MyHeap

Deadline: September 28th, 2014

1 Instructions

Implement your own version of C's `malloc`, `realloc` and `free` using only `mmap` and `munmap`.

Function	Description
`my_malloc(x)`	Allocate `x` bytes of memory and returns a pointer to the allocated memory. Return 0 if no memory could be allocated. The return address must be sufficiently aligned for C's `long` and `void*` data types.
`my_free(p)`	Deallocate the allocation pointed to by `p`, or do nothing if `p` is 0.
`my_realloc(p, n)`	Reallocate the memory pointed to by `p` to fit `n` bytes of memory instead. Return a pointer to the reallocated memory, or 0 if the reallocation failed.

- each function must be implemented in a `.c` file of its own, named after the function it contains. The function prototypes must be declared in a `.h` file, in accordance with the C coding standard. The submitted archive may (but needs not) include a test program. You must include a `Makefile` which properly places the functions in `libminic.a`.
- you may assume a page size of 8192 bytes.
- the only standard/system header you may include are `<stddef.h>` and `<sys/mman.h>`; the only external functions you may use are `mmap` and `munmap` (you may use functions from a previous assignment, by including their source in your submission).

2 Grading

- 6 points if `my_malloc`/`my_free`/`my_realloc` work as described. Tip: correctness > performance.
- +1 point if the implementation makes a best effort to fit multiple small objects in a page.
- +0.5 point if `my_malloc` attempts to reuse memory that was most recently freed.
- +1 point if the implementation makes a best effort at releasing pages to the operating system.
- +0.5 point if the implementation determines the page size automatically (using a portable mixture of `getpagesize` and `sysconf`; for this extra grade you may include `<unistd.h>`).
- +1 point if the implementation provides its own wrappers for system calls.