CS4444 Project 2

Shawn Cross, Ryan Crane

October 30, 2017

We plan to modified the the dispatch and add functions from the original noop scheduler to implement the LOOK algorithm for the scheduler. We made it so that you can go both forward and backward in the dispatch function when finding the next request to dispatch. If the distance to a request in the opposite direction is closer than a request in the current direction the direction will change. The queue sorted after a request has been dispatched. If the queue is empty then we add the new requests directly to the queue. If there is already requests in the queue then when adding then we sort during the insertion. the requests are inserted into the queue based on position.

1 Version Control log

Detail	Author	Description
88e93dc	Ryan Crane	dining philosophers problem, looks correct
35c460c	Shawn Cross	adding the write up for the concurrency problem.
3f0111c	Shawn Cross	More done on the write up and made it so you can compile for the makefile.
2d47960	Shawn Cross	slight change to tex file.
9aa562c	narxy	Update philosopher.tex
f8b1617	narxy	Update philosopher.tex
651df07	Shawn Cross	fixed spelling errors.
7555688	Shawn Cross	uploading tarball and created the sstf-iosched.c file
4bc64ff	Shawn Cross	Uptdate to the Kconfig file, update to the Makefile, update to menuconfig, and sstf-iosched.

2 Main point of the assignment

We believe that the main point of the assignment was to understand how requests are dealt with in an operating system. It was also begin to understand how to interact with the kernel.

3 How we approached the problem

First we started by figuring out how to get our scheduler working with the kernel instead of the noop scheduler. Once we were able to do that we began to figure out how we were going to implement the look schedluer. We knew we needed to be able move forward and backward and that there needed to be two sorts. First we firgured out how to get the forward and backward working by checking if the the next request in the direction that we were currently going is closer than the next request in the oppsite direction. If it was closer than we continue in that direction and if it was futher away then we change direction and go to the closer request. After a request is dispatched then we will re-sort the queue. We also needed to modify the add function so that we would sort the requests as they were inserted into the queue. We did this by using an insertion sort as requests were added into the queue.

4 Correctness and Testing

We know that our secheduler was being used because we used the command

cat /sys/block/hda/queue/scheduler

and got an output

noop [sstf] deadline cfq

this shows that our sstf scheduler is being used as we wanted. We then added print statements to our scheduler code so that we could determine when requests were being added and dispatched from the queue and by doing this we were able to show that for every requeust that was being added

to the queue it was also being dispatched as well which means that all requests were being dealt with. We also added print statements to declare when the direction flag had its value changed to show that we were in fact changing direction. With all of these things working as we expected them to we were able to prove that our look scheduler was properly working the way we wanted it to.

5 What we learned

We learned that you have to make modification to multiple files when adding new features to the operating system. We also learned that writing code for the kernel can be hard and confusing because we could not find a lot of information on some of the built-in function being used so it took us some time trying to figure what they did and how they worked. We also learned that can be a ton of I/O requests and they can all happen very quickly. We also learned that there are many different ways to do the scheduling.

6 TA Evaluation