Design document

In this project, the aim was to create the pfork system call along with other complimenting functions. To achieve this, multiple modifications were made to the kernel in order to make it work.

Fork.c

In this file multiple additions were made using define0 and define 1:

Pfork():

- In this system call, what it essentially did is that it called the _do_fork() 2 times, instead of only once in the fork() system call.
- Each _do_fork() made is expected to return a nr pid of the process activated.
- I use these nr pids to find the struct using the find_get_task_by_vpid(the returned), which I specify one to be the parent, and one to be the child.
- I specify the who of the parent to be 1 & the who of the child to be 2
- After this, I use the kill_pid and the SIGSTOP in order to stop the child from executing, this should put it into the wait queue.
- The stopped process will only continue when the parent process exits (which is modified in exit.c)
- Also, I put the pid of the child in the parents pfork_standby_pid, and the pid of the parent in the pfork_active_pid
- I give the parent a who value of 1, and the child a who value of 2 (since I considered that the parent will be the active and child will be the standby)
- Finally, I suspend the child process using the kill_pid and SIGSTOP signal, so that only the parent continues to execute (until it exits).

Pfork_who():

- This is just a basic syscall_define0 that returns the who value of the current

Get_pfork_sibling_pid():

- This is another basic syscall_define0 which checks the who value of current, which based on it it will return the standby_pid or active_pid

Get_pfork_status():

- A basic syscall_define0 that returns the pfork_status value of current

Set_pfork_status(long stat):

- This function sets the pfork_status of current
- If the who is 1, then it also sets the status of the standby task_struct to the same stat.

Finally, I modified the copy_process, such that it initializes the who and pfork_status of the p to 0, since this should be the default (maybe called from a normal fork)

Exit.c

- As mentioned earlier, I modified the exit of the active process, such that it signaled a continue for the standby process
- I did this by changing the do_group_exit, at the end before the do_exit, I added an if condition that checks if the who is ==1, if yes then it signals a kill_pid with SIGCONT to the standby task (I get the stanby task using the pfork_standby_pid stored in the current, and the find_vpid api)

Sched.h

- In the sched.h, I modified the task_struct so that I added new variables:
    - Pid_t pfork_standby_pid and pid_t pfork_active_pid, which are both set in the pfork syscall mentioned earlier
    - Long pfork_status
    - Long who (which is 0 default, 1 active pfork, 2 standby pfork)

To add the system calls:

- I added the asmlinkage of the 5 mentioned systemcalls in the include/linux/syscalls.h
- Also, I added them to the arch/x86/entry/syscalls/syscall_64.tbl, using numbers from 440 till 444.

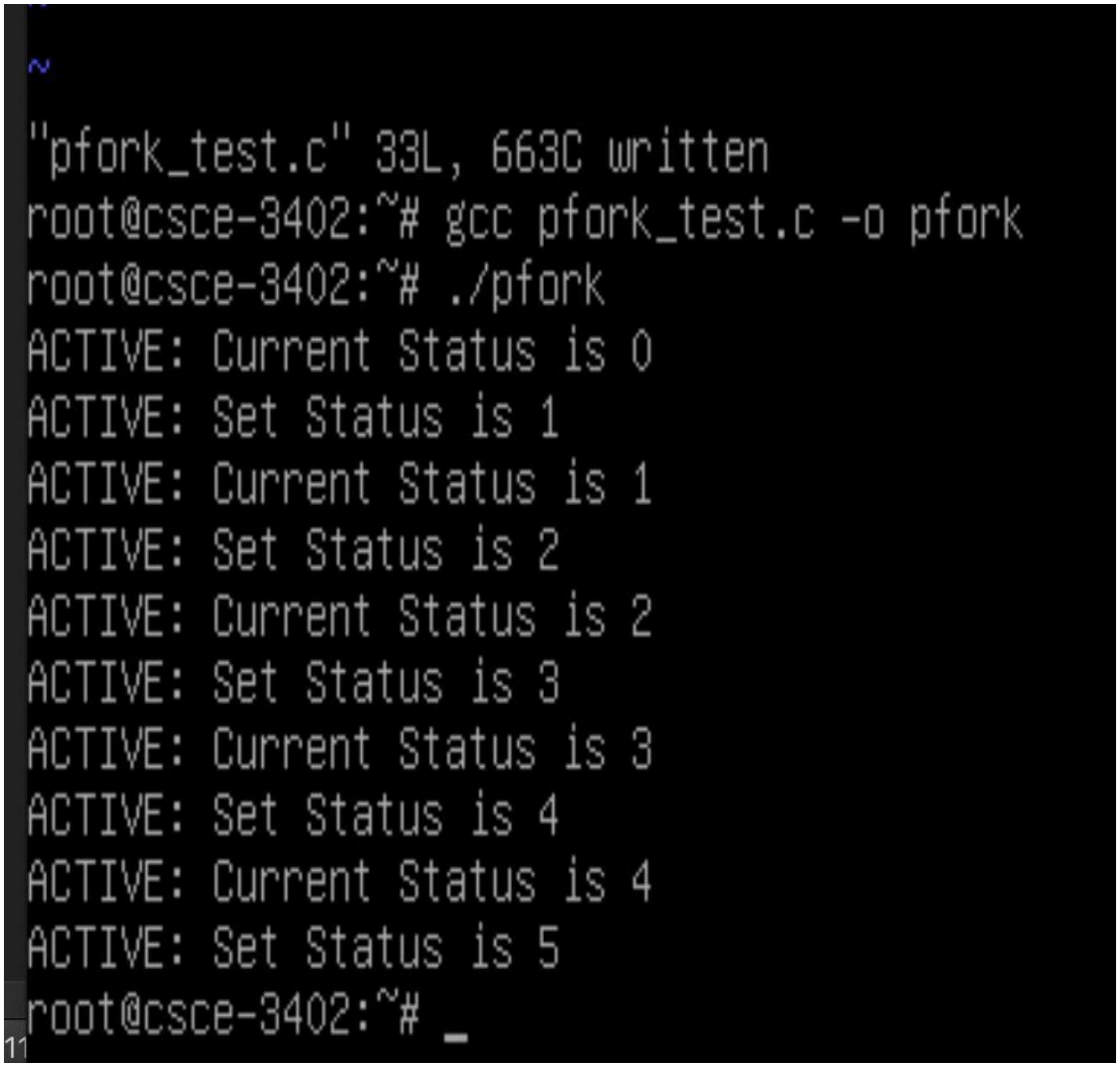
Pfork.h

- This was a wrapper to all the system calls, I inserted this file to the user/includes so that it can be imported to any user space test program.
- In this header file, I used the syscall() with the corresponding value to get the targeted system call.

Test cases:

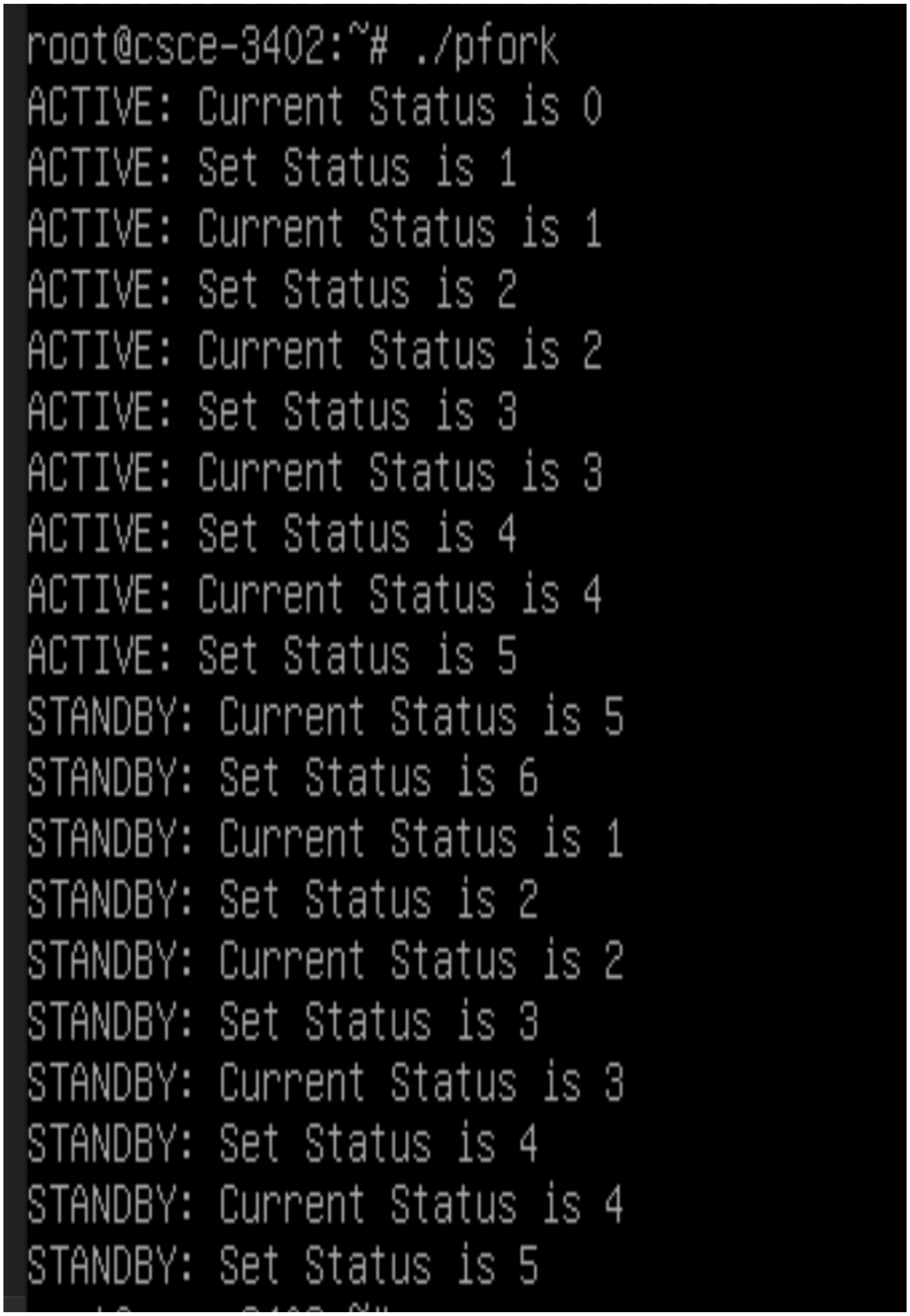
- I tried the provided testcase in the document, and it printed:
    - ACTIVE: current status is 0
    - ACTIVE: set status is 1
    - This repeated till 5
- I modified it so that it can also try the stanby to make sure that the STANDBY can still execute and it did (also 5 times after the active finished its loop).

Provided sample:



```
~
"pfork_test.c" 33L, 663C written
root@csce-3402:~# gcc pfork_test.c -o pfork
root@csce-3402:~# ./pfork
ACTIVE: Current Status is 0
ACTIVE: Set Status is 1
ACTIVE: Current Status is 1
ACTIVE: Set Status is 2
ACTIVE: Current Status is 2
ACTIVE: Set Status is 3
ACTIVE: Current Status is 3
ACTIVE: Set Status is 4
ACTIVE: Current Status is 4
ACTIVE: Set Status is 5
root@csce-3402:~# _
```

Modified sample:



```
root@csce-3402:~# ./pfork
ACTIVE: Current Status is 0
ACTIVE: Set Status is 1
ACTIVE: Current Status is 1
ACTIVE: Set Status is 2
ACTIVE: Current Status is 2
ACTIVE: Set Status is 3
ACTIVE: Current Status is 3
ACTIVE: Set Status is 4
ACTIVE: Current Status is 4
ACTIVE: Set Status is 5
STANDBY: Current Status is 5
STANDBY: Set Status is 6
STANDBY: Current Status is 1
STANDBY: Set Status is 2
STANDBY: Current Status is 2
STANDBY: Set Status is 3
STANDBY: Current Status is 3
STANDBY: Set Status is 4
STANDBY: Current Status is 4
STANDBY: Set Status is 5
```