Operating System

Lab Assignment 2

Synchronization in Linux Kernel

Name: ***Mohamed Samir Shabaan***

ID *: 55*

code organization:

the code consist of three class (mod.c & leds.c & user.c) .

leds.c : consist of main function and other function , the main function get input from user then call the other function to execute this instruction , the other function call the the function in class mod.c which included in the kernel .

Function in the mod.c execute this instruction and turn the caps on or off or get state

can user user.c and follow the steps that shown.

main functions:

leds.c:

main() : call the other function and determine if we in get instruction or set instruction.

Get_led(int led): using system to get the state store in led's file that store in /sys/ directory.

Set_Led (int led , int state): using system to set store in led’s file that store in /sys/ directory.

mod.c:

Store and Show : three function to read state stored in (num , caps or scroll)’s files or write new state in it .

set_led_state(int led,int state) :

to change the global varriable led_status_word and call update function to update new states

get_led_state(int led):

get led state on or off using global varriable led_status_word.

kbd_read_data(void):

check the output buffer of full read from it other wise wait to be full.

kbd_write_data(unsigned char data):

check the input buffer if clear write data using outb() other wise wait to be clear.

update_leds(unsigned char led_status_word):

update new states and call kbd_read_data() and kbd_write_data().

how to compile and run the code:

-Open the terminal and go to the project directory.

- write make in terminal.

- go to the file director by write

cd /sys/kernel/kobject.example in terminal then make the file writable by write :

> sudo chmod 777 caps

> sudo chmod 777 num

> sudo chmod 777 scroll

- then go to last director and using ./leds to turn on and off led such that :

> ./leds set caps on

> ./leds set caps off

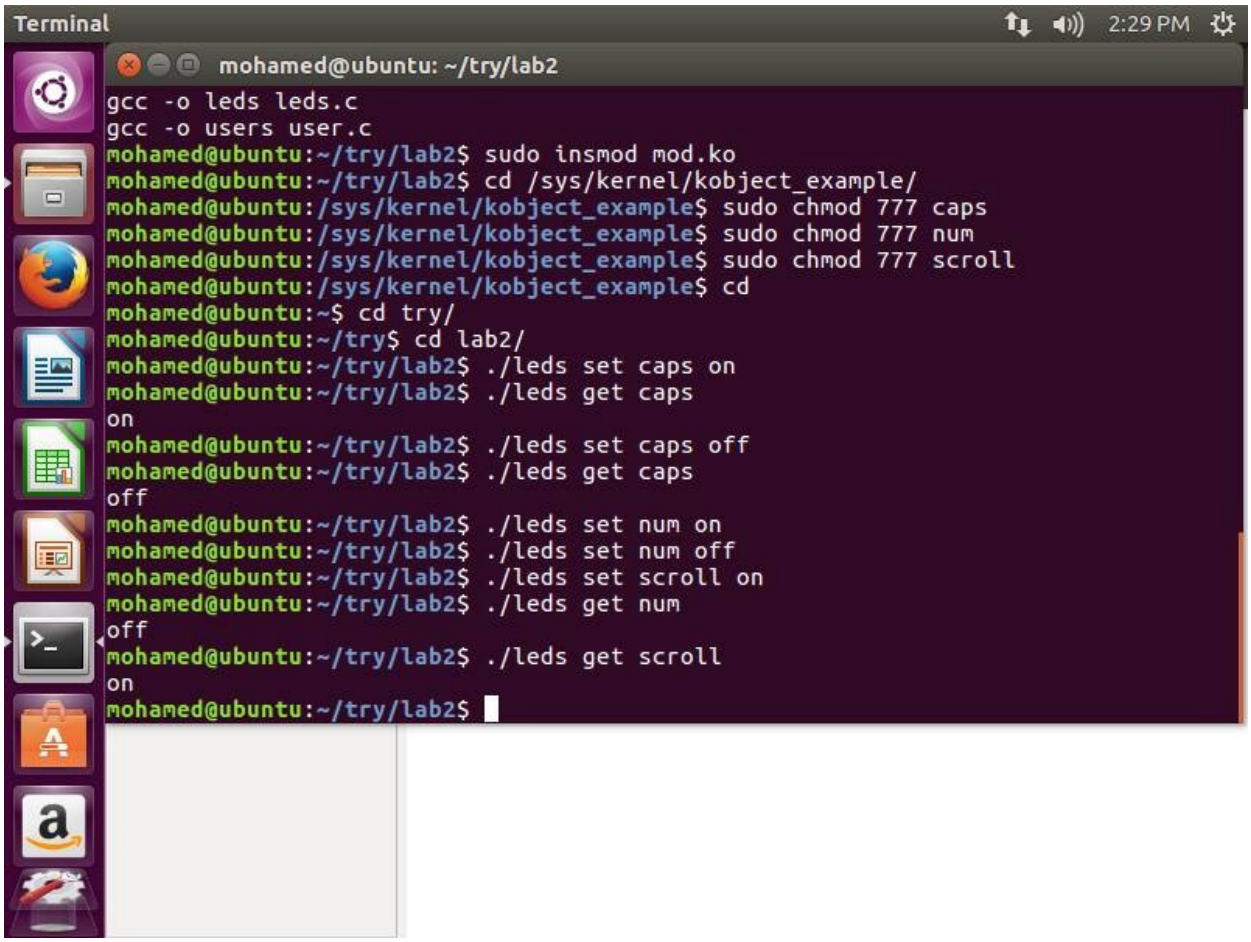
> ./led get caps

and so on.

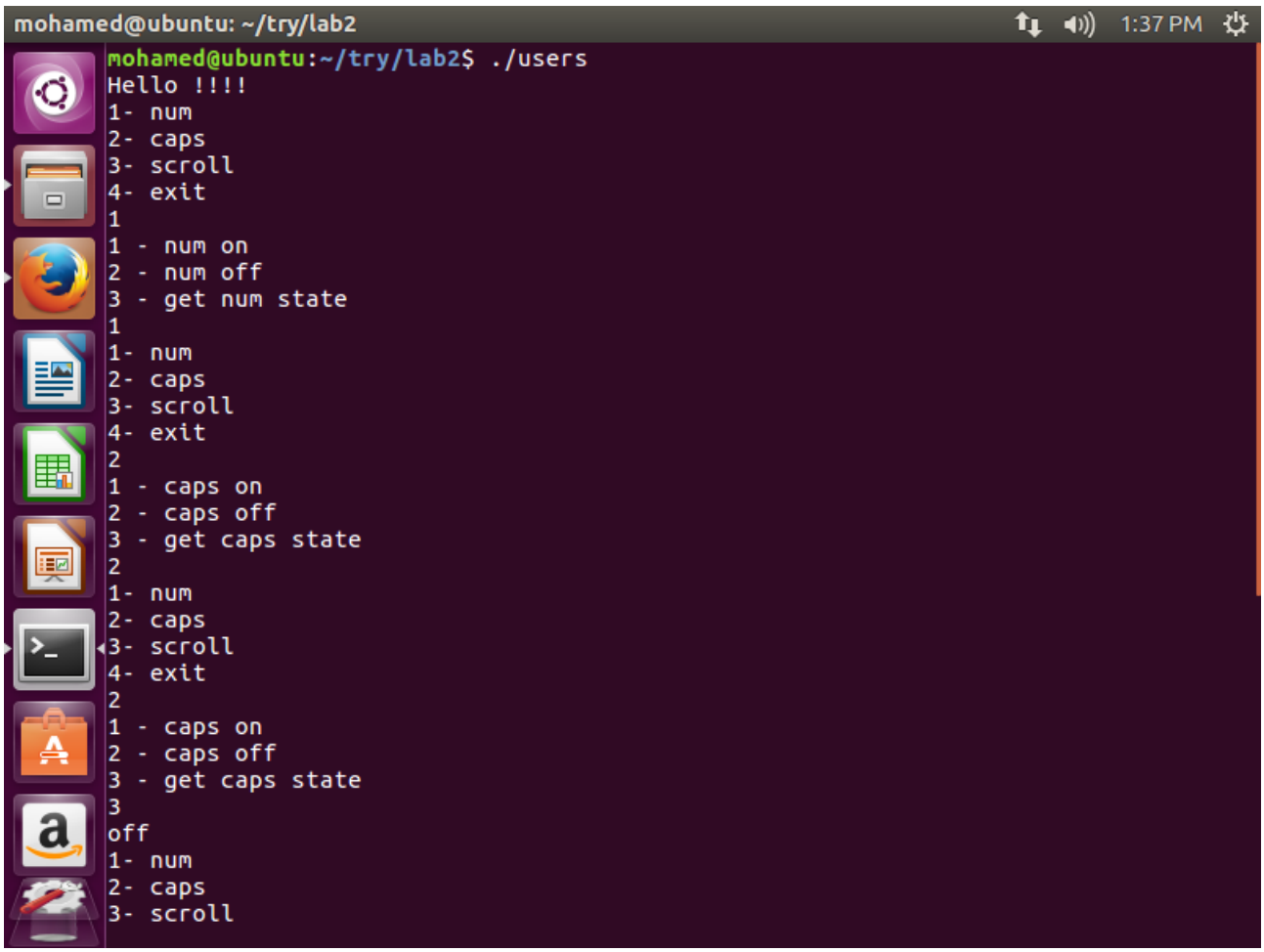
– you can user ./users that create in the director and follow the steps that shown .

Sample runs:

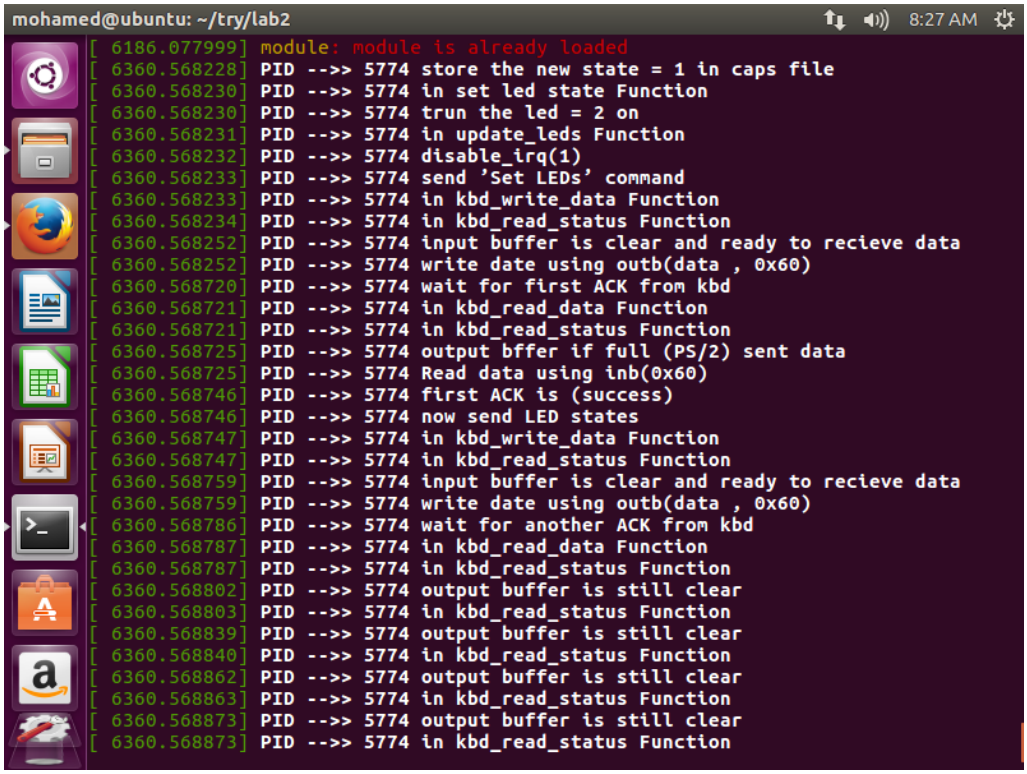
using ./leds:



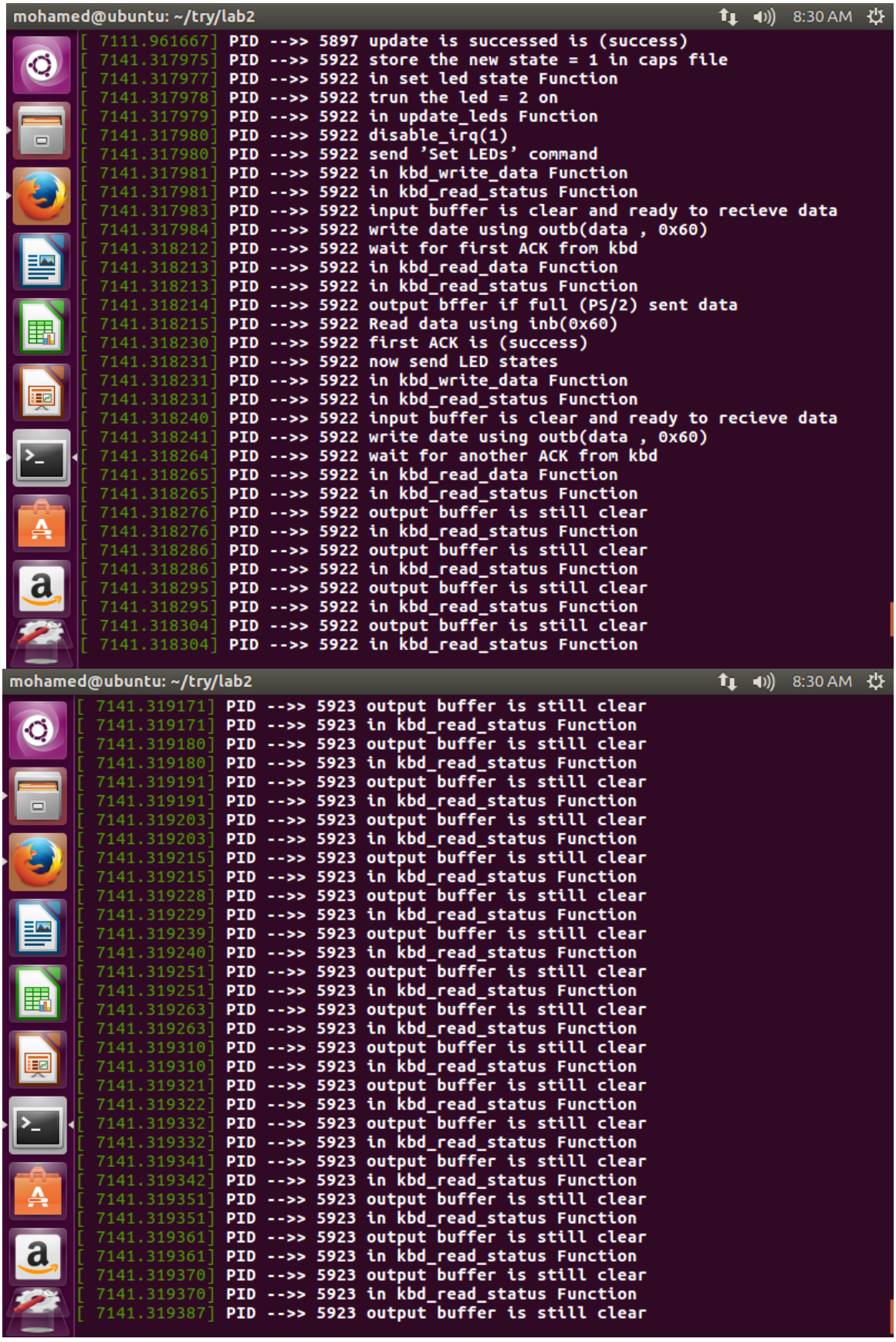
using ./users:



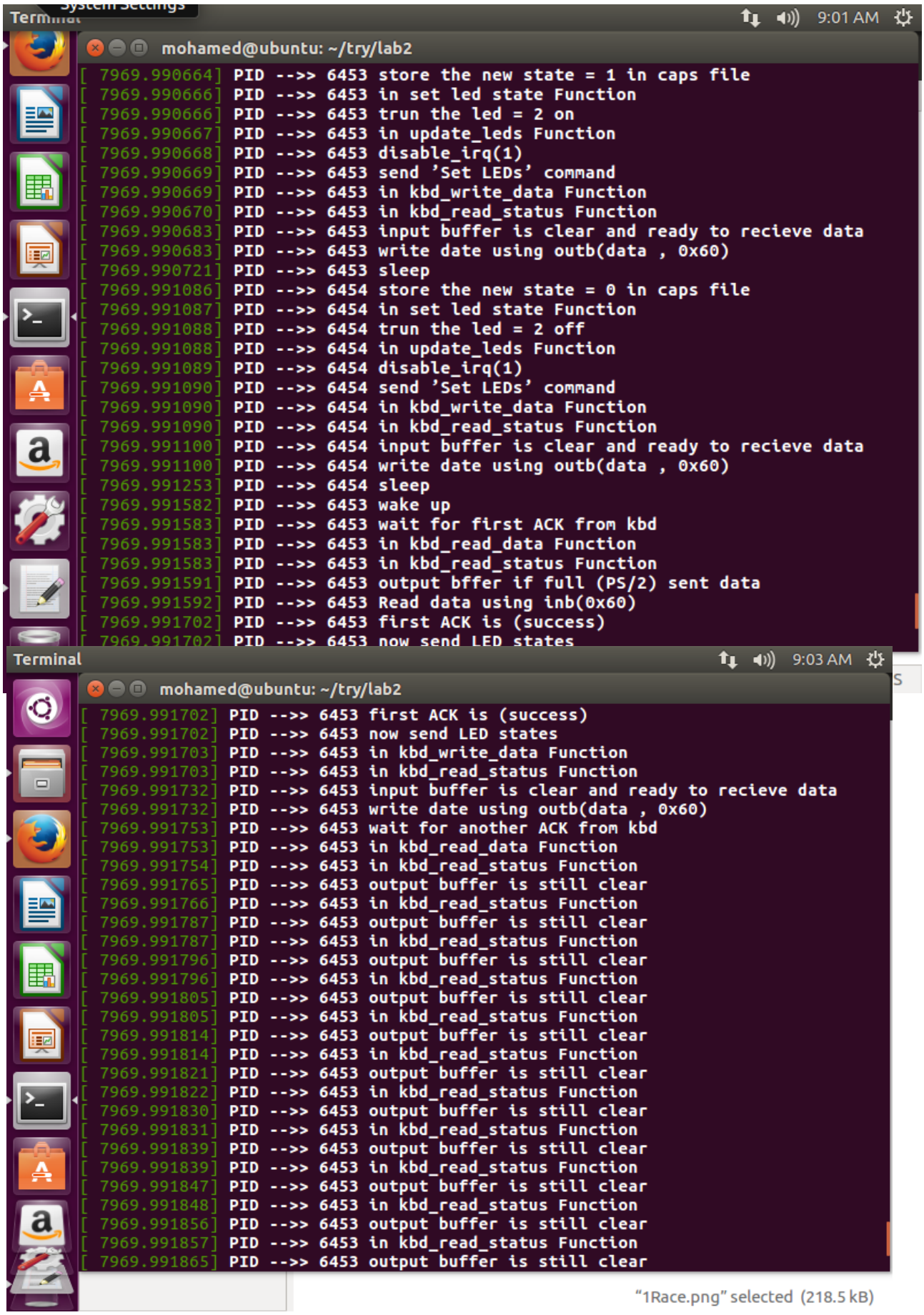
test printk in one process:



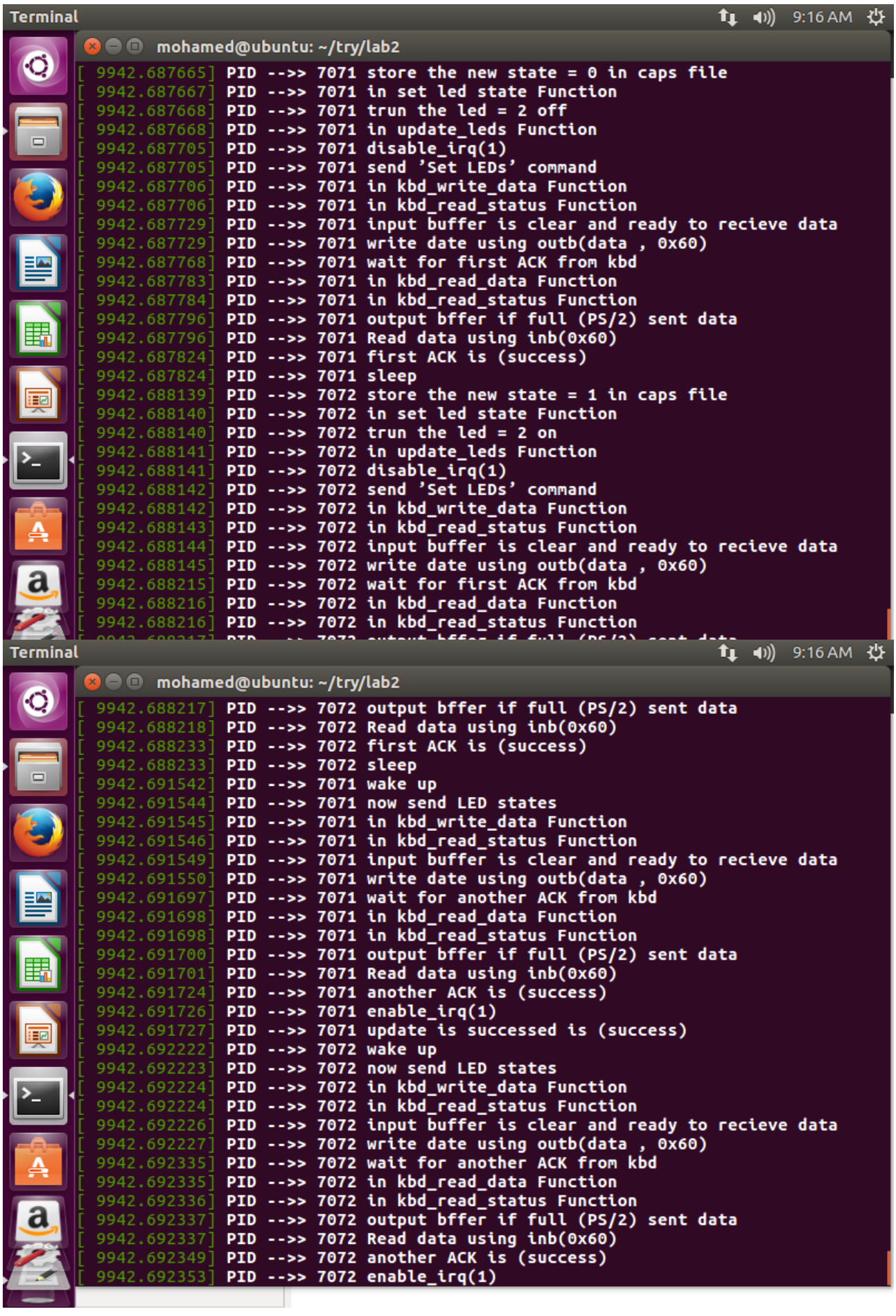
test two process but with out sleep:



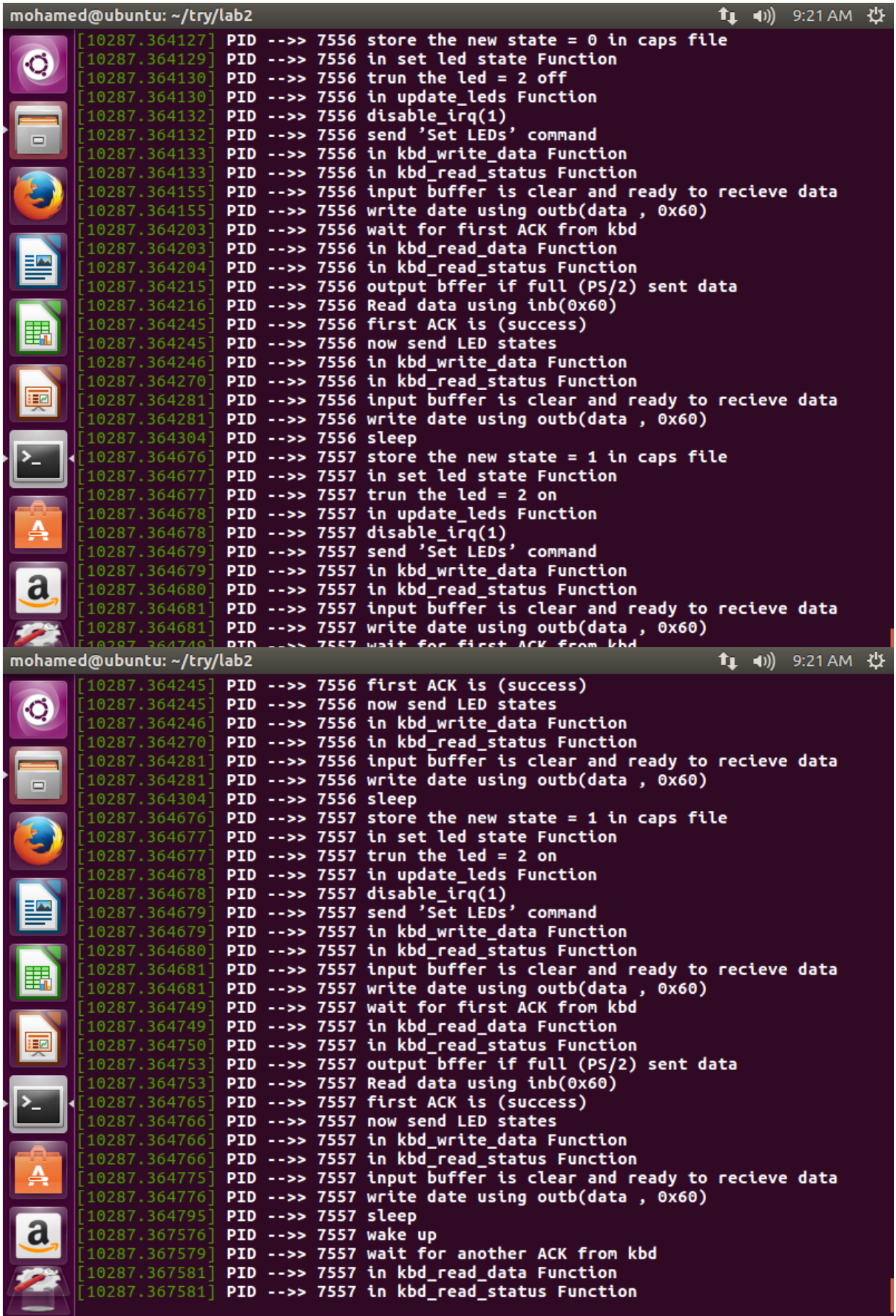
test race condtions 1 as shown in code(send 0xED):



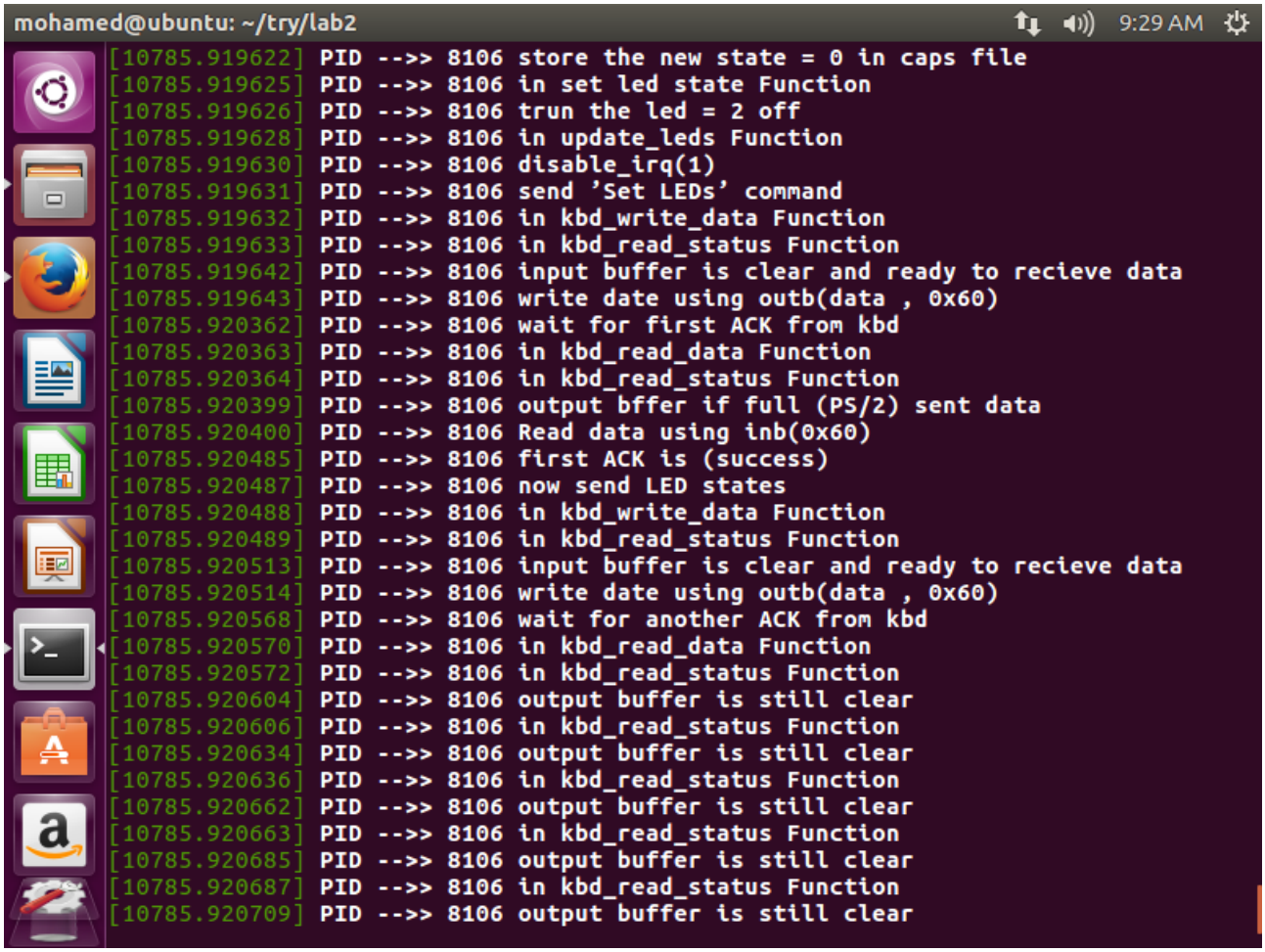
test race condtions as shown in (change 0xED data on >> off , off >> on):



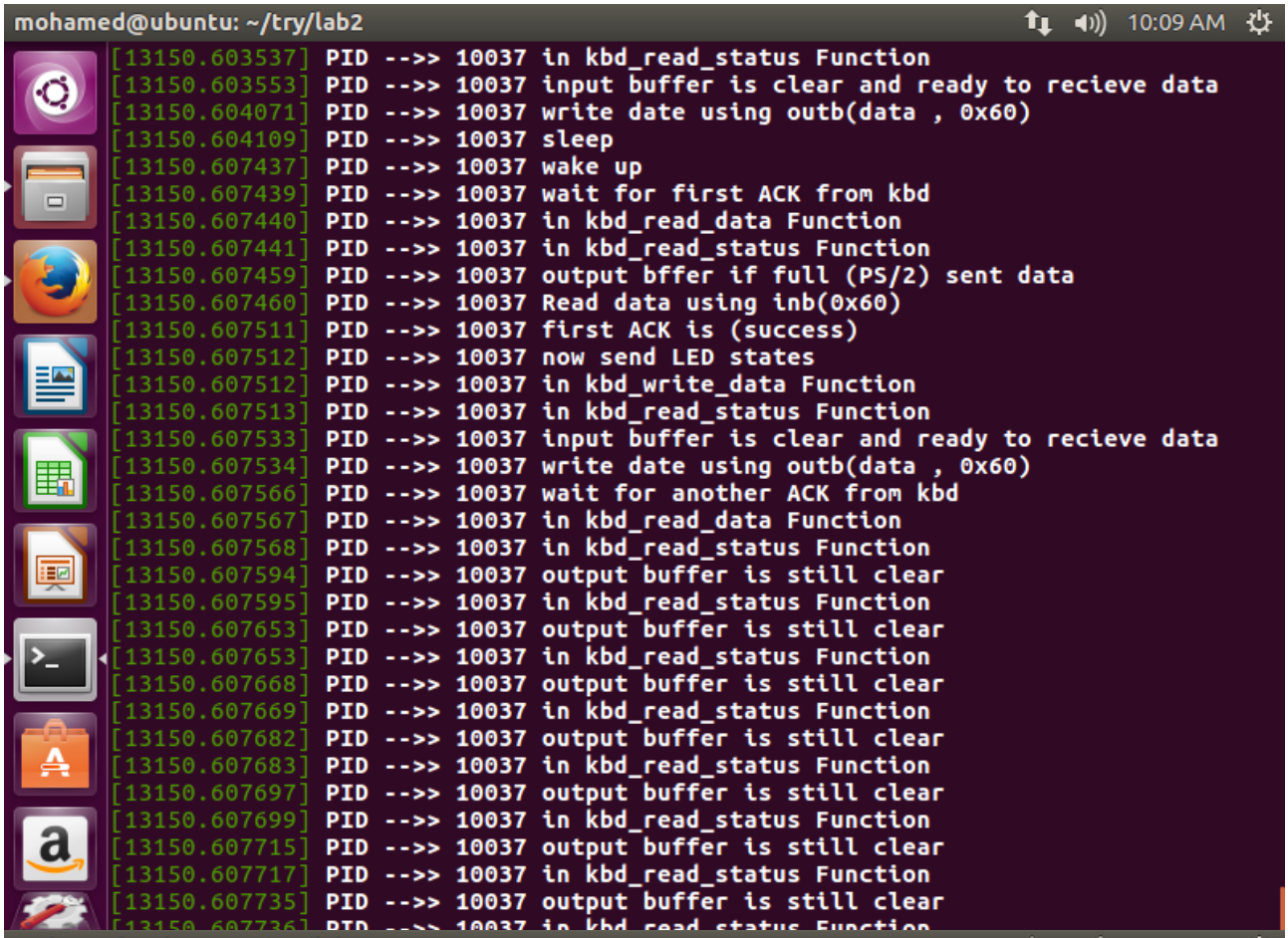
test race condtions as shown in (early enable irq(1) or change send led states):

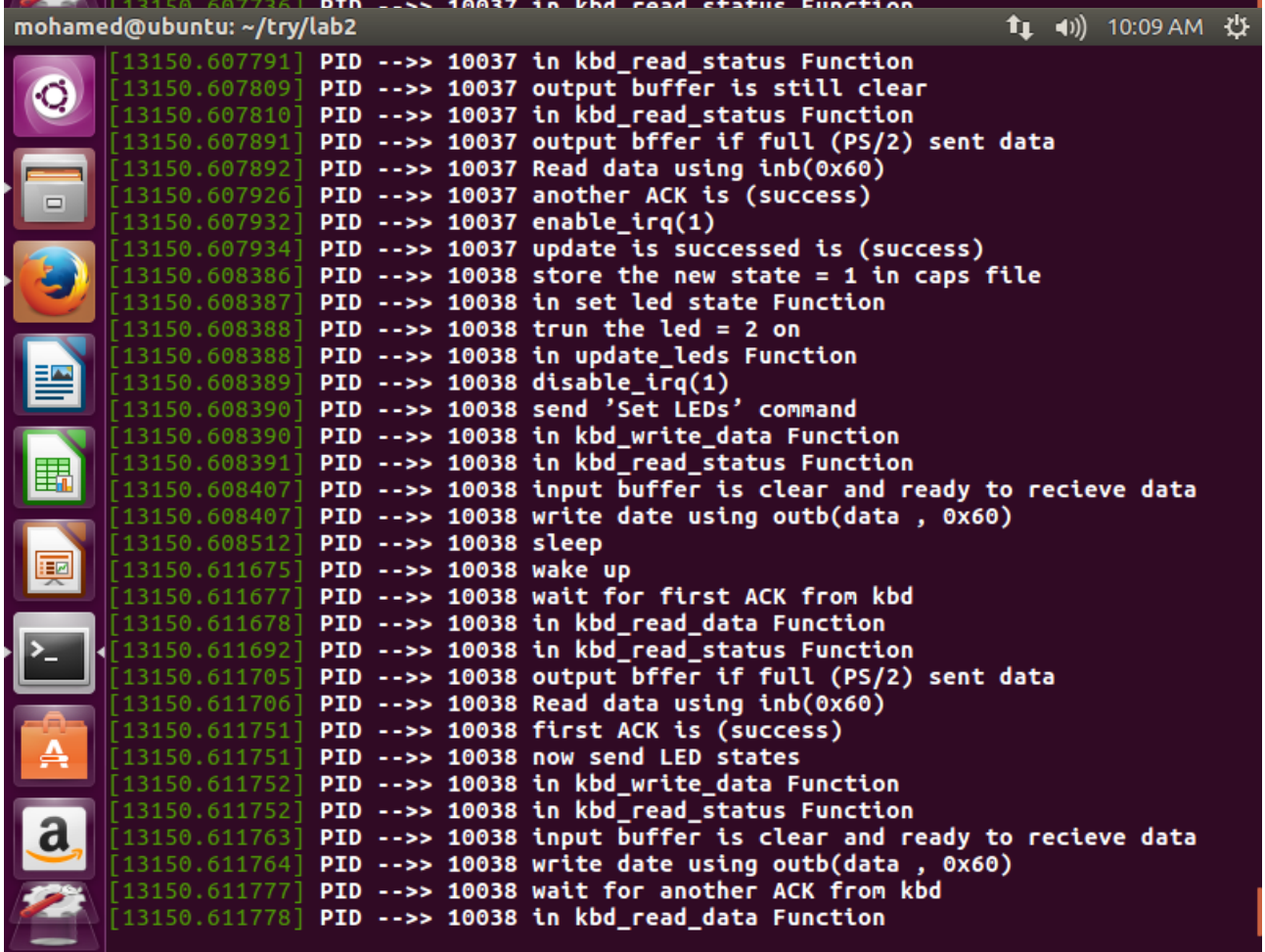


test race condtions as shown in (over write):



test semaphore(to handle first race condtion):





test semaphore for all race conditions: