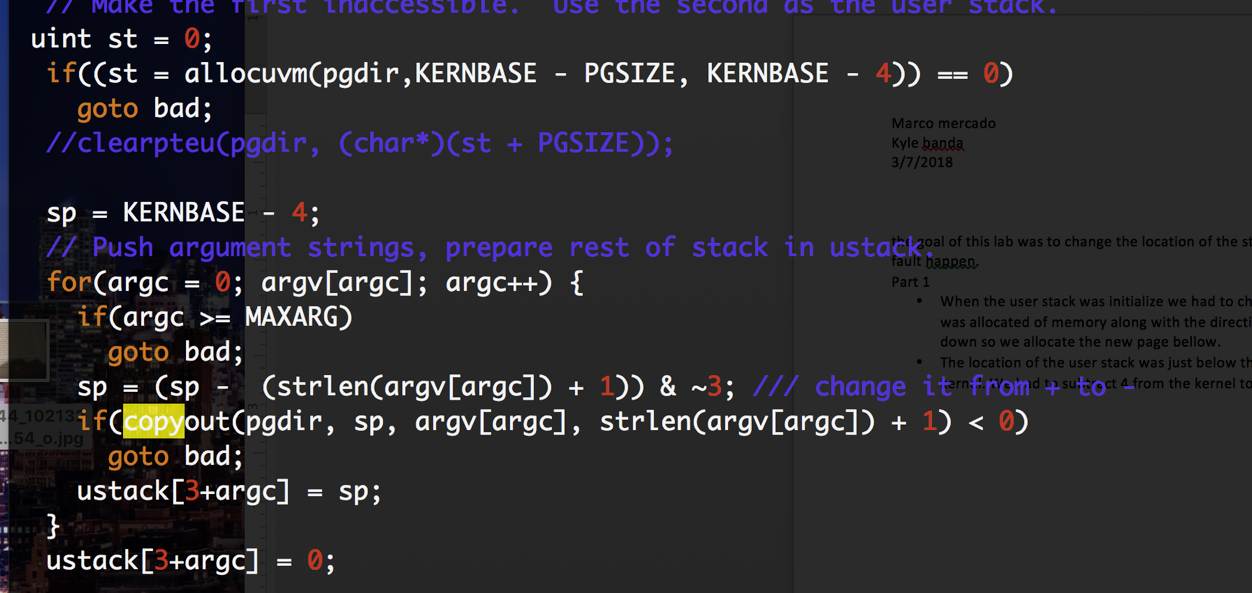
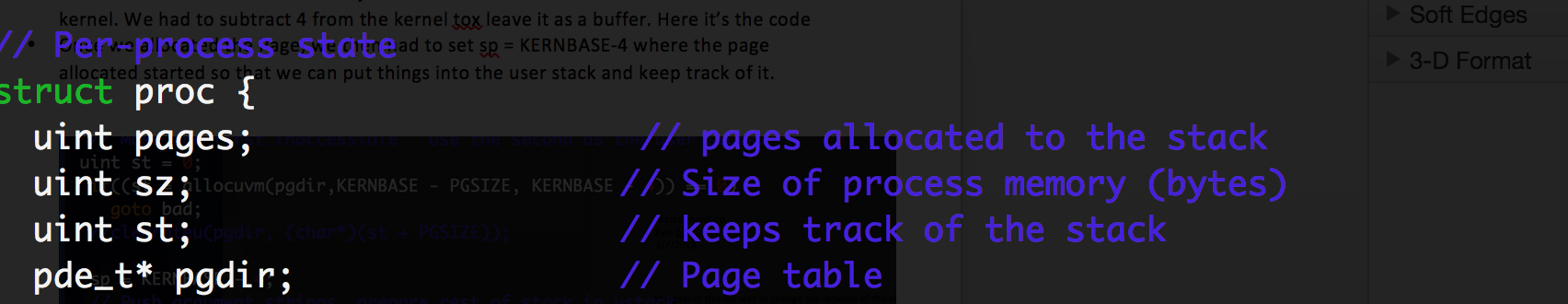
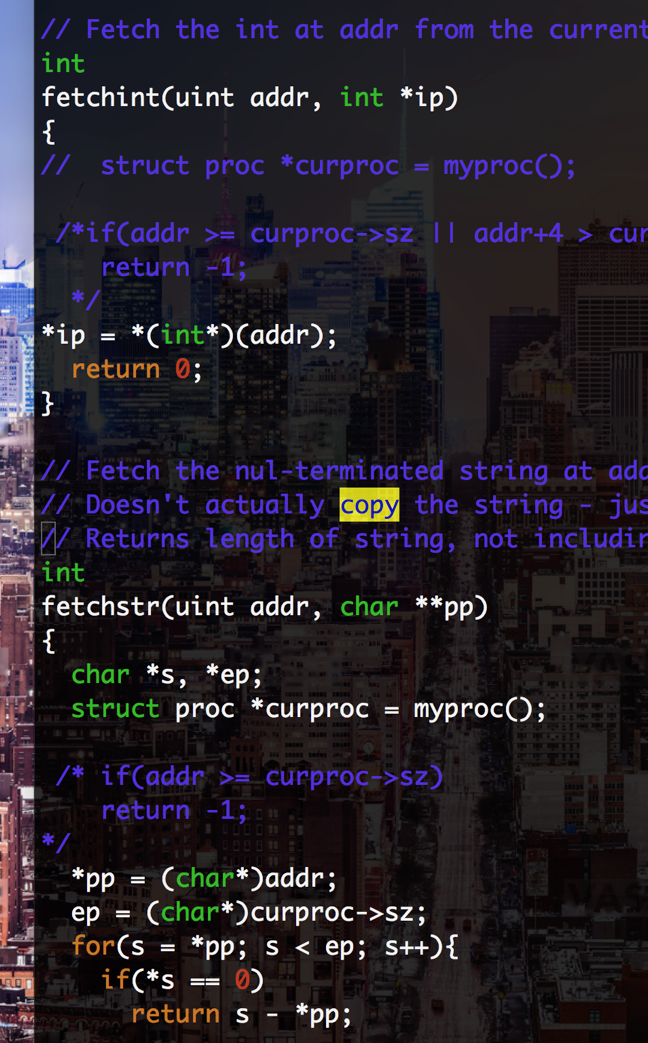
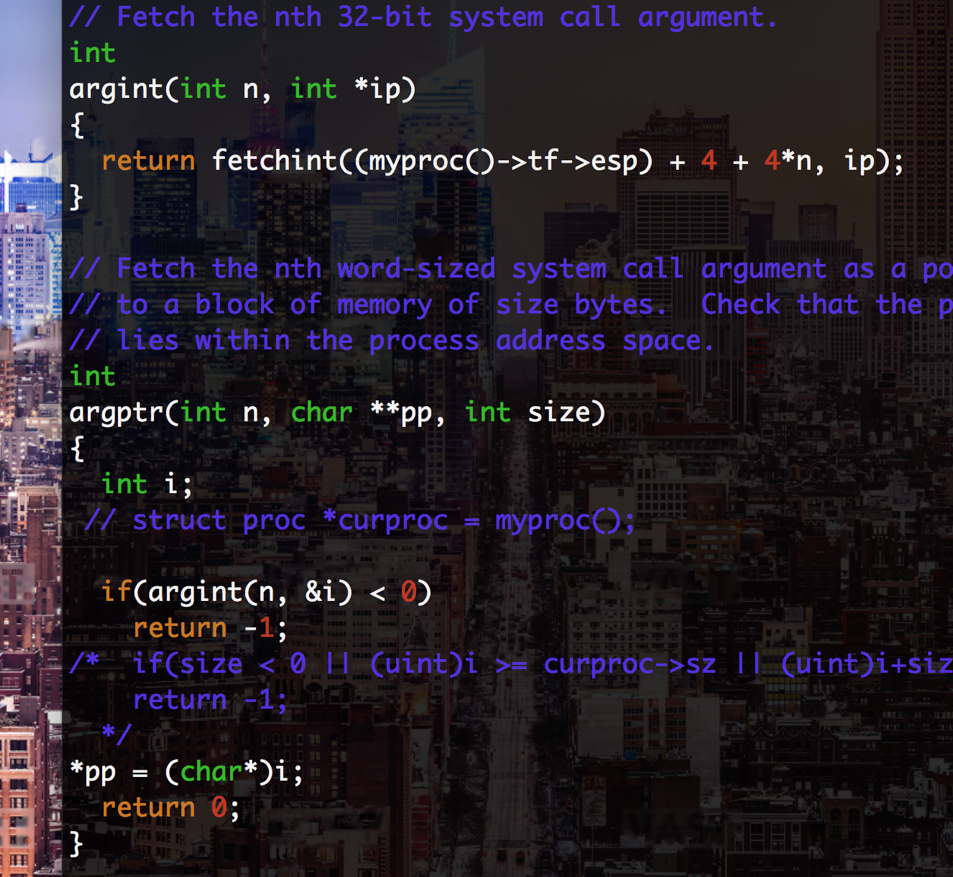
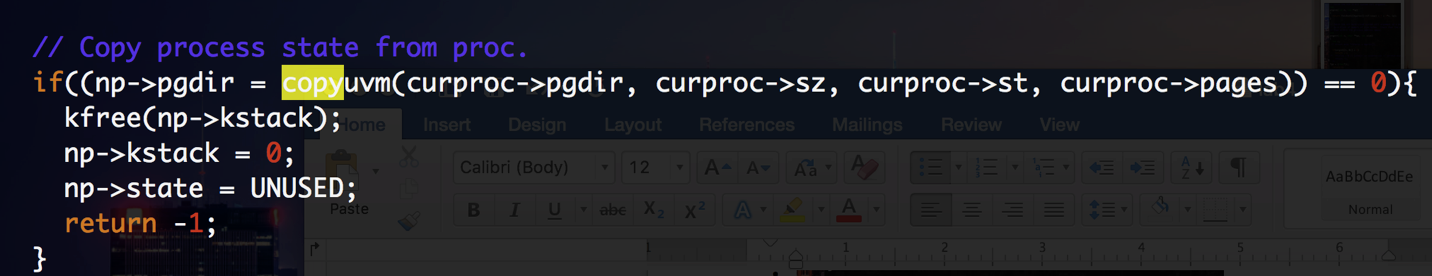
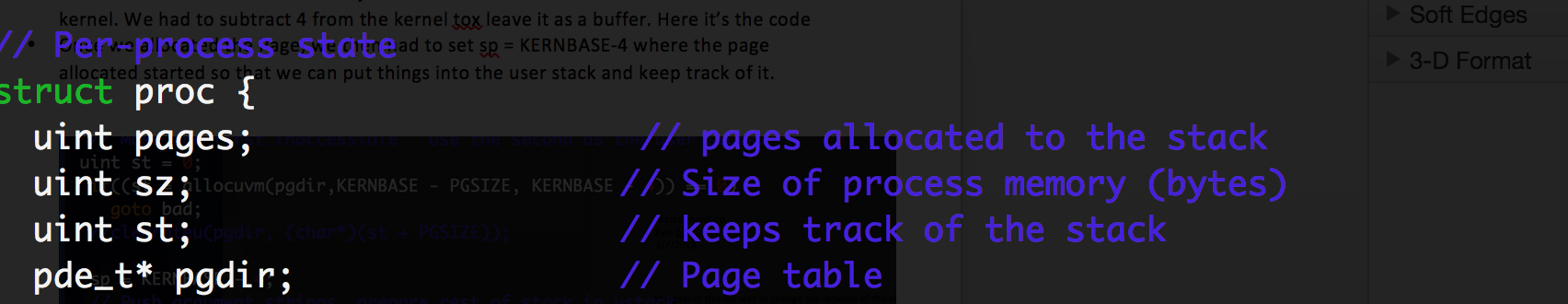
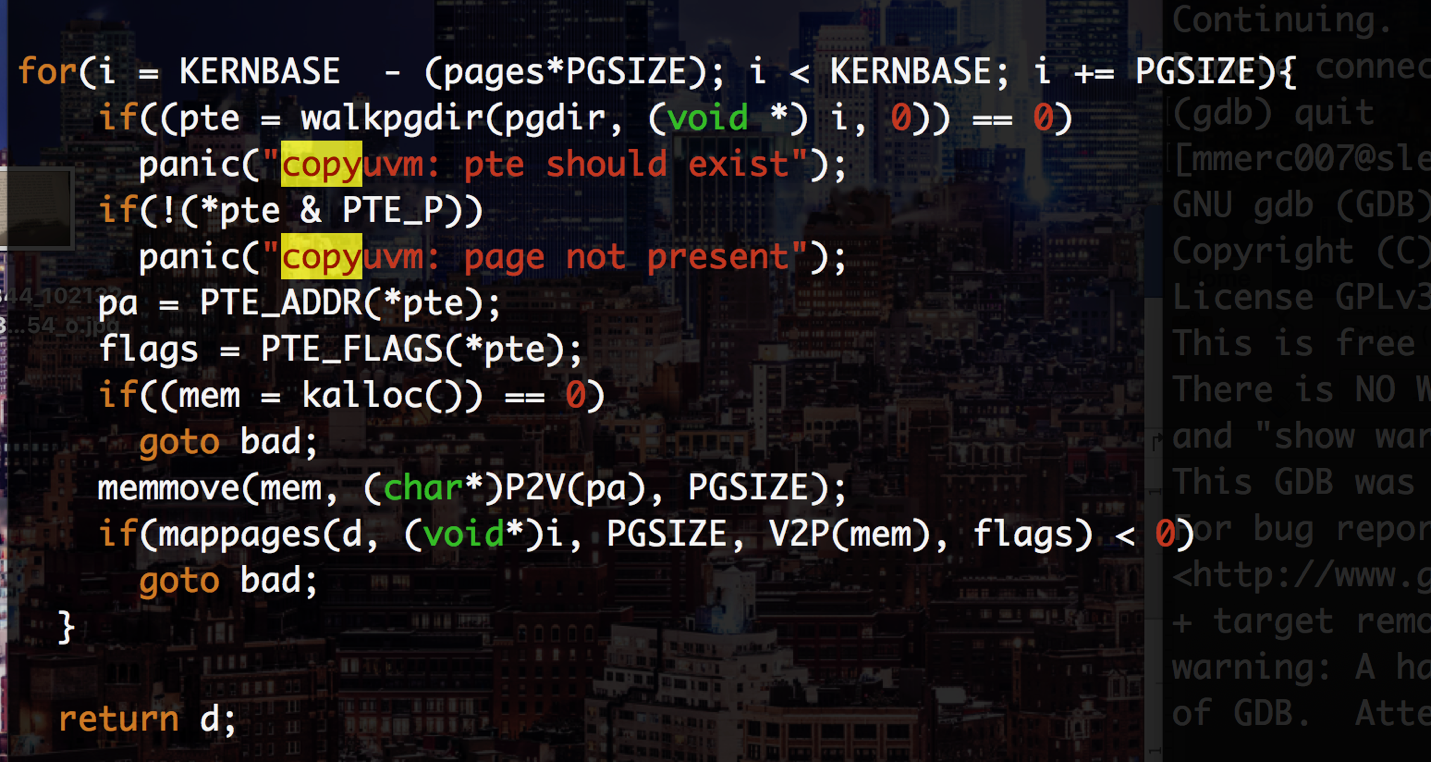
Marco mercado

Kyle banda

the goal of this lab was to change the location of the stack and the way it grew when a page fault happen.

Part 1

* When the user stack was initialize we had to change the location of where the page was allocated of memory along with the direction that it went. So now the stacks grows down so we allocate the new page bellow.
* The location of the user stack was just below the KERNBASE or the last line of the kernel. We had to subtract 4 from the kernel tox leave it as a buffer. Here it’s the code
* Once we allocated the Page, we then had to set sp = KERNBASE-4 where the page allocated started so that we can put things into the user stack and keep track of it.
* 
* I also added extra variables to the proc struct to keep track of the stack end and the number of pages in it. St keep track of the end of the user stack and the pages variable of the number of pages in it.
* 
* we then had to removed the checks in syscall.c so that it wouldn’t give any errors since we move the user stack. We choose to remove them rather than change them.
* 
* we had to change the arguments for copyuvm(), we added two, one for st, and one for the number of pages in the stack.
* 
* then we had to change the contents of copyuvm so that it also copys the user stack that’s in another place. We had to add another loop and copy all the usertack so that when we fork the child has a copy. This is in vm.c we also had to change where copyuvm was define which was in defs.h, we change the aguements.
* 
* 

part 2

* we added a case for when a page fault happens. We check if he RCR2() is lower the the end of the user stack if that’s true then we allocated a new page. And move on other wise we do nothing. We had to determine the conditions for the if statement and go base on the last page that of the user stack to grow it. Then added one to keep track of the number of pages
* 