Project 4

CS444 - Spring 2017

Terrance Lee, Raja Petroff, Markus Woltjer

Abstract

The following document contains information about Project 4 which includes the design plan of the Kernel Slob Slab, version control, and work process.

CONTENTS

1 The design we planned to use to implement the Simple Block Device. 2

2 Version Control Log 2

3 Work Log 2

4 2

4.1 What do you think the main point of this assignment is? 2

4.2 How did you personally approach the problem? Design decisions, algorithm, etc. 2

4.3 How did you ensure your solution was correct? Testing details, for instance. 3

4.4 What did you learn? 3

1 The design we planned to use to implement the Simple Block Device.

2 Version Control Log

User	Commit Message	Date
terrancelee81	Added Makefile	May 31st
terrancelee81	added project4.tex	May 31st
petroffr	pushed initial slob.c	June 1st
terrancelee81	added concurrency file	June 1st
petroffr	did part1 of concurrency	June 1st
markuswoltjer	added asmlinkage for slob_used and slob_free system calls	June 2nd
markuswoltjer	successful test of concurrency.c	June 2nd
terrancelee81	updated latex docx	June 4th

3 Work Log

- May 30th - began working on the kernel and concurrency part of the assignment
- May 31st - Makefile and latex got added
- June 1st - Slob file got added
- June 1st - Part 1 and 2 Concurrency file got added
- June 2nd - Slob file got updated
- June 2nd - Concurrency got tested
- June 4th - latex got updated
- June 5th - Finalized Report

4

4.1 What do you think the main point of this assignment is?

I believe the main point of this assignment is to understand how the SLOB first-fit alogorithm works, understand how it differs from the best-fit algorithm, and adjust the file to implement the best-fit allocation algorithm. While not specifically addressed under Project Four, concurrency assignments four and five were also completed simultaneously.

4.2 How did you personally approach the problem? Design decisions, algorithm, etc.

Almost all of the work to be done was, as one might expect, in the allocation and deallocation functions. A few other functions were added to support these changes. First we looked into what exactly was meant by the best-fit algorithm with respect to fragmentation, since it originally refers to curve-fitting, which is a very general method in multiple disciplines. Per the usual method, we went through the original slob.c file commenting more in depth where we saw the logical applications of each step of the first-fit application, and making note of what would need to replace that step.

Then, we carefully coded all replacements, since debugging isn't as easy as usual. Upon failed boot, we looked back at the file and considered which of the changes could have contributed to a fatal flaw. Our first correct boot reduced fragmentation as expected.

4.3 How did you ensure your solution was correct? Testing details, for instance.

We implemented some new system calls that returned the amount of memory used and the amount of free memory. We used this to calculate the percentage of fragmentation in a small test file that we run on the VM after kernel boot.

4.4 What did you learn?

We learned how to implement the best-fit memory allocation algorithm and Linux system calls. We also learned about the data structures and fundamental functions underlying memory management in the Linux kernel.