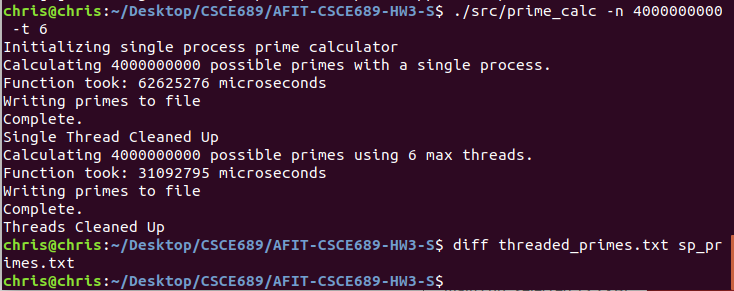
# Repository Location:

<https://github.com/chrisvoltz19/AFIT-CSCE689-HW3-S>

# HW3 Questions

Note: I ran my VM in VirtualBox with 6 processors.

Proof of working on 4000000000



1. Performed on the size 4000000000 to better illustrate differences
   1. Performance gets its best averages when the number of threads is equal to the number of cores. This is because that is the maximum number of threads that can be running at any one time on the machine. Using fewer threads than that doesn’t take full advantage of all the available resources, so the averages won’t reach their best performance. When more threads are used than cores available, all the resources are used, but since they switch more often, they tend to slow down due to the overhead associated with switching. This can be seen especially at the higher numbers (in the 20s).
   2. In this demonstration, I used 6 cores. The graph supports the concepts illustrated above. There is significant improvement for the first couple and then it is close to as good as it gets for threads = number of cores. It then slightly increases due to higher overhead.

|  |  |
| --- | --- |
| Trial # | Performance (s) |
| Trial 1 | 27.908296 |
| Trial 2 | 30.812403 |
| Trial 3 | 30.649052 |
| Trial 4 | 29.906314 |
| Trial 5 | 30.714239 |
| Trial 6 | 30.504918 |

Note: These trials were run on 4000000000 with 6 cores (and thus 6 threads).

The performances were relatively close. Some variation was likely due to different cache hits, and just different timing in which the threads got values they could operate on in slightly different timings according to the scheduler so that added some variation. However, all were below 31 seconds which is about half of the single threaded variant. Since they were pretty close, that means the algorithm used performs consistently.