CS 251: Real-time Embedded Systems Homework 1

Team 09: *Manish Chugani, Yeahn Kim, Jesus Martinez Vega*

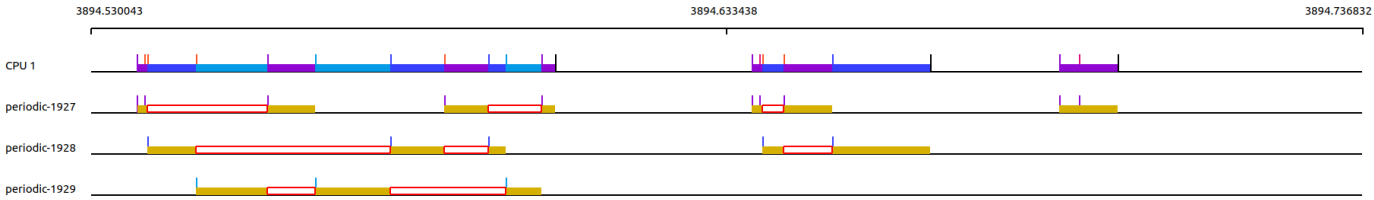
1. Run **three** instances of the periodic program (periodic.c) and capture their execution traces for 2 second using trace-cmd. Use the following task parameters:

   - Task 1: C = 10 ms, T = 50 ms, CPUID = 1
   - Task 2: C = 20 ms, T = 100 ms, CPUID = 1
   - Task 3: C = 30 ms, T = 200 ms, CPUID = 1

   To launch the tasks at the same time:

```
$ ./periodic 10 50 1 & ./periodic 20 100 1 & ./periodic 30 200 1 &
```

   Attach a screenshot of `kernelshark`. Remove other tasks (e.g., `ksoftirq`) from the plot by selecting only the three periodic tasks from `Filter - Show` tasks.

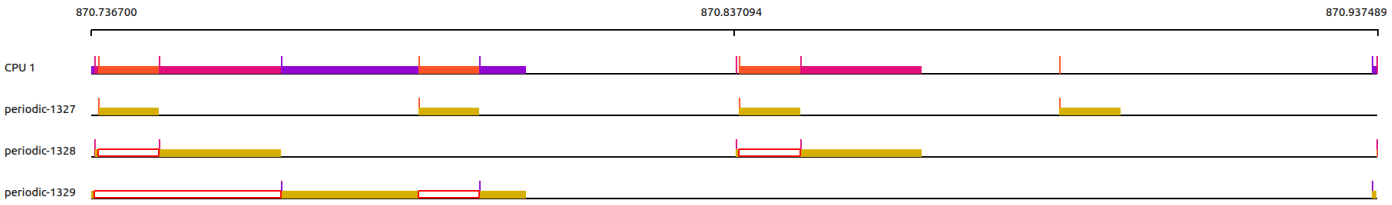


Task 1's PID is 1927, Task 2's PID is 1928, and Task 3's PID is 1929. We have captured the events for one hyper-period(200ms).

2. Run the same taskset, but after launching them, assign **real-time priorities** to each task by using `chrt` on the command line (e.g., `$ sudo chrt -f -p 3 <pid>`; see Lecture 7 slides). Use SCHED_FIFO.

   - Task 1: real-time priority = 3 (high)
   - Task 2: real-time priority = 2
   - Task 3: real-time prioritqy = 1 (low)

   Attach a screenshot of `kernelshark`. Remove other tasks from the plot.



Task 1's PID is 1327, Task 2's is 1328, and Task 3's PID is 1329. We have captured the events for one hyper-period(200ms).

3. Compare the two results and give a brief explanation of task behavior.

   The first result shows that each task is divided into multiple executions. This is because the tasks don't have real-time priorities and can preempt to execute another task. On the other hand, the second result shows that there are no preemptions among the tasks. Since all tasks have different priorities, a task with a higher priority executes first until it finishes.

4. Give a brief summary of the contributions of each member.

   - **Yeahn Kim**: Yeahn had implemented the rtmon_ioctl function and the lkm_cancel_rtmon function and made sure they executed as intended. She was also responsible for finalizing the homework by composing the report.
   - **Manish Chugani**: Manish was responsible for developing the print_rtmon function and the lkm_set_rtmon function. He tested the functions and examined that the lkm_print_rtmon function executed when a user read '`dev/rtmon`'.
   - **Jesus Martinez Vega**: Jesus was responsible for registering the misc device from the kernel module. He also worked on how hrtimer works and implemented the hrtimer_handler function and lkm_wait_for_next_period function. He successfully tested the module to verify that it performed correctly without any crashes.