Project 4: The SLOB SLAB

Christian Armatas and Mihai Dan

ABSTRACT

This document contains the design for necessary SLOB first-fit algorithms, a version control log table, a work log, and answers to the provided questions. The solution to Project 4 can be deduced from the previously mentioned sections, as well as the implementation. The purpose of this project was to better our understanding of allocation algorithms and system calls.

DESIGN FOR IMPLEMENTATION OF NECESSARY ALGORITHMS

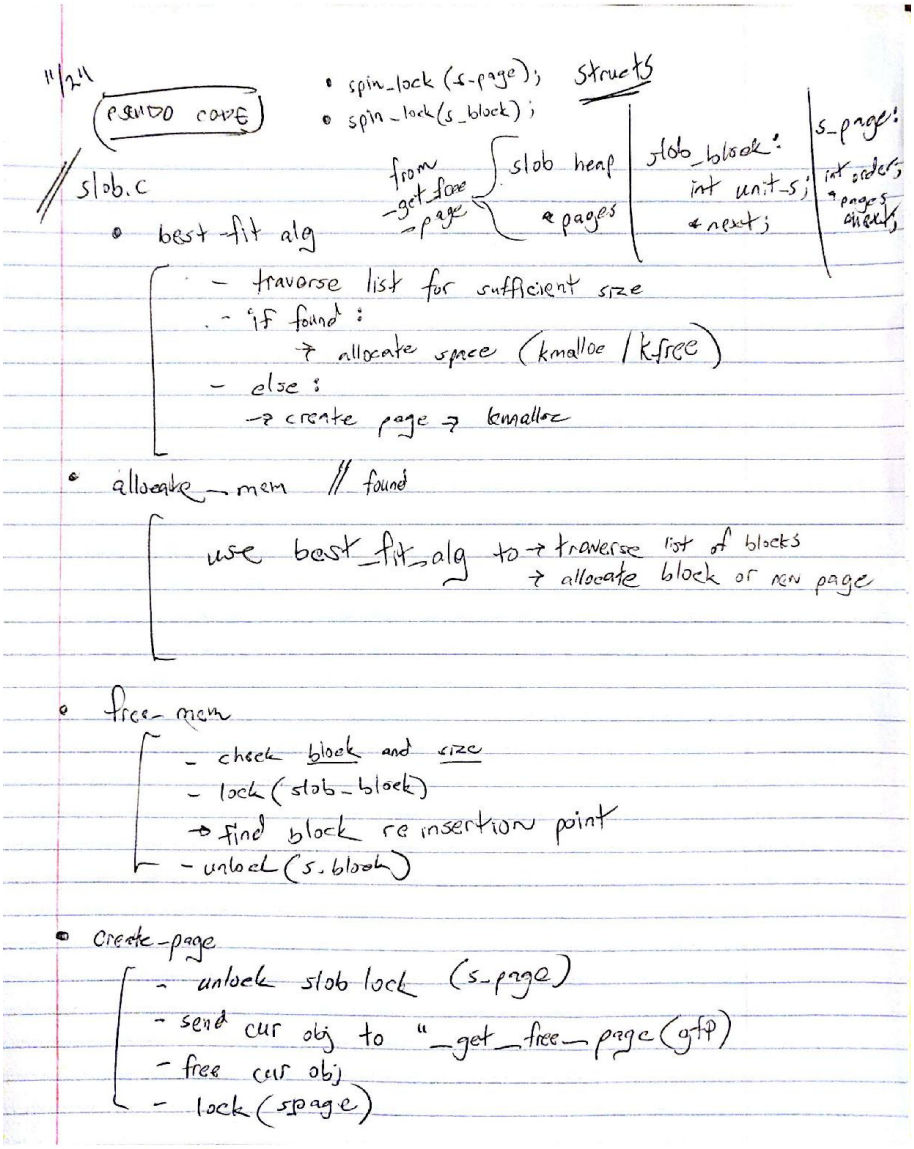

Fig. 1: Handwritten pre-implementation design notes.

VERSION CONTROL LOG

Table organized by most recent commit.

commit 8efddef23396135ebe4ef4db4fa4cccb53b07587 Author: armatasc ¡armatasc@onid.oregonstate.edu¿ Date: Mon Nov 28 20:46:39 2016 -0800 commit message: Pushing final code changes, as well as SLOB fragmentation measuring program.
commit 3acd201e9919a082c3621b2e86ddd0a4b676bd04 Author: mihaidan (danm@oregonstate.edu) Date: Mon Nov 28 17:29:34 2016 -0800 commit message: updated project write-up
commit 07db444726d8595486569dff2597e4740bbb1333 Author: mihaidan (danm@oregonstate.edu) Date: Sun Nov 27 21:15:18 2016 -0800 commit message: updated slob.c file
commit 659fee07c647c311cacc075aee865374221163f4 Author: mihaidan (danm@oregonstate.edu) Date: Fri Nov 25 17:45:28 2016 -0800 commit message: design for implementation
commit e064051f8e12ba12162acf69f872b8cb04382e85 Author: mihaidan (danm@oregonstate.edu) Date: Thu Nov 24 22:22:05 2016 -0800 commit message: project4 initiated

WORK LOG

- Friday, 11/25:
  - After gaining some insight to the SLAB/SLOB project in class, we began planning and outlining the assignment.
  - The project was going to consist of a few main portions: A program to measure slob.c fragmentation metrics, a best-fit algorithm (designed/edited in slob.c), and implementation of custom system calls to get the metrics.
  - We posted the preliminary project outline doc to GitHub and began the slob.c code, as well as understanding how to create custom system calls.
- Sunday, 11/27:
  - Today, we nearly completed a large portion of the coding changes required to run best-fit in slob.c...
  - Within slob.c, we created syscall definitions to measure fragmentation metrics for our algorithms (first-fit and best-fit).
  - We created (a rough attempt at) the best-fit algorithm, though we ran into issues when we attempted to compile our new kernel and image (new slob code; set in make menuconfig) and the kernel would crash.
  - Because the ”target remote :5517” (to target our Qemu kernel) failed get a response from our kernel, we had to trouble shoot numerous alternatives, including the syscall table, slob.c, qemu parameters, previous assignment menuconfigs, etc... We later found out the slob.c code and the inconsistent order of the system call declarations were causing our issues.
  - Lastly, we made final changes to the 32 bit system call table and the syscalls.h header file to reference our new ”sys_slob_free” and ”sys_slob_used”.
- Monday, 11/28:
  - We did some more research pertaining to system call mechanics and implementation. We figured out where we need to declare the system calls in order for them to be recognized by the kernel.
  - Once me made appropriate changes to slob.c and necessary syscall files, our kernel was able to run.
  - From within the vmlinux kernel, we created a file ”test.c” to test our system calls, 353 (slob_used) and 354 (slob_free), to ensure our slob.c code was accurate.
  - Pushed the final changes to GitHub, along with our syscall_32.tbl, syscalls.h, and slob.c code changes.

QUESTIONS

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

- The main point of this assignment was to familiarize us with the implementation of various memory allocation algorithms. Specifically, this assignment highlighted the first-fit and best-fit allocation algorithms. The first-fit algorithm places a specified block in the first available spot. The best-fit algorithm traverses through all available pages until an ideal spot is found for the specified block of memory.

  Both of these algorithm come with different benefits and are each used in different scenarios. First-fit allocation is more efficient than best-fit because it may not necessarily need to traverse the whole page to find a suitable spot. This first-fit advantage is presented with a clear disadvantage when compared to best-fit allocation. Best-fit allocation leads to a more organized outcome, as the first spot found may not be the best fitting. As previously mentioned, there are used depending on the intention of the engineer and the purpose of the machine.

  System call implementation was also a learning opportunity in this assignment. By implementing the `slob_used` and `slob_free` system calls, we got first hand experience with the mechanics behind system calls.

  The best-fit implementation which replaced the first-fit algorithm is shown below.

```
/*** Best Fit Algorithm Implementation ***/
/* Checks if the next slob page is NULL */
if(next_sp == NULL)
        next_sp = sp;

/* Find the smallest available page */
if(next_sp->units > sp->units)
        next_sp = sp;

/* Attempt allocation on page */
if(next_sp != NULL)
        b = slob_page_alloc(next_sp, size, align);

/* Getting fragmentation metrics */
temp_list= &free_slob_large;
list_for_each_entry(sp, temp_list, list){
        available_units = available_units + sp->units;
}
temp_list= &free_slob_medium;
list_for_each_entry(sp, temp_list, list){
        available_units = available_units + sp->units;
}
temp_list= &free_slob_small;
list_for_each_entry(sp, temp_list, list){
        available_units = available_units + sp->units;
}
```

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

- Before any project planning or implementation, we researched the best-fit algorithm and what it entailed. Simply put, the best-fit algorithm searches through all the pages until a desired spot is found. The existing implementation of the slob.c file inside our kernel used the first-fit algorithm because it took the specified block of memory and placed it in the first spot that offered enough free space. In order to change this to our desired algorithm, we needed to replace the first-fit with the best-fit allocation in the slob.c file, specifically in the `slob_alloc(...)` function.

  Being unsure about the correct syntax and mechanics of system call, we were faced with a bit of a challenge. Extensive research led to the discovery of several files that needed to be modified in order to integrate new system calls into the system. After adjusting our kernel with the new found information, we achieved proper linking between the different files.

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

- A portion of this assignment was focused on testing the efficiency and fragmentation of the first-fit and best-fit allocation algorithms for block allocation. Designing and implementing a tester was significantly easier

than the implementation of the SLOB or system calls. The `test.c` file consists of two system calls made to `slob_free` and `slob_used`, which return the amount of free and used space, respectively. This test file can be ran on both the first-fit and best-fit allocation algorithms. As expected, the results show that the best-fit allocation algorithm yields lower fragmentation than the first-fit algorithm. As stated before, this is due to the fact that best-fit produces more organized pages.

4) What did you learn?

- This assignment was filled with learning opportunities. The first being that there are different algorithms when deciding block allocation. The two algorithms covered in this assignment are best-fit and first-fit. Neither is better than the other, but each have their own advantages that are accounted for when deciding which to use on a system. While first-fit may be faster, best-fit provides better organization in the available pages.

  As previously mentioned, the system call implementation was also another learning opportunity. We were given first hand experience as how to make a system call that returns meaningful information about the machine. Pretty neat.