CSE 438
Embedded Systems Programming

Event Handling and Signaling in Linux
Assignment 4

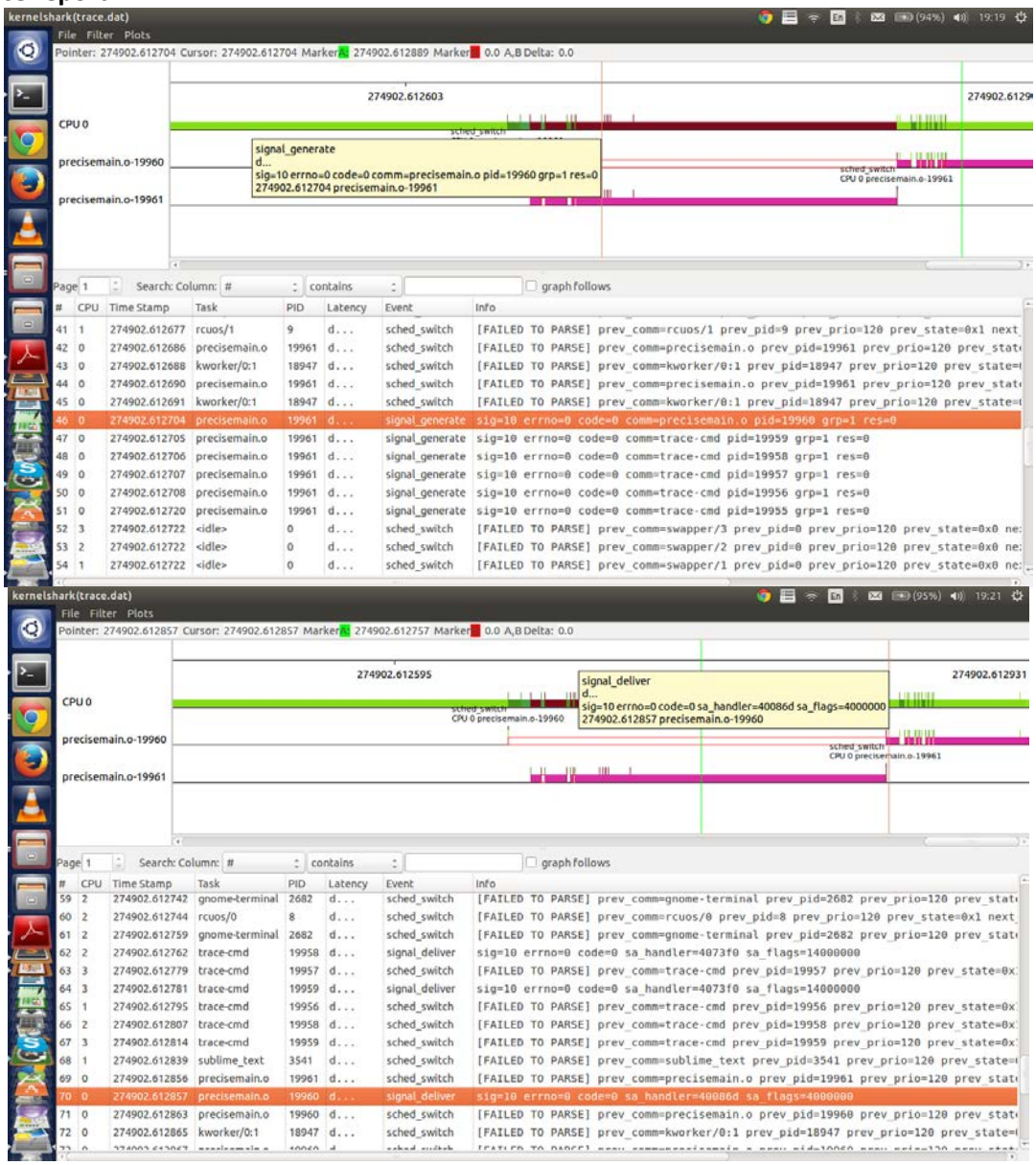
Report

Kushal Anil Parmar
ASU ID: 1207686255

Table of Contents

PART 1 – The Thread is running

Kernelshark trace report:-



Linux & POSIX facilities used: - (A)

sigaction() - system call to change the action taken by a process on receipt of SIGUSR1 signal.
kill() - used to send signal SIGUSR1 to a process.
pthread_create() - To create a thead to generate the signal

Analysis of Part 1:-

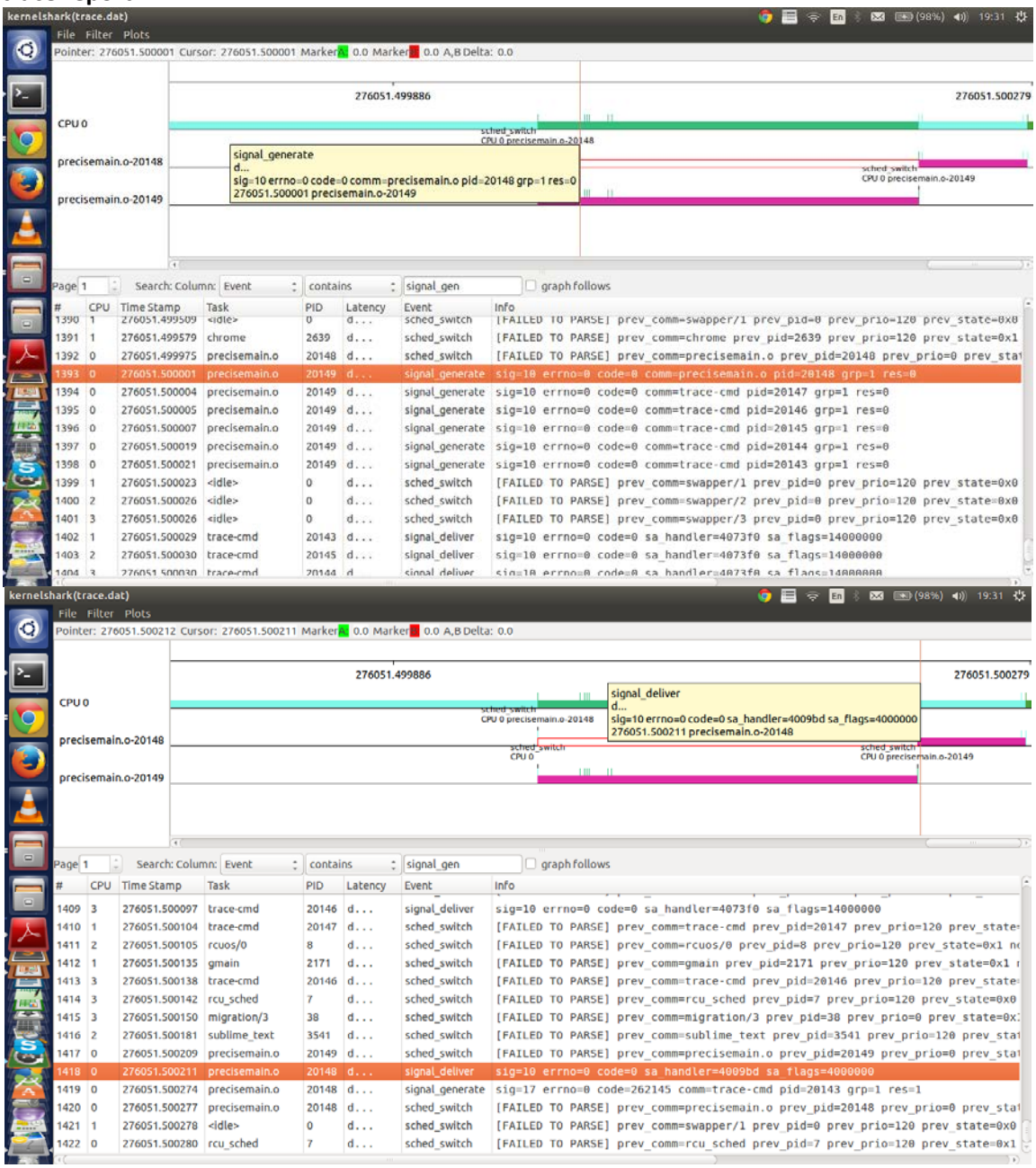
The main thread creates a pthread which is used to generate the signal SIGUSR1 using kill(). The kernel takes this signal and gives it to the signal handler.
After the signal is handled, the signal handler will set the global flag “sigusr1_flag”, so that the main thread will print a message stating that the signal is handled and breaks from there.

Trace command used :- trace-cmd record –e sched_switch –e signal taskset –c 0 ./precisemain.o

Signal generated on time :- 274902.612704
Signal delivered on time :- 274902.612857
Time taken = 153 uS

PART 2 – The Thread is runnable (running thread has a higher priority)

Kernelshark trace report:-



Linux & POSIX facilities used: (A) + additional

pthread_getschedparam()-used to retrieve the scheduling policy & parameters of the thread.
pthread_setschedparam()-used to set the scheduling policy(SCHED_RR) & parameters of thread

Analysis of Part 2:-

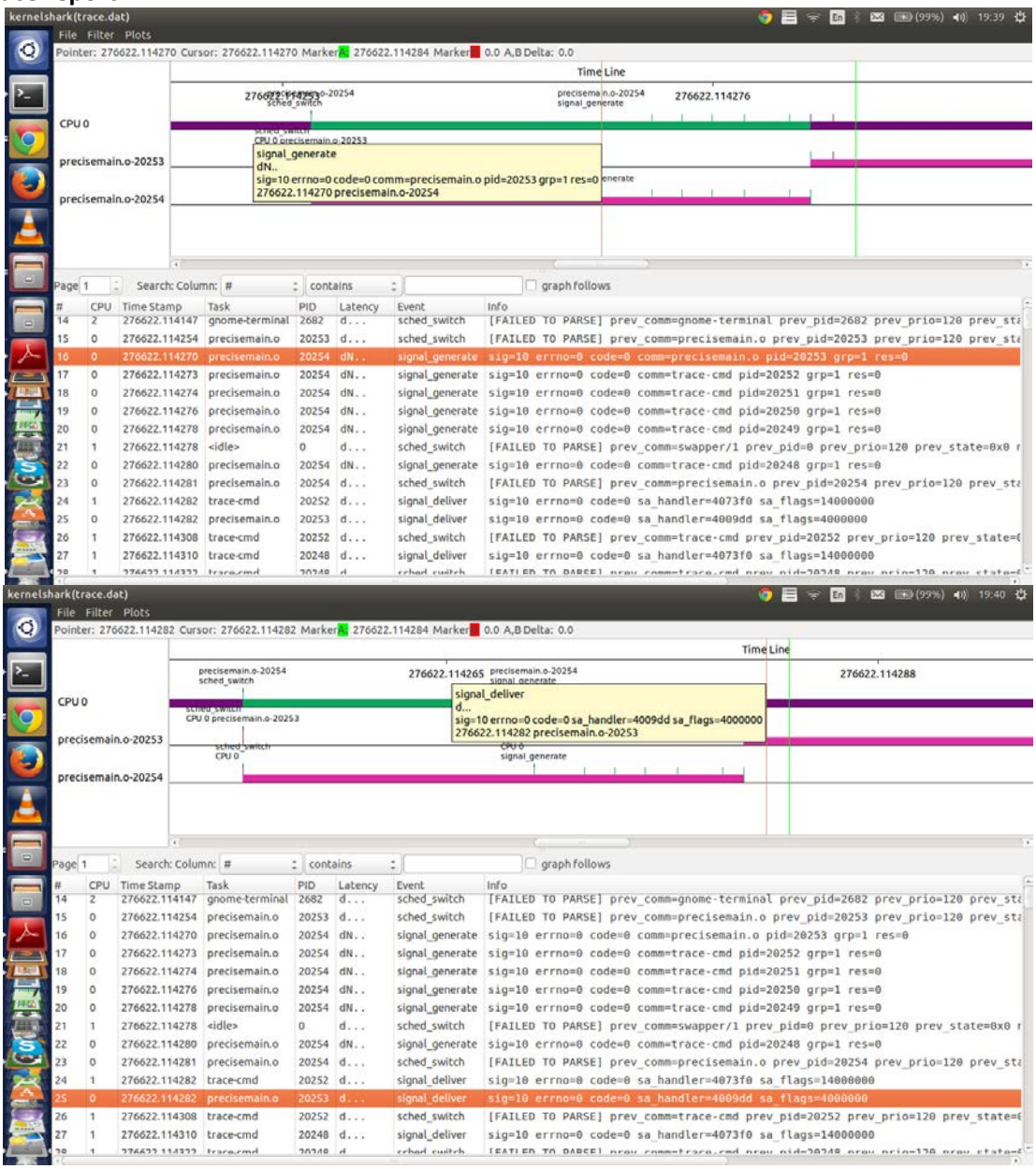
The main thread creates a pthread which is used to generate the signal SIGUSR1 using kill(). The kernel takes this signal gives it to the signal handler.
After the signal is handled, the signal handler will set the global flag “sigusr1_flag”, so that the main thread will end the task.

Trace command used :- trace-cmd record –e sched_switch –e signal taskset –c 0 ./precisemain.o

Signal generated on time:- 276051.500001
Signal delivered on time :- 276051.500211
Time taken = 210 uS

PART 3 – The Thread is blocked by a semaphore or a File IO

Kernelshark trace report:-



Linux & POSIX facilities used: (A) + additional

sem_wait() - This blocks the calling thread and locks the semaphore.
sem_post() - This unlocks the semaphore and wakes up the process or thread waiting for it.

Analysis of Part 3:-

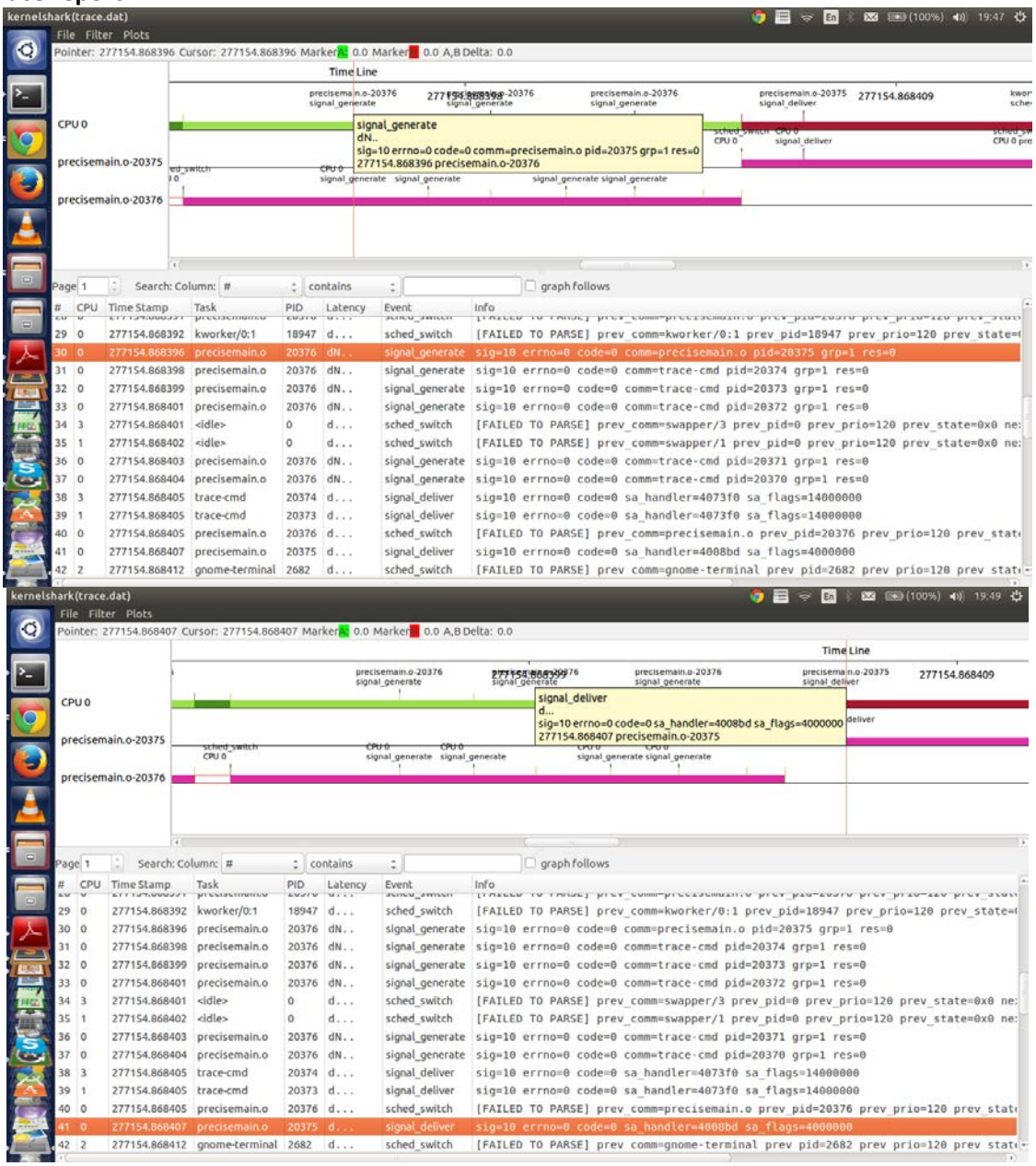
The main thread creates a thread and initialises a semaphore. The main thread is in blocking state until we use another thread which generates the signal and delivers it to the main thread to interrupt it. Now the signal is delivered to the main thread and the other thread continues to execute before acquiring the semaphore lock.

Trace command used :- trace-cmd record –e sched_switch –e signal taskset –c 0 ./precisemain.o

Signal generated on time:- 276622.114270
Signal delivered on time :- 276622.114282
Time taken = 12 uS

PART 4 – The Thread is delayed (nanosleep() is called)

Kernelshark trace report:-



Linux & POSIX facilities used: (A) + additional

nanosleep(time, remaining) - suspends the execution of the calling thread until either at least the time specified in time has elapsed, or the delivery of a signal that triggers the invocation of a handler in the calling thread or that terminates the process.

Analysis of Part 4:-

The main thread created a thread which generated a signal using kill(), returns to the main thread. The main thread goes to sleep using nanosleep(). The main thread wakes up on receiving an interrupt from the signal from the other thread. After the main thread receives the signal it stays awake and the other thread goes to sleep for the remaining time and later execute the code.

Trace command used :- trace-cmd record –e sched_switch –e signal taskset –c 0 ./precisemain.o

Signal generated on time:- 277154.868407
Signal delivered on time :- 277154.868396
Time taken = 11 uS