PROJECT 2

Lance Umagat

Project 2 Write Up

The design you plan to use to implement the SSTF algorithms Since it has to have requests in queue in order for dispatch to even run its implement as displayed in the noop I/O scheduler code with lots of printing. I pretty much add all of the pending requests to the queue based on cases. If the queue is empty just simply add the pending requests. If the queue is not empty check to see if its sector value is between its front and back sectors then insert it. If those conditions don't suffice then add it to the back since it clearly does fit anything in the front of the queue. For the dispatch was pretty simple since you just dispatch whatever has the smallest gap thus the SSTF I/O scheduler's purpose, but since it's sorted it's always the next one that gets dispatched.

Answer the following questions in sufficient detail:

1. What do you think the main point of this assignment is? To learn how to build the kernel with a I/O scheduler and to figure out a simple algorithm using the already created functionalities.

2. How did you personally approach the problem? Design decisions, alogorithm, etc. I spent hours on looking at noop's I/O scheduler and the Deadline I/O scheduler to see how everything worked. After that it seemed pretty simple since I just follow what noop is doing except I went a little further by comparing sectors with each other then dispatching them using noop's dispatch as a reference.

3. How did you ensure your solution was correct? Testing details, for instance. I used print statement to see where I'm in the code since it literally jumps everywhere, but it's pretty synchronous. It would fill up the queue with requests then dispatch them, so I was able to see if they are able to back merge correctly by checking how many are on the queue while checking what gets dispatched. Since it is sorted everytime, it will aways back merge and dispatch in the same order.

4. What did you learn? I learned to deal with kernel panics since there's so many things that can go wrong if one little thing has an error. It was pretty cool that I can change the default scheduler to my own and using it to see what happens with the kernel.

Version control log (formatted as a table) – there are any number of tools for generating a TeX table from

repo logs

acronym	**meaning**
V	`version`
tag	`git tag`
MF	Number of `modified files`.
AL	Number of `added lines`.
DL	Number of `deleted lines`.

V	**tag**	**date**	**commit message**	**MF**	**AL**	**DL**
1		2016-4-21	Modified Makefile and Kconfig, and added sstf file	3	276	1

Work log. What was done when?

1) 4/19/2016 I copied the noop code over and modified the Makefile/Kconfig.iosched file.
2) 4/21/2016 I read the noop and deadline I/O scheduler to see what helps.
3) 4/21/2016 I started using prints everywhere and started playing around.
4) 4/21/2016 I finished the assignment and convinced myself it works with how it add and dispatch the requests and committed.