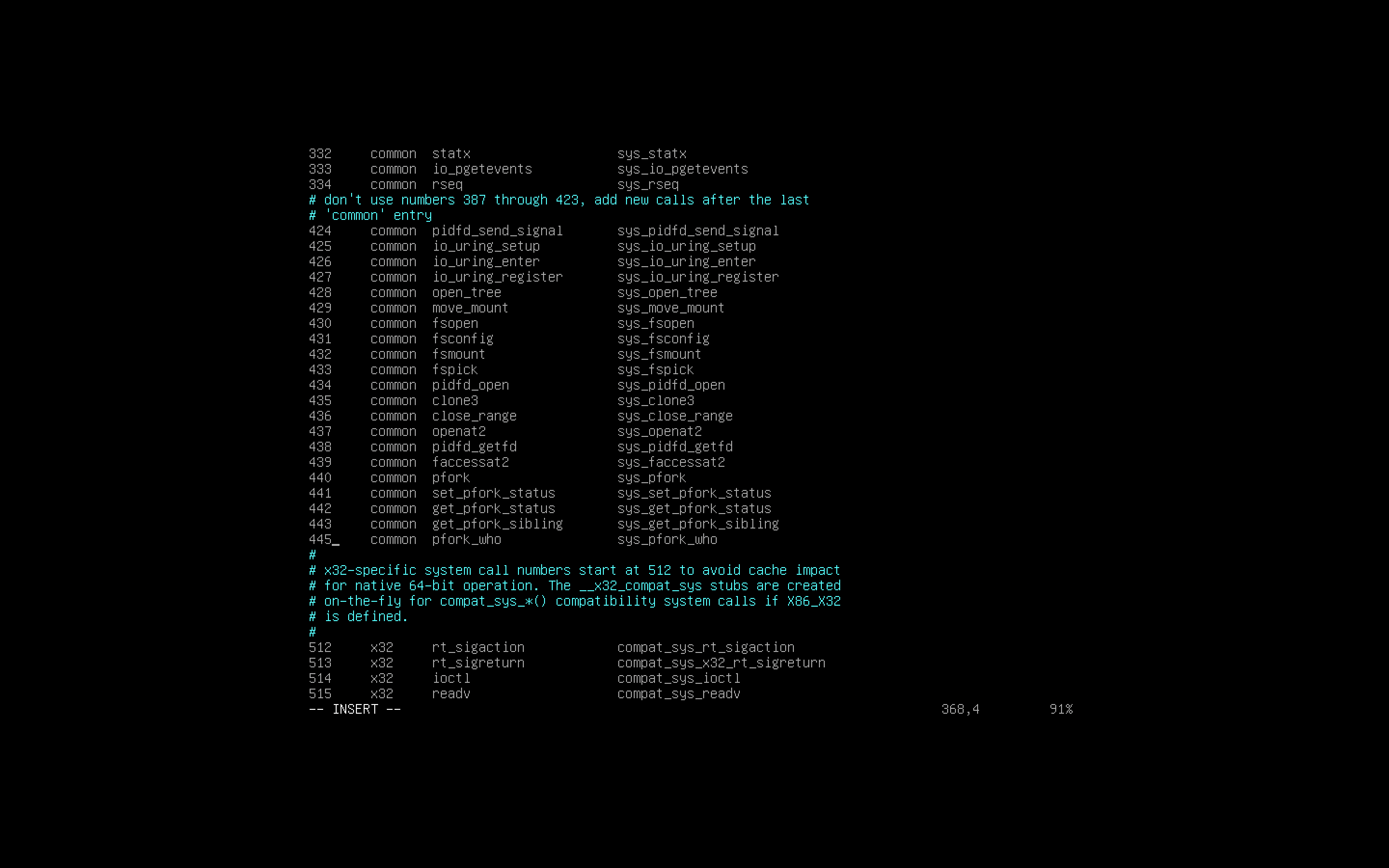


**OS lab**

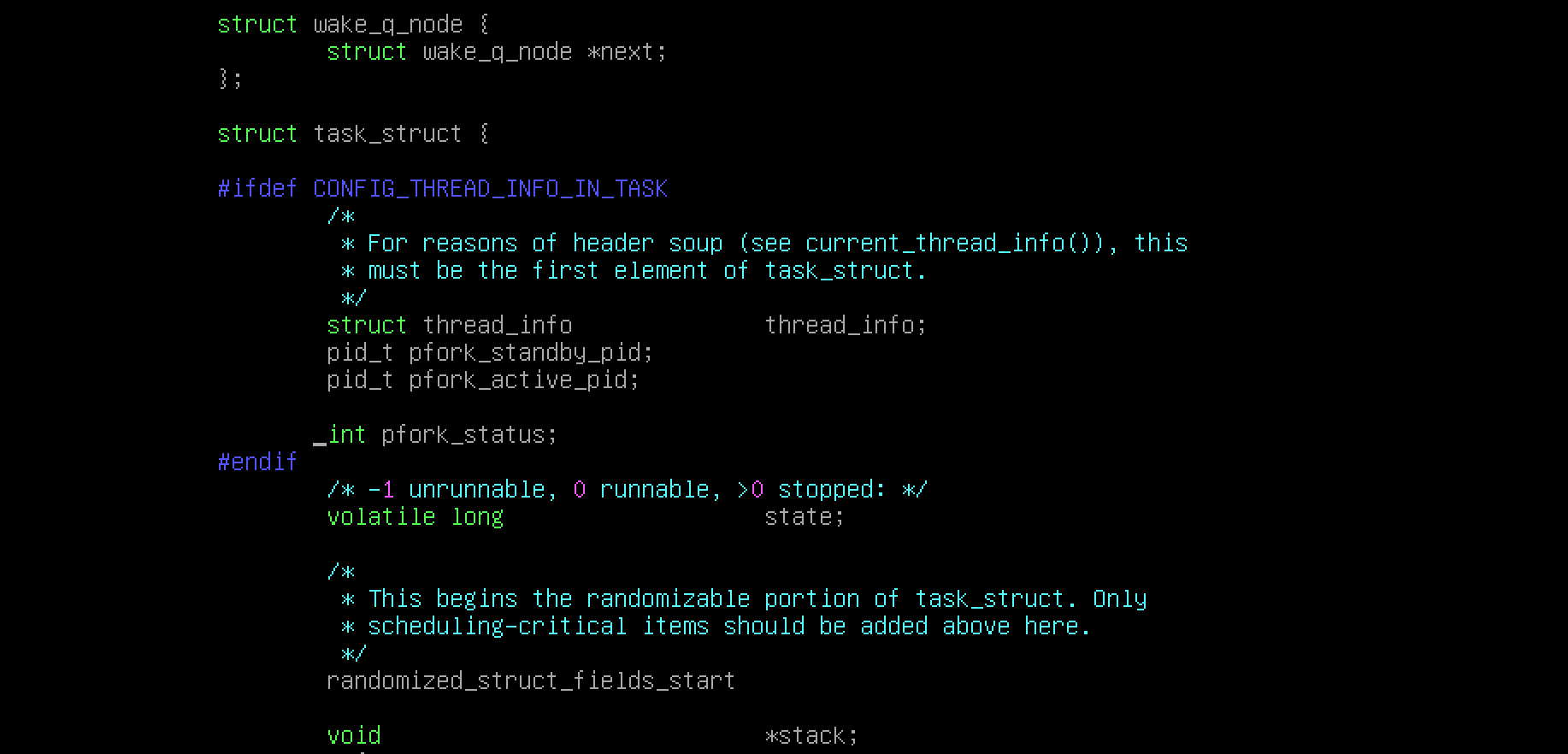
Andrew Nady

900184042

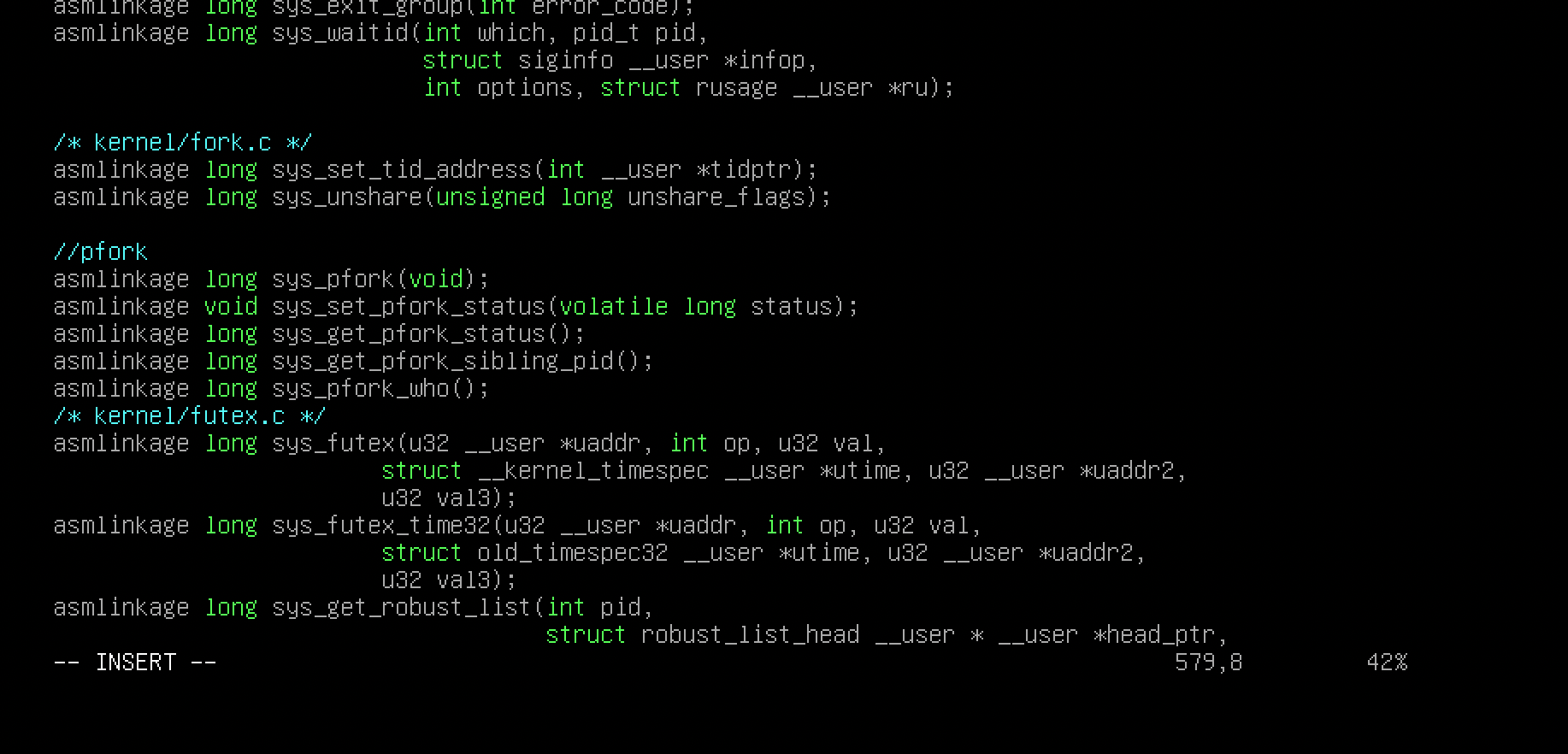
* Creating System calls and wrappers to them in syscall\_64.tbl file as following



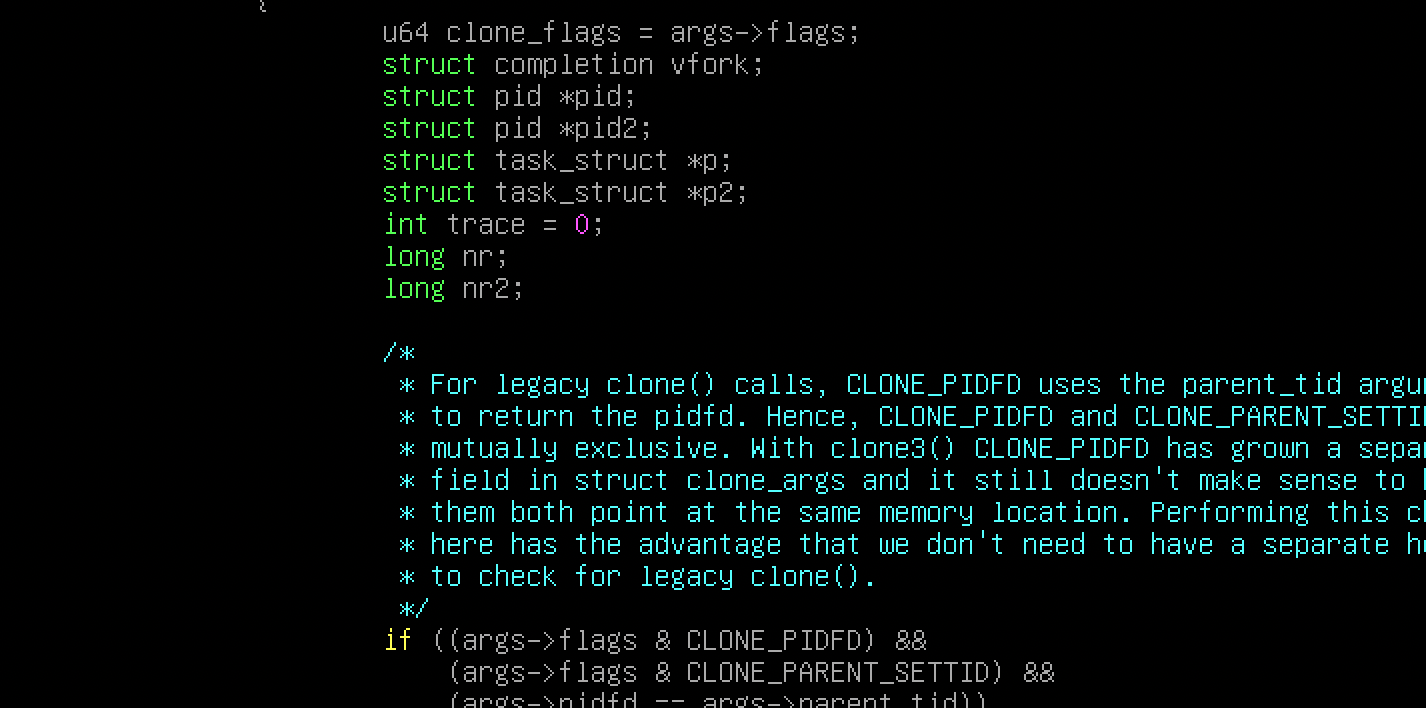
* Second, we need to add the variable to the task struct, these variables are the status and standby process id and the active id. So, we can store them in the this struct. This is in Sched.h

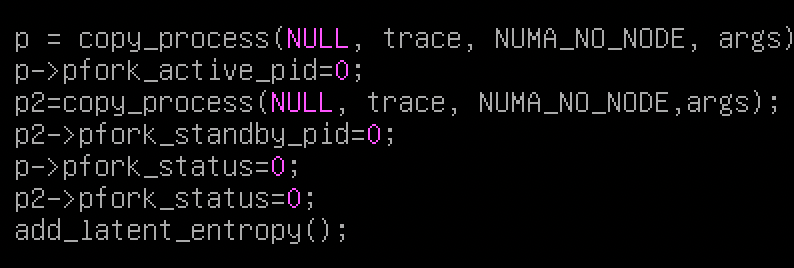


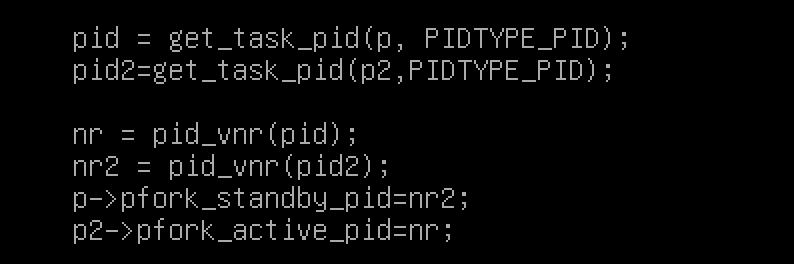
* Thirdly, we need to add asmlinkage to our new system calls, so we will add these linkages in the syscalls.h



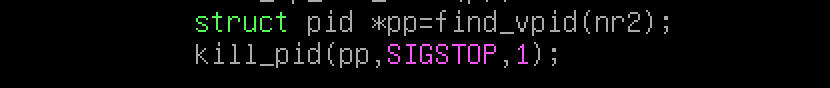
* Then, we need to create new function for the pfork which is somehow similar to \_do\_fork. We will modify on it to add another pid and another nr and another process to our variables. (In Fork.c)



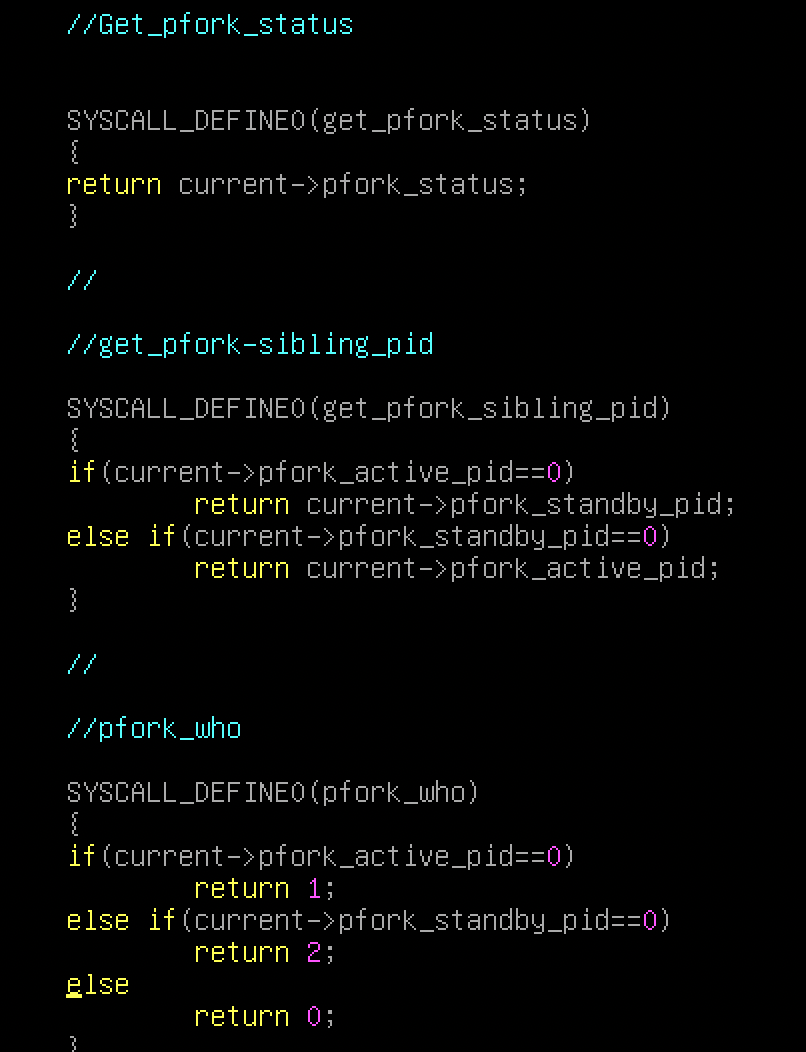
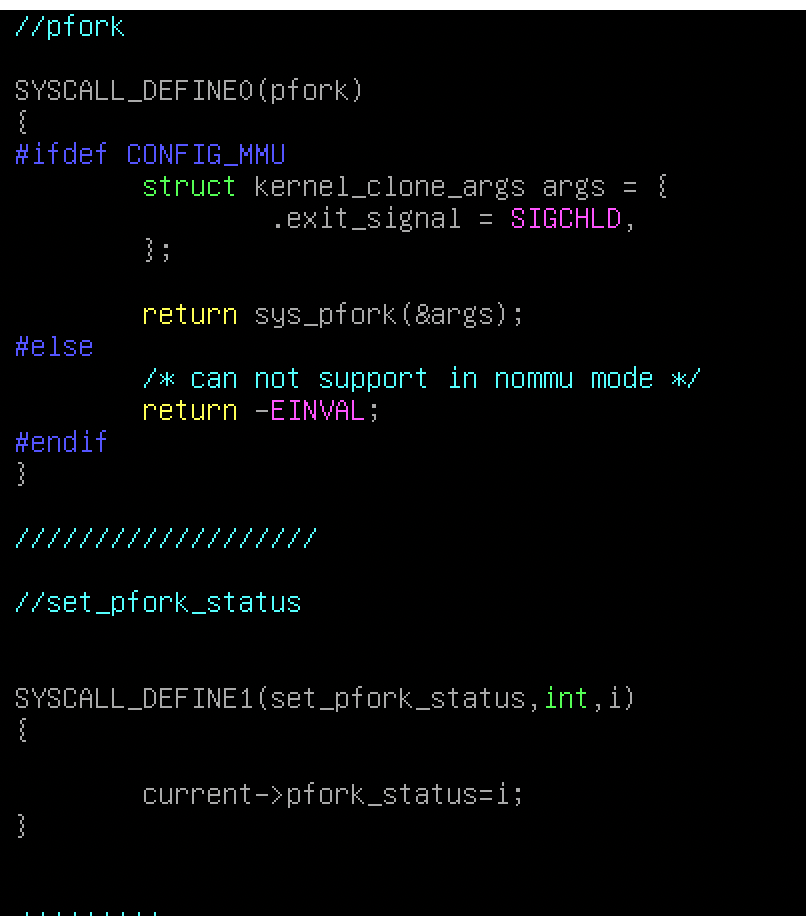
* Then, we need to call copy\_process another time and give the value to p2(which indicates another process), also we need to initialize the active id of the first process by zero and the standby id of second by zero. We need to initialize the status of the two processes by zero. (In Fork.c)
* Then get the id of the second process and give it to pid2, we also need to do with the virtual id. Moreover, we need to give the standby id of the first process the nr (virtual id) of the second process and the active id of second process should be initialize by nr of the first process. (In Fork.c)



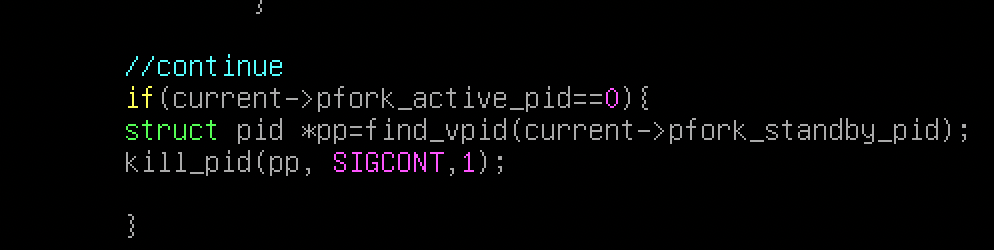
* Also, we need to stop the second process till the first finishes, so we use kill\_pid where it takes the SIGSTOP which put the process in the waiting stage



* Then, we need to define the system calls, where SYSCALL\_DEFINEX to define a system call, where x represents the number of parameters to the system call. (In Fork.c)



* Finally, in the exit.c, we need to make the other process to return back from the waiting stage to the active stage.



After a lot of trials and errors, finally my new version of kernel compiled successfully. And now I have a new entry for the new kernel

