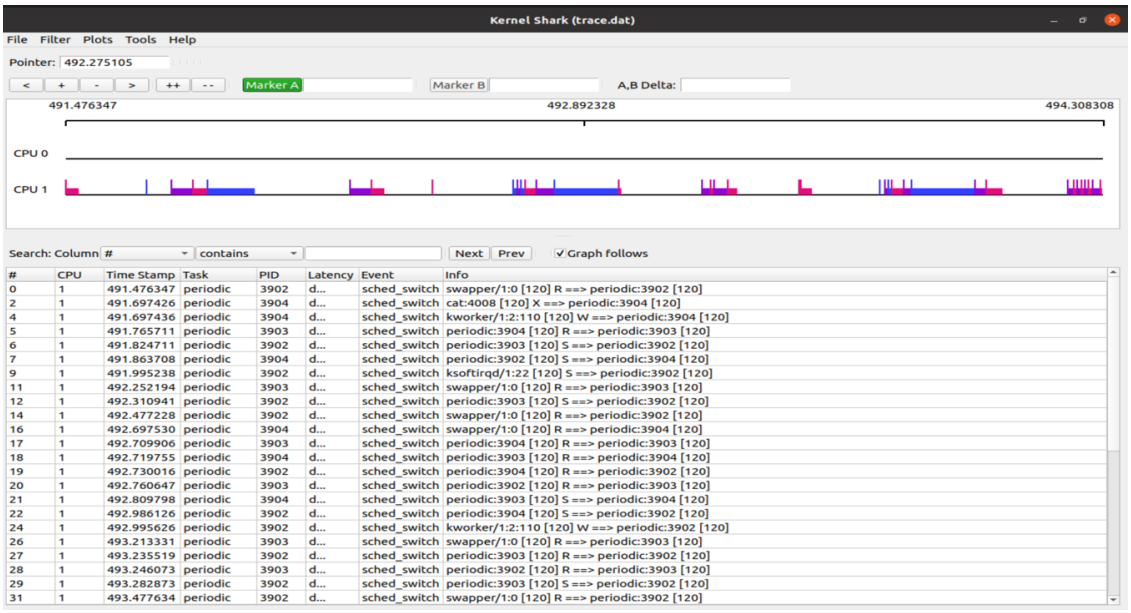
HW Project #2: Task Scheduling and Monitoring
Real-time Embedded System Assignment – CS251

Shixun Wu (swu264) Anvaya B Narappa (an001) Simran Saha (ssaha031)

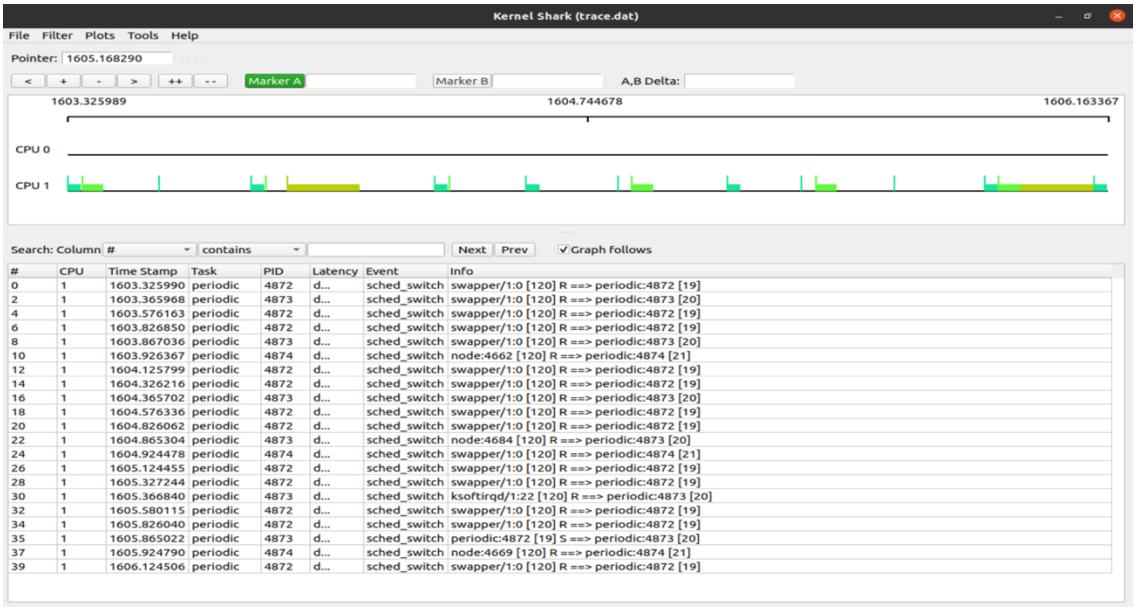
4.6 Write-Up Questions (4.6):

1.



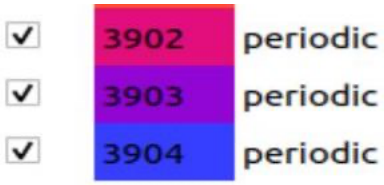
#	CPU	Time Stamp	Task	PID	Latency	Event	Info
0	1	491.476347	periodic	3902	d...	sched_switch	swapper/1:0 [120] R ==> periodic:3902 [120]
2	1	491.697426	periodic	3904	d...	sched_switch	cat:4008 [120] X ==> periodic:3904 [120]
4	1	491.697436	periodic	3904	d...	sched_switch	kworker/1:2:110 [120] W ==> periodic:3904 [120]
5	1	491.765711	periodic	3903	d...	sched_switch	periodic:3904 [120] R ==> periodic:3903 [120]
6	1	491.824711	periodic	3902	d...	sched_switch	periodic:3903 [120] S ==> periodic:3902 [120]
7	1	491.863708	periodic	3904	d...	sched_switch	periodic:3902 [120] S ==> periodic:3904 [120]
9	1	491.995238	periodic	3902	d...	sched_switch	ksoftirqd/1:22 [120] S ==> periodic:3902 [120]
11	1	492.252194	periodic	3903	d...	sched_switch	swapper/1:0 [120] R ==> periodic:3903 [120]
12	1	492.310941	periodic	3902	d...	sched_switch	periodic:3903 [120] S ==> periodic:3902 [120]
14	1	492.477228	periodic	3902	d...	sched_switch	swapper/1:0 [120] R ==> periodic:3902 [120]
16	1	492.697530	periodic	3904	d...	sched_switch	swapper/1:0 [120] R ==> periodic:3904 [120]
17	1	492.709906	periodic	3903	d...	sched_switch	periodic:3904 [120] R ==> periodic:3903 [120]
18	1	492.719755	periodic	3904	d...	sched_switch	periodic:3903 [120] R ==> periodic:3904 [120]
19	1	492.730016	periodic	3902	d...	sched_switch	periodic:3904 [120] R ==> periodic:3902 [120]
20	1	492.760647	periodic	3903	d...	sched_switch	periodic:3902 [120] R ==> periodic:3903 [120]
21	1	492.809798	periodic	3904	d...	sched_switch	periodic:3903 [120] S ==> periodic:3904 [120]
22	1	492.986126	periodic	3902	d...	sched_switch	periodic:3904 [120] S ==> periodic:3902 [120]
24	1	492.995626	periodic	3902	d...	sched_switch	kworker/1:2:110 [120] W ==> periodic:3902 [120]
26	1	493.213331	periodic	3903	d...	sched_switch	swapper/1:0 [120] R ==> periodic:3903 [120]
27	1	493.235519	periodic	3902	d...	sched_switch	periodic:3903 [120] R ==> periodic:3902 [120]
28	1	493.246073	periodic	3902	d...	sched_switch	periodic:3902 [120] R ==> periodic:3903 [120]
29	1	493.282873	periodic	3902	d...	sched_switch	periodic:3903 [120] S ==> periodic:3902 [120]
31	1	493.477634	periodic	3902	d...	sched_switch	swapper/1:0 [120] R ==> periodic:3902 [120]

2.



#	CPU	Time Stamp	Task	PID	Latency	Event	Info
0	1	1603.325990	periodic	4872	d...	sched_switch	swapper/1:0 [120] R ==> periodic:4872 [19]
2	1	1603.365968	periodic	4873	d...	sched_switch	swapper/1:0 [120] R ==> periodic:4873 [20]
4	1	1603.576163	periodic	4872	d...	sched_switch	swapper/1:0 [120] R ==> periodic:4872 [19]
6	1	1603.826850	periodic	4872	d...	sched_switch	swapper/1:0 [120] R ==> periodic:4872 [19]
8	1	1603.867036	periodic	4873	d...	sched_switch	swapper/1:0 [120] R ==> periodic:4873 [20]
10	1	1603.926367	periodic	4874	d...	sched_switch	node:4662 [120] R ==> periodic:4874 [21]
12	1	1604.125799	periodic	4872	d...	sched_switch	swapper/1:0 [120] R ==> periodic:4872 [19]
14	1	1604.326216	periodic	4872	d...	sched_switch	swapper/1:0 [120] R ==> periodic:4872 [19]
16	1	1604.365702	periodic	4873	d...	sched_switch	swapper/1:0 [120] R ==> periodic:4873 [20]
18	1	1604.576336	periodic	4872	d...	sched_switch	swapper/1:0 [120] R ==> periodic:4872 [19]
20	1	1604.826062	periodic	4872	d...	sched_switch	swapper/1:0 [120] R ==> periodic:4872 [19]
22	1	1604.865304	periodic	4873	d...	sched_switch	node:4684 [120] R ==> periodic:4873 [20]
24	1	1604.924478	periodic	4874	d...	sched_switch	swapper/1:0 [120] R ==> periodic:4874 [21]
26	1	1605.124455	periodic	4872	d...	sched_switch	swapper/1:0 [120] R ==> periodic:4872 [19]
28	1	1605.327244	periodic	4872	d...	sched_switch	swapper/1:0 [120] R ==> periodic:4872 [19]
30	1	1605.366840	periodic	4873	d...	sched_switch	ksoftirqd/1:22 [120] R ==> periodic:4873 [20]
32	1	1605.580115	periodic	4872	d...	sched_switch	swapper/1:0 [120] R ==> periodic:4872 [19]
34	1	1605.826040	periodic	4872	d...	sched_switch	swapper/1:0 [120] R ==> periodic:4872 [19]
35	1	1605.865022	periodic	4873	d...	sched_switch	periodic:4872 [19] S ==> periodic:4873 [20]
37	1	1605.924790	periodic	4874	d...	sched_switch	node:4669 [120] R ==> periodic:4874 [21]
39	1	1606.124506	periodic	4872	d...	sched_switch	swapper/1:0 [120] R ==> periodic:4872 [19]

3. **PID: 3904, C: 200, T: 1000, CPUID: 1**
   PID: 3903, C: 60, T: 500, CPUID: 1
   PID: 3902, C: 40, T: 250, CPUID: 1



In the first scenario, when they are launched without real-time priorities, the periodic tasks will be carried out in accordance with their period and execution time. By default, the OS schedules the processes by attempting to distribute CPU time fairly amongst all processes. On the assigned CPU, each job will run uninterrupted by any other non-real-time processes. However, the CPU scheduler might switch between tasks, which might cause some jitter or fluctuation in how long it takes for each task to finish.

PID: 4872, C: 40, T: 250, CPUID: 1 priority: 80
PID: 4873, C: 60, T: 500, CPUID: 1 priority: 79
PID: 4874, C: 200, T: 1000, CPUID: 1 priority: 78

In the second scenario, where real-time priorities are assigned to each task using the 'chrt' tool, the jobs will be scheduled with a priority according to the set priority level. So, because we are using SCHED_FIFO, the processes with higher priority will have full access to the processor for as long as it needs. This means that if tasks with a higher priority are already working, they might take precedence over tasks with a lower priority. As a result, the execution of the task set as a whole might become more deterministic and predictable.
In conclusion, as the image in Ans.2 suggest that tasks 4872, 4873, and 4874 execute in this specified order, till they are executed for that period.

4. Contributions: All the members of the group worked together towards the completion of the assignment through frequent meetings over zoom.
   Simran Saha: Writeup and 4.1 (Periodic Real-time User Level Test Program) and 4.2 (Setting Task timing parameters)
   Anvaya B Narappa: 4.3 (Printing Task timing parameters) and 4.4 (Admission Control)
   Shixun Wu: 4.5 (Computation time tracking) and 4.6 (Periodic Task Support)

4.7 Enforced Budget:

We implemented the bonus part: 4.7 Enforce budget. We implement it by using one more *hrtimer*, *budget_hrtimer*. The *budget_hrtimer* initializes with a period with C. Due to the task being interruptable, the callback function first compares whether the accumulated CPU time CMT with the budget C. If the accumulated CPU time CMT is less than C, then the budget_hrtimer will be extended for (C-CMT). Otherwise, the budget_hrtimer will be ended and wait for the next period begins.