1 Design Plan

2 Questions

1) **What do you think the main point of this assignment is?**
   We believe that the point of this assignment was to allow us to learn the process of task scheduling that is done by the kernel. This also allowed us to figure out various task scheduling algorithms that are implemented into the Linux kernel already. Knowing how to select or create a good scheduling algorithm is good because it lets you have the power to choose a scheduling algorithm that better suits the needs of an application or process. There is no perfect algorithm, many have pros and cons that are better or worse for specialized situations.
2) **How did you personally approach the problem? Design decisions, algorithm, etc.**
   We looked for resources online and found out what it takes to allow a Linux system to change the scheduling algorithm. We learned that you can either change the scheduler algorithm at run time or as a boot setting. For simplicities sake, we just change it at run time using the command "echo (algorithm of choice) ¿ /sys/block/(disk)/queue/scheduler". We confirm that the algorithm has been changed with cat /sys/block/(disk)/queue/sc which displays which algorithm is being used. For our scheduler we decided to go with the "CLOOK" approach, or circular look. This will dispatch requests for sector information in a linear fashion and wrap around to the other side of the disk when there are no more sector values higher than the current request sector value. We achieve this by sorting requests by sector value in the queue. The dispatched requests will flow chronologically up the queue and wrap around to the smallest sector value of the request queue after it dispatches the highest sector request.
3) **How did you ensure your solution was correct? Testing details, for instance.**
   Once we patched our linux default schedulers to include our sstf implementation we changed the scheduler using the make menuconfig command. Once we changed to our scheduler, we did a simple read and write test. We created a new "testFile" and echo'd various statements into that file. We then used "cat" to read out the file contents to the terminal to see if the contents matched the expected output from the echo commands and the file was structered as expected. From this, we can assume our algorithm is functioning correctly. We also used checked print statements of our sorted list to make sure that it was indeed sorted and wrapping to the front of the list as expected.
4) **What did you learn?**
   We learned a lot about the details about various scheduling algorithms, the basic structure of what a scheduler does (merging and sorting), the levels of task completion from request to disk read and write, the process of choosing a scheduling algorithm from Linux, the structure of a hard disk drive, the pros and cons of various scheduling algorithms, and more.

3 Version Control Log

4 Work log

10/26: began research on scheduling algorithms

10/26: began research on structure of hard disk drives

10/26: began research on levels of task completion (request to read/write on disk)

10/27: began latex write up file

10/27: began c file implementation

10/27: edited kernel files to include our algorithm

10/27: researched elevator algorithm implementations

10/28: continued work on c file implementation

10/28: continued work on latex write-up

10/29: finished work on c file implementation

10/29: created python testing script

10/29: continued work on latex write-up

10/30: testing via python scripts