Project 3 report

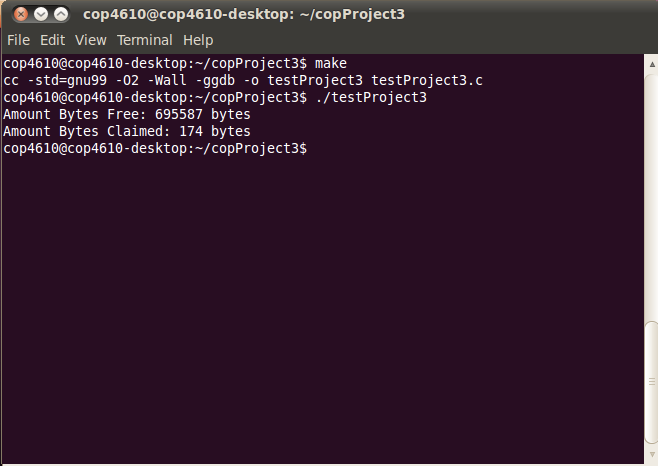
In project 3 we modified the file slob.c to use the best-fit algorithm to allocate a page and a block of memory, and implement two system calls to compare the degree of fragmentation between first-fit and best-fit.

First we implemented both system calls in the existing slob.c file to test the first-fit algorithm and followed the instructions in project 2 to do so . To do this we created two arrays, one for the amount of memory claimed by the slob allocator and the other one for the amount of memory in the list of free pages. We used a flag to check if we are iterating through the free\_slob\_small list. To test the system calls we wrote the program mycall\_test.c which allocates different amount of memory using the function malloc() and then prints the values returned by the system calls.

From the values that we got from the system calls we can arrive at the conclusion that the first-fit algorithm is better than best-fit for small memory allocation. The best-fit algorithm ends up leaving tiny spaces of memory that will never be used causing fragmentation. Best-fit is also slower than first-fit since it has to search through the entire list to find the best candidate.

To implement the best fit algorithm we first modified the function slob\_alloc(). We decided that to find the best page we need to keep the amount of memory that would remain in the page if we decide to choose it for the allocation. To find the amount of free memory remaining after the allocation we decided to implement a helper function slob\_check\_page(). This function works exactly the same as slob\_page\_alloc() but instead of allocating the block, it returns the amount of remaining if we decide to choose that page. So while we iterate through the list we need to call slob\_check\_page() and compare it to the best page yet and keep the one with the smallest remaining memory.

This one is using the first-fit algorithm



and this one is for best fit