CSE 438/598: Embedded Systems Programming

EVENT HANDLING AND SIGNALING

Report

Assignment 5

05-DEC-2014

Author: Ankit Rathi

ASU ID: 1207543476

Table of Contents

INTRODUCTION

Assignment Statement: In vxWorks' Kernel API Reference Manual, it is stated that "If a task is pended (for instance, by waiting for a semaphore to become available) and a signal is sent to the task for which the task has a handler installed, then the handler will run before the semaphore is taken. When the handler returns, the task will go back to being pended (waiting for the semaphore). If there was a timeout used for the pend, then the original value will be used again when the task returns from the signal handler and goes back to being pended. If the handler alters the execution path, via a call to longjmp( ) for example, and does not return then the task does not go back to being pended."

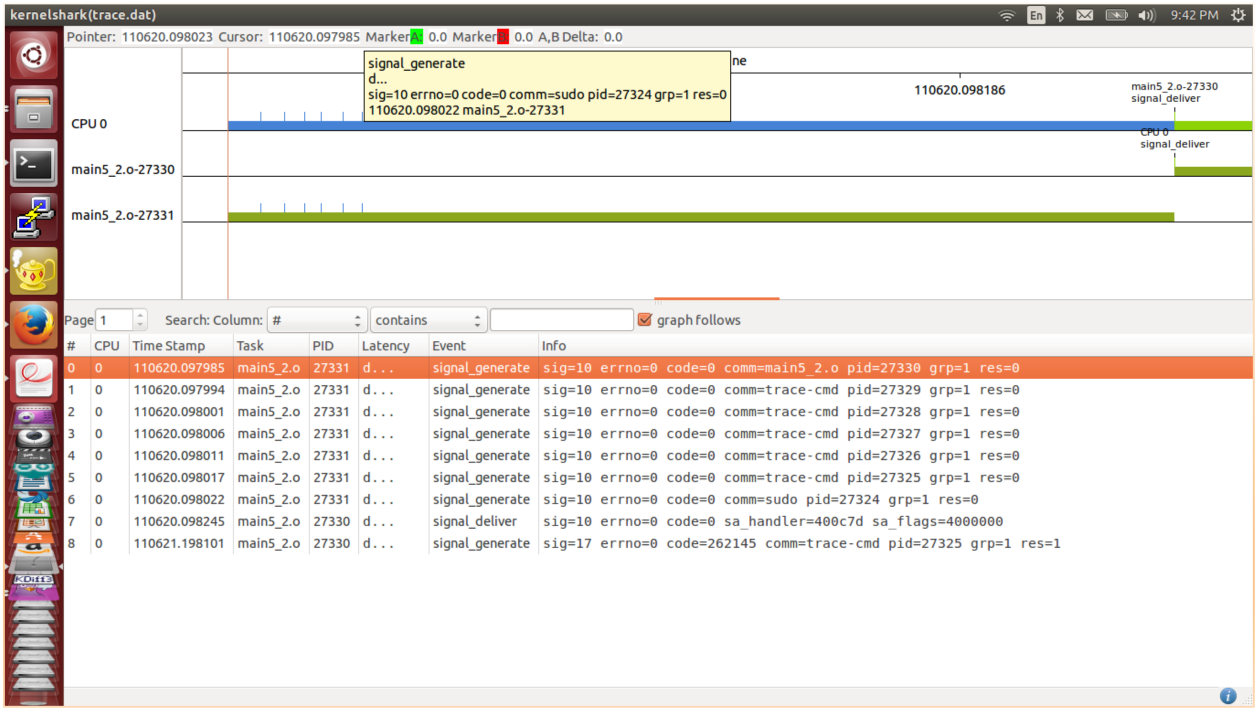
This description of the signal delivery mechanism is certainly OS dependent. In task 2 of the assignment, a program is developed to test what the Linux's signal facility does precisely and to show the time that a signal handler associated to a thread gets executed in the following four conditions:

a. The thread is running.
b. The thread is runnable (i.e. the running thread has a higher priority).
c. The thread is blocked by a semaphore (i.e. sema_wait() is called) or a file IO (e.g., read).
d. The thread is delayed (i.e., nanosleep() is called).

PART 1: THE THREAD IS RUNNING

2.1 IMAGE

Command: *sudo trace-cmd record -e signal taskset -c 0 ./main5_2.o*



Pointer: 110620.098023 Cursor: 110620.097985 Marker A: 0.0 Marker B: 0.0 A,B Delta: 0.0

```
signal_generate
d...
sig=10 errno=0 code=0 comm=sudo pid=27324 grp=1 res=0
110620.098022 main5_2.o-27331
```

#	CPU	Time Stamp	Task	PID	Latency	Event	Info
0	0	110620.097985	main5_2.o	27331	d...	signal_generate	sig=10 errno=0 code=0 comm=main5_2.o pid=27330 grp=1 res=0
1	0	110620.097994	main5_2.o	27331	d...	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27329 grp=1 res=0
2	0	110620.098001	main5_2.o	27331	d...	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27328 grp=1 res=0
3	0	110620.098006	main5_2.o	27331	d...	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27327 grp=1 res=0
4	0	110620.098011	main5_2.o	27331	d...	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27326 grp=1 res=0
5	0	110620.098017	main5_2.o	27331	d...	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27325 grp=1 res=0
6	0	110620.098022	main5_2.o	27331	d...	signal_generate	sig=10 errno=0 code=0 comm=sudo pid=27324 grp=1 res=0
7	0	110620.098245	main5_2.o	27330	d...	signal_deliver	sig=10 errno=0 code=0 sa_handler=400c7d sa_flags=4000000
8	0	110621.198101	main5_2.o	27330	d...	signal_generate	sig=17 errno=0 code=262145 comm=trace-cmd pid=27325 grp=1 res=1

2.2 ANALYSIS

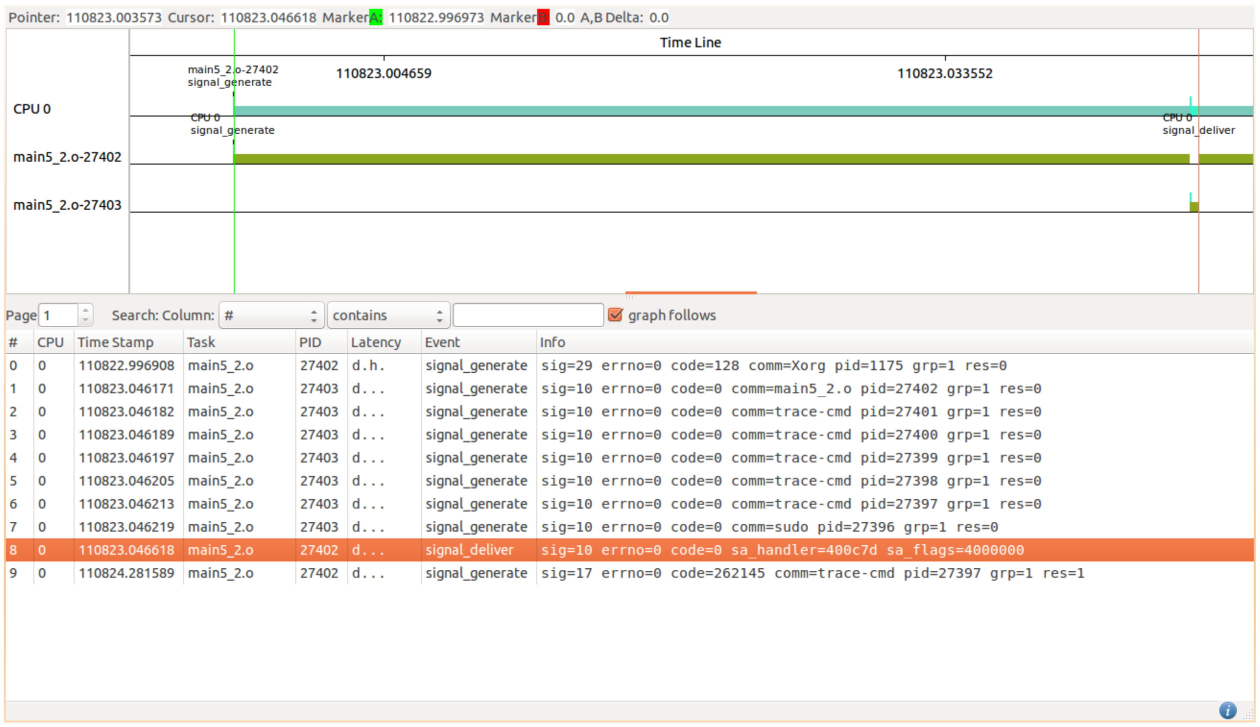
In part 1 of the assignment, the main thread creates one thread. One thread is created to generate signal. A signal is generated by the thread, by using the kill() function. This will call the signal handler.

As it can be seen from the figure, the signal_generate was at time stamp 110620.097985 and the signal was delivered at 110620.098245. Time difference from delivery to generation is 260uS.

PART 2: THE THREAD IS RUNNABLE

3.1 IMAGE

Command: *sudo trace-cmd record -e signal taskset -c 0 ./main5_2.o*



Pointer: 110823.003573 Cursor: 110823.046618 Marker A: 110822.996973 Marker B: 0.0 A,B Delta: 0.0

#	CPU	Time Stamp	Task	PID	Latency	Event	Info
0	0	110822.996908	main5_2.o	27402	d.h.	signal_generate	sig=29 errno=0 code=128 comm=Xorg pid=1175 grp=1 res=0
1	0	110823.046171	main5_2.o	27403	d...	signal_generate	sig=10 errno=0 code=0 comm=main5_2.o pid=27402 grp=1 res=0
2	0	110823.046182	main5_2.o	27403	d...	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27401 grp=1 res=0
3	0	110823.046189	main5_2.o	27403	d...	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27400 grp=1 res=0
4	0	110823.046197	main5_2.o	27403	d...	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27399 grp=1 res=0
5	0	110823.046205	main5_2.o	27403	d...	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27398 grp=1 res=0
6	0	110823.046213	main5_2.o	27403	d...	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27397 grp=1 res=0
7	0	110823.046219	main5_2.o	27403	d...	signal_generate	sig=10 errno=0 code=0 comm=sudo pid=27396 grp=1 res=0
8	0	110823.046618	main5_2.o	27402	d...	signal_deliver	sig=10 errno=0 code=0 sa_handler=400c7d sa_flags=4000000
9	0	110824.281589	main5_2.o	27402	d...	signal_generate	sig=17 errno=0 code=262145 comm=trace-cmd pid=27397 grp=1 res=1

3.2 ANALYSIS

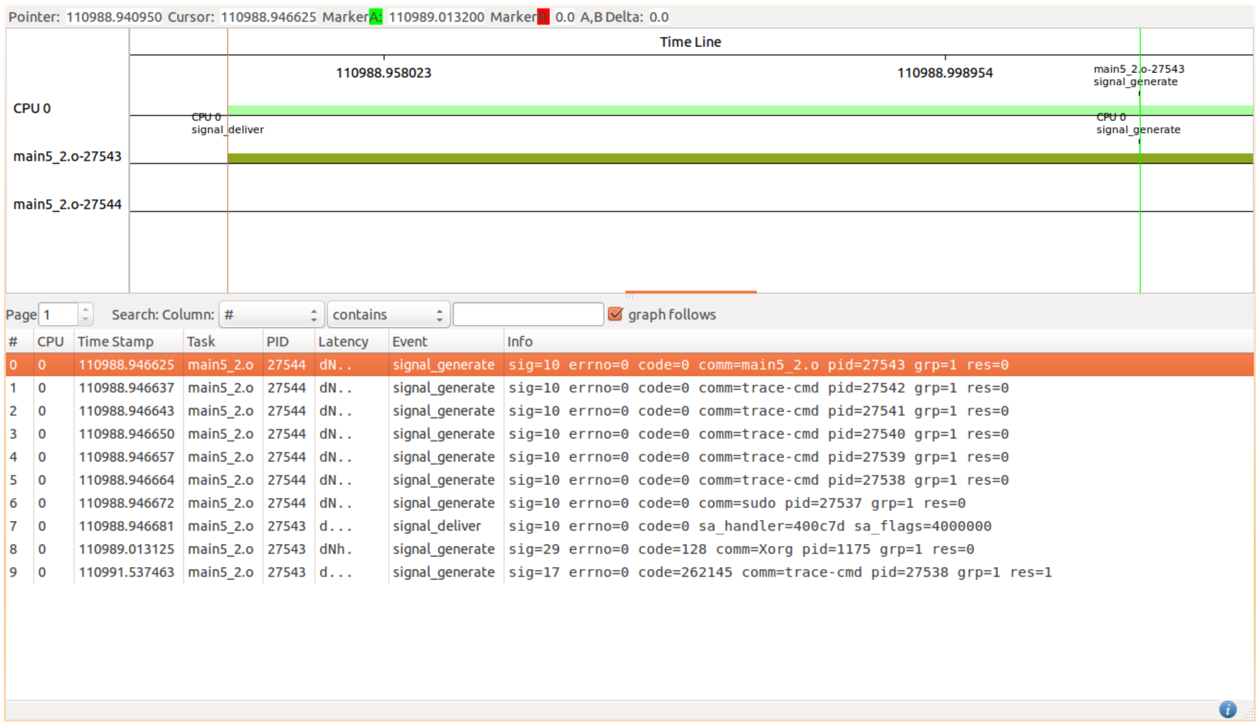
In part 1 of the assignment, the main thread creates one thread. One thread is created to generate signal. A signal is generated by the thread, by using the kill() function. This will call the signal handler.

As it can be seen from the figure, the signal_generate was at time stamp 110823.046171 and the signal was delivered at 110823.046618. Time difference from delivery to generation is 447uS.

PART 3: THE THREAD IS BLOCKED BY A SEMAPHORE

4.1 IMAGE

Command: *sudo trace-cmd record -e signal taskset -c 0 ./main5_2.o*



#	CPU	Time Stamp	Task	PID	Latency	Event	Info
0	0	110988.946625	main5_2.o	27544	dN..	signal_generate	sig=10 errno=0 code=0 comm=main5_2.o pid=27543 grp=1 res=0
1	0	110988.946637	main5_2.o	27544	dN..	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27542 grp=1 res=0
2	0	110988.946643	main5_2.o	27544	dN..	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27541 grp=1 res=0
3	0	110988.946650	main5_2.o	27544	dN..	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27540 grp=1 res=0
4	0	110988.946657	main5_2.o	27544	dN..	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27539 grp=1 res=0
5	0	110988.946664	main5_2.o	27544	dN..	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27538 grp=1 res=0
6	0	110988.946672	main5_2.o	27544	dN..	signal_generate	sig=10 errno=0 code=0 comm=sudo pid=27537 grp=1 res=0
7	0	110988.946681	main5_2.o	27543	d...	signal_deliver	sig=10 errno=0 code=0 sa_handler=400c7d sa_flags=4000000
8	0	110989.013125	main5_2.o	27543	dNh.	signal_generate	sig=29 errno=0 code=128 comm=Xorg pid=1175 grp=1 res=0
9	0	110991.537463	main5_2.o	27543	d...	signal_generate	sig=17 errno=0 code=262145 comm=trace-cmd pid=27538 grp=1 res=1

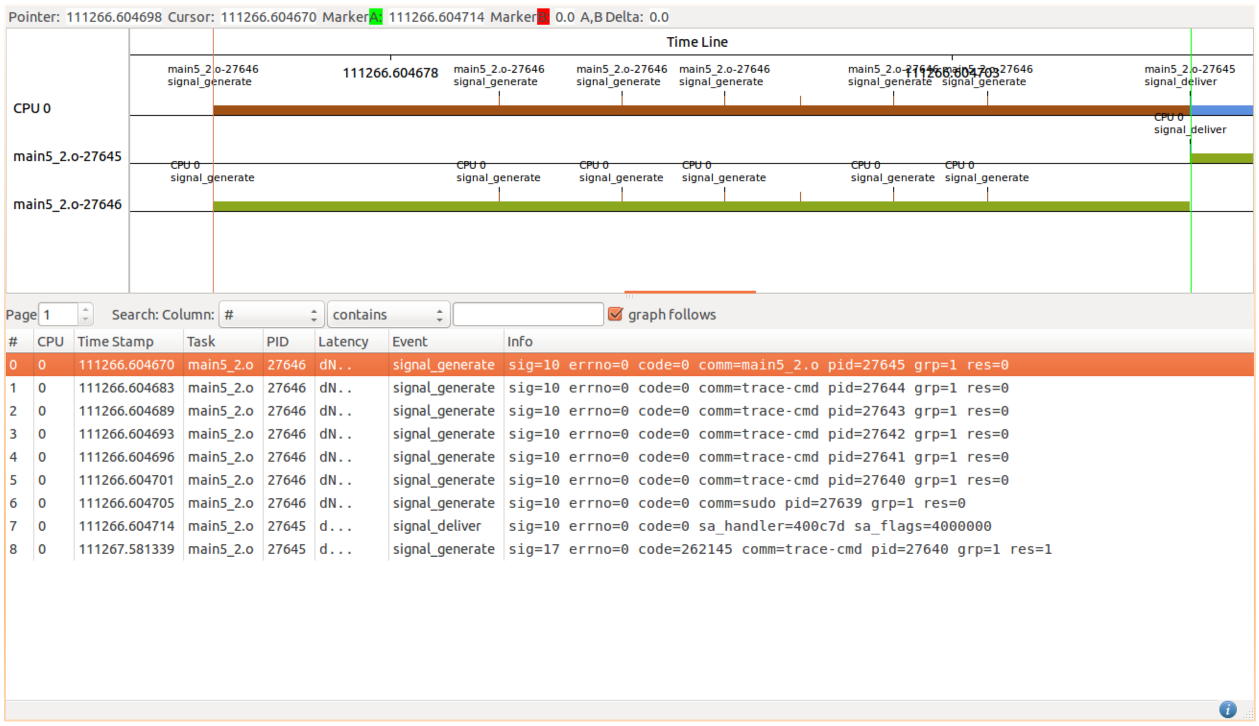
4.2 ANALYSIS

In this part of the assignment, main thread makes use of the semaphore. It attempts to get a semaphore lock, which is not free. So it gets blocked. It is in blocking state until the signal is generated by another thread is delivered to it, to interrupt it. After the signal has been delivered to the main thread, the thread continues to execute even before acquiring the lock for the same.
As it can seen from the figure that, The signal was generated at 110988.946625 and was delivered to the main thread at 110988.986681. The time between the signal delivered after it was generated is 56uS.

PART4: THE THREAD IS DELAYED

5.1 IMAGE

Command: *sudo trace-cmd record -e signal taskset -c 0 ./main5_2.o*



Pointer: 111266.604698 Cursor: 111266.604670 MarkerA: 111266.604714 MarkerB: 0.0 A,B Delta: 0.0

#	CPU	Time Stamp	Task	PID	Latency	Event	Info
0	0	111266.604670	main5_2.o	27646	dN..	signal_generate	sig=10 errno=0 code=0 comm=main5_2.o pid=27645 grp=1 res=0
1	0	111266.604683	main5_2.o	27646	dN..	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27644 grp=1 res=0
2	0	111266.604689	main5_2.o	27646	dN..	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27643 grp=1 res=0
3	0	111266.604693	main5_2.o	27646	dN..	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27642 grp=1 res=0
4	0	111266.604696	main5_2.o	27646	dN..	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27641 grp=1 res=0
5	0	111266.604701	main5_2.o	27646	dN..	signal_generate	sig=10 errno=0 code=0 comm=trace-cmd pid=27640 grp=1 res=0
6	0	111266.604705	main5_2.o	27646	dN..	signal_generate	sig=10 errno=0 code=0 comm=sudo pid=27639 grp=1 res=0
7	0	111266.604714	main5_2.o	27645	d...	signal_deliver	sig=10 errno=0 code=0 sa_handler=400c7d sa_flags=4000000
8	0	111267.581339	main5_2.o	27645	d...	signal_generate	sig=17 errno=0 code=262145 comm=trace-cmd pid=27640 grp=1 res=1

5.2 ANALYSIS

In part 4 of the assignment, the signal is generated by the thread and delivered to main thread. The main has delayed itself by going into sleep by calling nanasleep(). The main thread wakes up as soon as it receives interrupt signal generated by the thread. This signal after being delivered to main thread, leaves the main thread in the awake conditions and the threads sleep for the remaining time and then starts to execute the code.

From the figure it can be seen that, the signal was generated at 111266.604670 and it was delivered to the main thread at 111266.604714. The delay in signal being generated to being delivered is 44uS.