Event Handling and Signaling in Linux

Assignment 4

Submitted for the Subject

Embedded Systems Programming

CSE438

By:

Nisarg Trivedi (1213314867)

Vishwakumar Doshi (1213322381)

Under the guidance of:

Prof. Yann-Hang Lee

Harsh Patel

Arizona State University

Tempe Campus, AZ

September, 2017

ACKNOWLEDGEMENT

We owe a great thanks to many people who helped and supported us during the completion of this project.

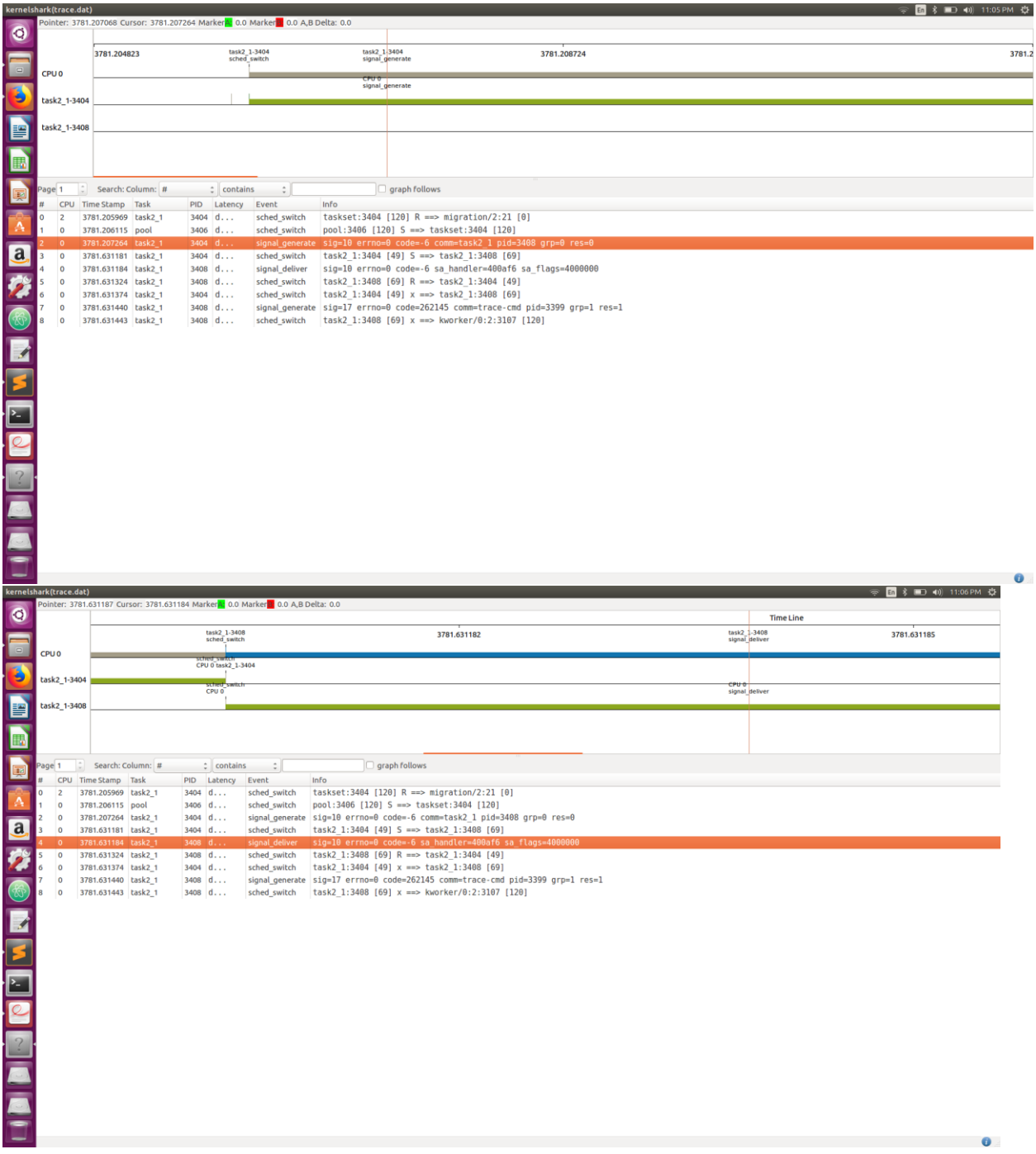
Our deepest thanks go to **Prof. Yann-Hang Lee**, Professor for the subject Embedded System Programming who acted as our mentor and guided us throughout this project. We thank him for giving ideas and briefs for the work that had to be done.

A special thanks to **Harsh Patel**, Teaching Assistant, Embedded System Programming, for constant help and guidance in the labs as well as discussion group.

We also thank all our friends for helping us out during the downs. We respect everyone for giving their time when we were in need.

Thank You.

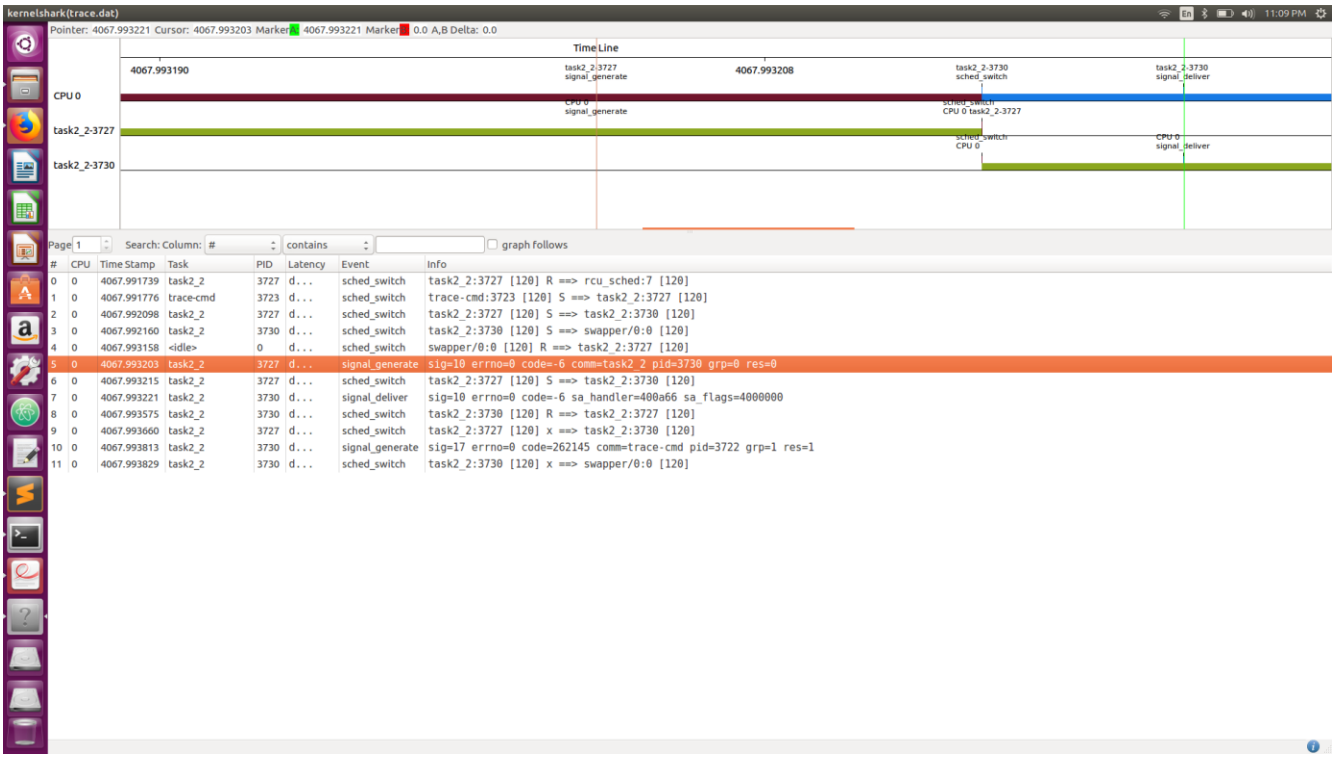
Task 2 part 1 (runnable but not running):



In this case a runnable thread is created and a signal is generated for that particular thread, the thread waits until the higher priority thread exits and then the signal handler is called by the thread of lower

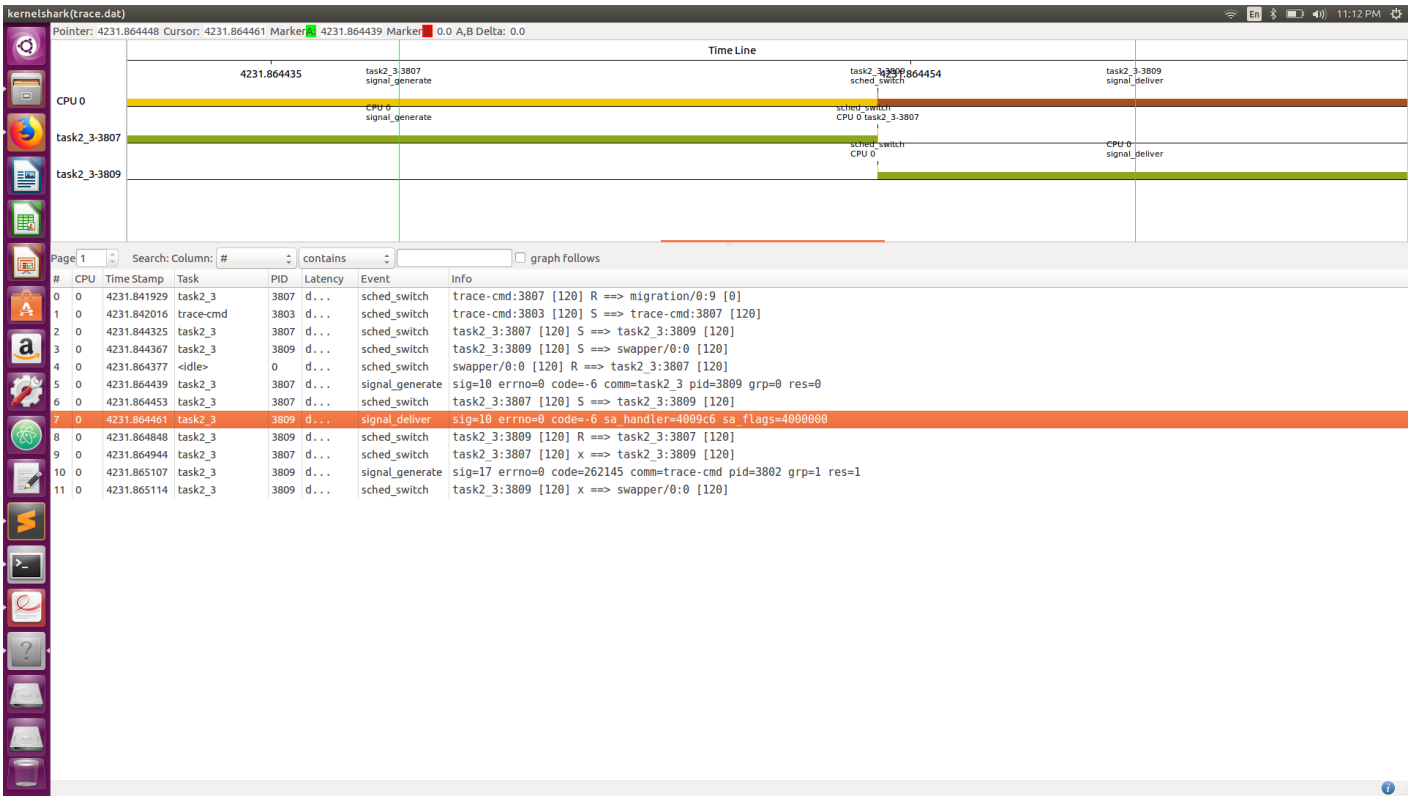
priority. The same can be seen in the above images where signal is generated in the high priority thread and when it completes its task the low priority thread (7635) receives the signal and the process is killed.

Task 2 part 2(Waiting for a semaphore)



In this case a thread is created and a semaphore is initialized. now when a signal is generated from the main thread to the thread which is in blocked state it receives the signal and gets killed and the program is exited. The same is shown in the above image where the signal is sent by the main thread and is received by the thread which is in blocked state.

Task 2 part 3(Sleeping and signal is sent)



In this case a thread is created and a is in sleep state using nanosleep. now when a signal is generated from the main thread to the thread which is in sleeping state it receives the signal and gets killed and the program is exited. The same is shown in the above image where the signal is sent by the main thread and is received by the thread which is in sleeping state.