Page 77 of the PDF, the code is using ncpu to control processes and find parents/childs. There is also a flagging system being used that is supported by the cpu threading they have implemented. We've seen the same method for when finding the child or parent data in processes, which is how I can tell that it is similar to what we had learned in class.

Page 33 of the PDF has shown states. We've practiced this to learn about deadlocks in class and it looks like in the code given, it's used the same way. The code implements a state named procstate. This provides the code many different states to see how the code is running. A good example of this would be the state ZOMBIE. This state uses panic() for a way to display an error when the process has turned into an error. These states are used from the earlier mentioned cpu threading to find what state the process we are following is in. We've seen this method with the dining philosopher's problem. We used states to find what philosopher (process) was eating, hungry, and thinking. The same concept is applied.

The entirety of Xv6 demonstrates an operating system created in C. Using most of the methods we've learned in class, it was able to create a stable and easy to read program. The purpose of the code establishes a good example of what an OS should be built like.

Most of the code is written in a different way from what we've learned in class. The same ideas are presented but the creators of the program wrote it with different classes. A good example of this would be the fact they use struct or enum a lot more than we have in the entirety of the class.

The way I know I've learned something from this class would be the fact that I understand good sections of the code. Even though it's structured differently, I can understand what method they're going for. For threading, they didn't use pthread but instead initialized two different variables to find/create threading. As far as it goes, what I want to learn more about in this class would be about the other methods to create a stronger/stable OS or programs. For example, I've never heard of panic() before but it looks helpful for debugging. Otherwise, I'm happy with what I've learned in this class.