Group 15 Assignment 2 Report

Ryan Howerton, Jinfeng Li, Vincenzo Piscitello
CS444 Spring 2018

Abstract

In the second assignment for Operating Systems II, we edited the provided kernel code to implement a new I/O scheduling algorithm. We decided to implement the C-LOOK algorithm, which continually services I/O requests in order from the lowest targeted sector on the HDD to the highest. We started by looking at a reference file for our SSTF scheduler, then creating a design based on the functions in that file. We logged all of the changes we made to files and files added in a version control log. We created a work log detailing when the group met and what we did when we met. Finally, we reflected on our process for approaching the problem as well as the work we did.

I. Design

Our design involved taking the noop-iosched.c file and changing it to sort new requests by the sector they target, then service the requests in ascending order (C-LOOK algorithm). Specifically, we modified the add_request function to check the sector on the new request, search through the existing elevator list to find the two elements with a lower and higher sector and insert the new request in between them. This way, as the elevator approaches each request, it will approach them from lowest to highest, and once it hits the top of the list it will switch back down to the bottom.

II. Version Control Log

Commits
commit 2dfc59e0bd867fc86569bedbad944e0a90b7158d **Author:** Vincenzo Piscitello <piscitev@os2.engr.oregonstate.edu > **Date:** Sun May 6 16:35:40 2018 -0700 Ensure SSTF Scheduler is Commited
commit 5ea65c92514eb9f2890b02f6ebd0eaaff17f7662 **Author:** Vincenzo Piscitello <piscitev@os2.engr.oregonstate.edu > **Date:** Sun May 6 16:02:05 2018 -0700 Completed Assignment 2
commit e736deabb4c7498afb804c84ec5dda0bb56ad72d **Author:** Vincenzo Piscitello <piscitev@os2.engr.oregonstate.edu > **Date:** Sun Apr 15 15:42:00 2018 -0700 Added Kernel Debugging Print Statements
commit f0c5ea77098c2ded47d2b438509a628907489ca6 **Author:** Vincenzo Piscitello <piscitev@os2.engr.oregonstate.edu > **Date:** Sun Apr 15 15:30:00 2018 -0700 Implemented C-LOOK Add Request
commit 398f79b23165ecb79a0efeac34ca6b3787ec84ca **Author:** Vincenzo Piscitello <piscitev@os2.engr.oregonstate.edu > **Date:** Sun May 5:19:20 2018 -0700 Initialization Complete
commit 1222d7e6c7804d9a7232654df3906500f653c2b8 **Author:** Vincenzo Piscitello <piscitev@os2.engr.oregonstate.edu > **Date:** Sat Apr 14 15:36:44 2018 -0700 Initial Commit
commit 660613d1a4e94144490850b6c3d350331860fac4 **Author:** Greg Kroah-Hartman <gregkh@linuxfoundation.org > **Date:** Wed Mar 18 14:11:52 2015 +0100 Linux 3.19.2

III. Work Log

Saturday, April 29th	Researched and planned our method of implementing the C-LOOK algorithm.
Saturday, May 5th	De-constructed each elevator operation to understand how it works then wrote pseudo-code to alter NO-OP to meet the C-LOOK requirements. After we completed our pseudo-code we implemented the C solution in sstf-iosched.c. We then recompiled menuconfig and recompiled the linux kernel so that the SSTF scheduler could be used.
Sunday, May 6th	Completed the homework document and tested the system to ensure it would build.

IV. Guided Question Answers

1) What do you think the main point of this assignment is?
   - The main point of the assignment is to learn how to implement the LOOK algorithm in Linux I/O scheduler and configure it in VM, as a real life example for how kernel schedulers operate and are implemented.
2) How did you personally approach the problem?
   - We created a plan by referencing the noop scheduler, breaking down the elements of it on a whiteboard, and writing out our potential changes as pseudo-code. As soon as we thought we had a solution, we copied the noop file and renamed it for sstf, then implemented our changes with printk statements for debugging.
3) How did you ensure your solution was correct?
   - We compiled our kernel with the correct flags and modified Makefile, ensured that its default scheduler was the new sstf scheduler, then booted up the Qemu virtual machine and checked the kernel output for our print statements.
4) What did you learn?
   - First, we searched through the dependencies in the kernel to learn about potential tools at our disposal. We learned a great deal about all of the structures included in linux, and how to manipulate them. Then we delved deeper into the workings of the noop scheduler, to see what needed to be changed to implement the C-LOOK algorithm. We refreshed our memories on circular doubly-linked lists, and how to implement insertion sort on such a list.
     As for version control, we learned how to use the git diff command to create Linux patch file between two git commits and apply this patch file to source code without editing the source code directly.
5) How should the TA evaluate your work?
   - The TA should evaluate our work by viewing the changes made through the patch file provided and ensuring the description of our solution is correct. To apply the changes use the bash git patch command passing in the ”group15_hw2.patch” file.