Kyle Banda

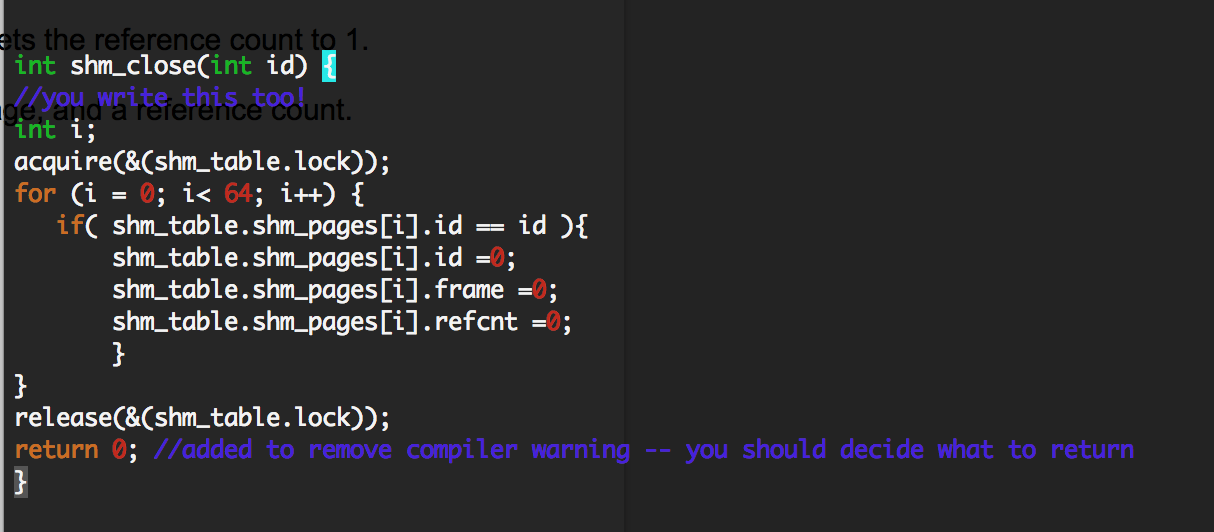
Marco Mercado

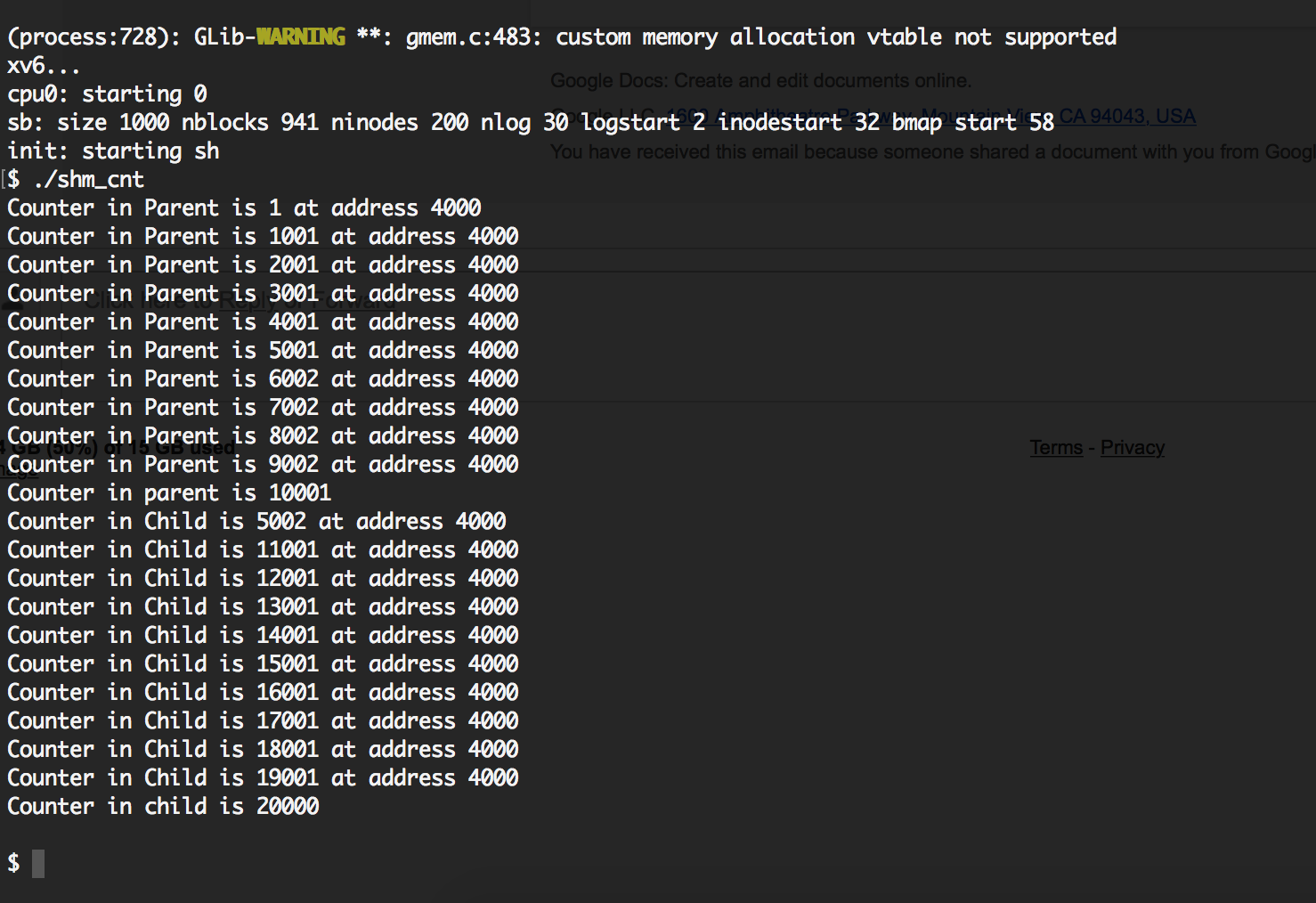
CS 153 Lab 4

Lab 4 was about creating functionality on xv6 that allows processes to share pages. This involves managing a shared page table that keeps track of what pages are being shared or can be shared between processes. We also have to make sure both page table entries for the processes that are sharing a page point to the same physical page. We had to code two major functions: shm\_open() and shm\_close.

Shm\_open: This function is passed an id number (int), and a pointer for storing the page location (char\*\*). It looks through the table to see if the id number matches any entry. If it does, it increases the reference count of that entry, and uses mappages to map the virtual and physical pages together for that given process . If it does not find the id number, then we have to find an empty slot in the shm table and then a page is allocated in physical memory, and with memset we set all the addresses to zero. Then map the phsycal page to the virtual page and give it permission so that the user could write with PTE\_U AND PTE\_W so that they can be access, and pointer to the page is store in shm\_table. It sets the reference count to 1. Lastly we return a pointer to the virutal page that was map with and increased the sz for the process because we added that entry page. Each shm\_table entry holds an id, a pointer to the physical page, and a reference count.



Shm\_close: This function is passed an id number. It looks for the id number in the shm\_table, and if it finds it, then we clear all the shm table so that it can be reused .This is according to the TA that said we are supposed to just clear it off the table once one of the process that has access calls shm\_close(id) with the given id for a given page. 

The test file was provided by the Professor this are the results.

So the process counting a variable that they share in the page that they share and it how it’s supposed to

Other Important functions:

Mappages(pgdir, va, size, pa, perm): Mappages maps a virtual address to a physical address. It returns 0 if it runs correctly.

1. Pgdir -> page directory of current process
2. Va -> virtual address
3. Size -> page size
4. Pa -> physical address
5. Perm -> Permissions

Memset: we use this function after allocating a new page. It sets all of the memory in a passed in address to a given parameter. In this case we set the newly allocated page to all 0s so it could be written to properly.