Project 4

Lance Umagat

Project 4 Write Up

Prepare a document containing the design you plan to use to implement the necessary algorithms. It took a while to understand how slob.c works, but after a good chunk of print statements it became more clear. We have prev and cur where both gets the previous and current units. It goes to slob_page_alloc to allocate a slob block into memory. What slob_page_alloc does first for first fit is that it looks for the first available space defined as avail that is big enough to first the unit size which apparently is units delta then it allocates it when it finds enough space. To implement best fit was simple after seeing how it works. Since slob.c already provided with the if statement where it checks for spaces big enough for the slob block, it all comes down to finding the smallest difference between all of the available spaces throughout the page and get the smallest difference which is just a min statement. After we found the min I just reused slob.c first fit algorithm and not find first fit, but to allocate the slob block to the the available space that best fits it. To do the system call and fragmentation part I had to learn that from a youtube video where it showed how to implement system calls within the kernel. Since all of the allocation is done with the slob_alloc function and slob_free for freeing of the memory, I just use mem_used and mem_claim for the amount of memory that was used and claimed by the implementation respectively. In slob_alloc, when it allocates which is after the $\text{slob}_page_alloccall, Iaddedthesizeallocatedtomem\_used.Ifitistoobigtotakeupspaceandneedstoallocateanewpage Iaddedthesize$

Answer the following questions in sufficient detail

1. What do you think the main point of this assignment is? The main point of this assignment is to learn about how allocation of memory works and how with different algorithm such as first fit and best fit, it would have trade off such as for first fit it takes more fragmentation to do the same work as best fit. The pro of first fit is that it allocates a lot quicker than best fit. Best fit has to search through the whole page just to find where to allocate memory.

2. How did you personally approach the problem? Design decisions, alogorithm, etc. I had to read slob.c first fit algorithm and tried to figure out how first fit truly worked even though I have a general idea. After looking at the functionality and using plenty of print statements to see what the function was doing I was able to write the code for best fit since it was a straightforward implementation of finding the min difference and allocating the memory. That was the best solution that I can think of. For the system calls, I was pretty lost and it takes hours online to figured the basics then try my own way. After implementing the system calls, I added the calls to the first fit and best fit slob.c files and created a test file that can be ran in qemu to look at the fragmentation. That's how I compared the two algorithms. From the results it was clear that first fit claimed a lot more pages to allocate memory compared to best fit.

3. How did you ensure your solution was correct? Testing details, for instance. I used print statements to print out the slob block size that I'm trying to allocate and print out all of the available space sizes for the slob block. Then I print out the best available space found for the slob block. It was clear when I saw all of the available spaces that can fit the slob block and the best available found space being the one with the smallest difference since it doesn't print a ton of available spaces for a page.

4. What did you learn? I learned how to implement a memory allocation algorithm based on the existing first fit slob.c file. The biggest learning experience from this was the system calls where I implemented the system calls from slob.c and linked it to the syscall header file and syscall table where I inserted the system call prototype and system call number respectively. After that I was able to created a test file to be scp into qemu then ran which invokes my system calls to get the memory used and claimed by my slob.c. It took

some trial and error to get it working since I only learned how to compile a system call with the kernel, but there's no way for me to get specific variables inside my slob.c from invoking built in kernel calls.

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		2015-06-03	Final push of project 4, needed fresh linux master	15	40	2129

Work log. What was done when?

1. 05/28/2016 Played around and printed a lot inside slob.c
2. 05/28/2016 Started trying to implement best fit, Got new Linux yocto to test with.
3. 05/30/2016 Got best fit to work, added print statements to view output
4. 06/01/2016 Working on getting syscalls to work, got it to work, testing and comparing first and best fit
5. 06/03/2016 I add, commit, and pushed all work to the repository.