Global Section

Carmi Merimovich

This is your final mission. You are to implement a full scale global section support in the kernel. That means:

1. There should be up to `NGlobalSections` in the kernel.
2. The following system calls should be supported. The `name` and the `id` returned should be in the same way semaphores and files were implemented.
   (a) `int gs_create(char *name, int size)`: This system call allocates `size` bytes for the global section. It returns the local `id` of the section.
   (b) `int gs_open(char *name)`: This system call inicates we plan to use the global section. It returns the local id for success and -1 for failure. If the global section does not exist then the system call fails. Note that a process can call `gs_open` many times without intervening `gs_close`.
   (c) `int gs_close(int id)`: This system call inicates we are no longer interested in the global section. It returns 0 for success and -1 for failure. If the global section exists but we did not open it or create it, the system call fails. If after a close no one in interested in the section, then it should be free.

Implement the above three system calls. A good way to debug is to add `cprintf`s in the `kalloc` and `kfree` routine and checking that the pages alllocated are freed by following the addresses of the pages.

In order for a process to use the global section, the address allocated should be `map`'ed to the process address space. A successful mapping returns an $\widehat{\text{id}}$, in the same way as a successful `open` return `fd`. Let `NOMAP` be the number of mappings allowed to a process.

1. `int map(int sid, void *addr)`: The `map` system call search for an unused continuous virtual address range large enough to map the global section identified by `sid`. The return value identifies the newly created mapping. The

returned pointer is the process virtual address of the first bye of the global section. Each `map` call returned a different id (like in `open`, (and certainly a different addr!), until the maximum value allowed.

2. `int unmap(int id)`: If the `id` is legal the mapping is removed. Note the global section is NOT deleted. Just the mapping to it is to be removed.

Write user mode programs demonstrating sharing of two global sections.