HOMEWORK 4

Memory Allocation

Issued	2/18/14
Due Date	~~2/27/14~~ 3/13/14 **before class**

NOTE: As a reminder, homework is handed in when it is committed to your personal subversion repository before the deadline. Changes made after the deadline will be ignored.

Goal

One basic service each operating system provides is memory allocation. In this homework, we will be developing code that will be used to:

- allocate pages for use by user space programs and our kernel
- implement `malloc` - like functionality, for both our kernel and user space programs.

The focus is on the algorithms to do this; No kernel/virtual memory/page mapping/etc. knowledge is needed for this homework.

What to hand in

As for all the other homework assignments, you will need to commit your homework to your assigned subversion repository, under the correct directory (`homework/homework4`).

Please commit the following files:

- The kernel page allocator
- The userspace/kernel object allocator
- A README.page and README.object file which describes how you implemented your allocator.

Pay attention to the file names!

Task 1: Kernel Page Allocator

On the x86 architecture, the Memory Management Unit (MMU) has a concept of a page (typically 4K). A page is the smallest granularity used to keep track of memory translations.

For example, this means that access control is done on a per-page basis, and that a page is either present for a user space program or not.

Because of this, we need a memory allocator which can keep track of which pages are free and which pages are in use. Even though this code will eventually be used in our kernel, the code itself does not depend on kernel structures or specifics, so you can develop it as a normal C program. **You do not need to worry about memory mapping or virtual memory**.

Task 2: Object Allocator

While the hardware might enforce the page concept, a typical program does not. Instead variable length memory regions are allocated (for example, enough memory to hold an array of integers).

For example, look at `malloc` (`man malloc.h`).

Your second task is to implement a malloc-like library for our operating system.

Note: The kernel will also need to be able, for its internal bookkeeping, to allocate variable size memory regions. We will be reusing the same code in the kernel as well, where possible. **This should not affect your design**.

Logistics

The shared repository (under solutions/homework4) contains example C code, indicating which functions need to be completed.

Evaluation

Your solution will be evaluated based on:

- Code documentation and clarity
  - So far, for most people, the documentation can be substantially improved!
- Clear description of your algorithm in the respective `README.xxx` files.
- Correctness.
- Speed (less important).