Lab 2 - Task Control
CEC 450 Spring 2020, 02/26/20

Cameron Stark (CS - starkc1@my.erau.edu)
Kevin Dumitrescu (KD - dumitrek@my.erau.edu)

Section 1: Effort

Planning and Preparation:
CS - 1 h 15 m
KD - 1 h 15 m
Experiment:
5 h
Report Writing:
4.5 h

Section 2: Objectives

This week's lab experiments focused on using the System Viewer feature in the VXWorks simulator. task priorities, and execution order. The objective of this lab was figuring out how the System Viewer worked, identifying task execution output, and altering source code to change the order of execution of tasks.

Section 3: Procedures and Results

Part A: Using System Viewer

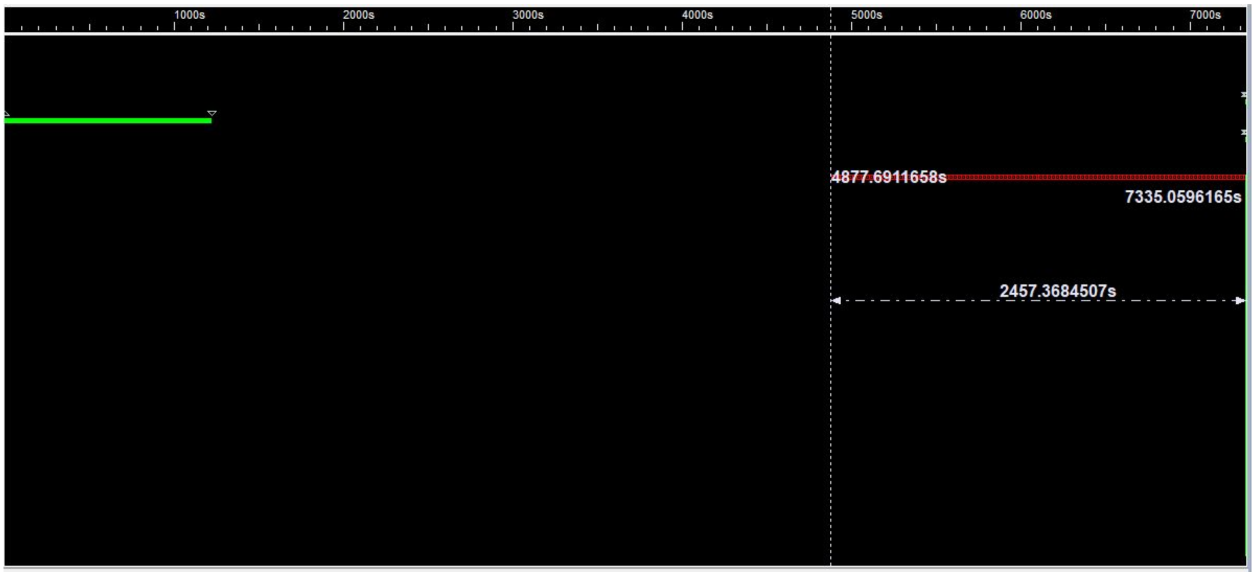
During the course of lab 2, the entirety of the lab was done using the VXWorks Simulator. To begin working on the lab, a new project needed to be created in which we added the source code given to us, complied with said source code, downloaded the object file, and then launched shell (A2). We then proceeded to setup the System Viewer, which we found to be rather simplistic due to the clear names of the options presented through the menu (A3). After launching the System Viewer and began recording by pressing the “Go” button, we then executed the *spawn_tasks* function with an argument of 4 from the shell in the remote target console (A5). After executing the function command, we then stopped recording and took screenshots of our timing diagram as seen in the Appendix below as *System Viewer graph 1* and *System Viewer graph 2*. We then used the System Viewer graph to then find out the number of delays of the four generated tasks. In the case of t1, we found out that there were 3 delays, for t2 there were 3 delays, t3 had 2 delays, and t4 had 3 delays as well (A7). After finding out the delays, we then used the data generated from the System Viewer to then fill out the *Table 1* given to us in A8. This table data showcased that each task executed once with tasks being in the running state for longer.

Part B: Task Priorities and Execution Order

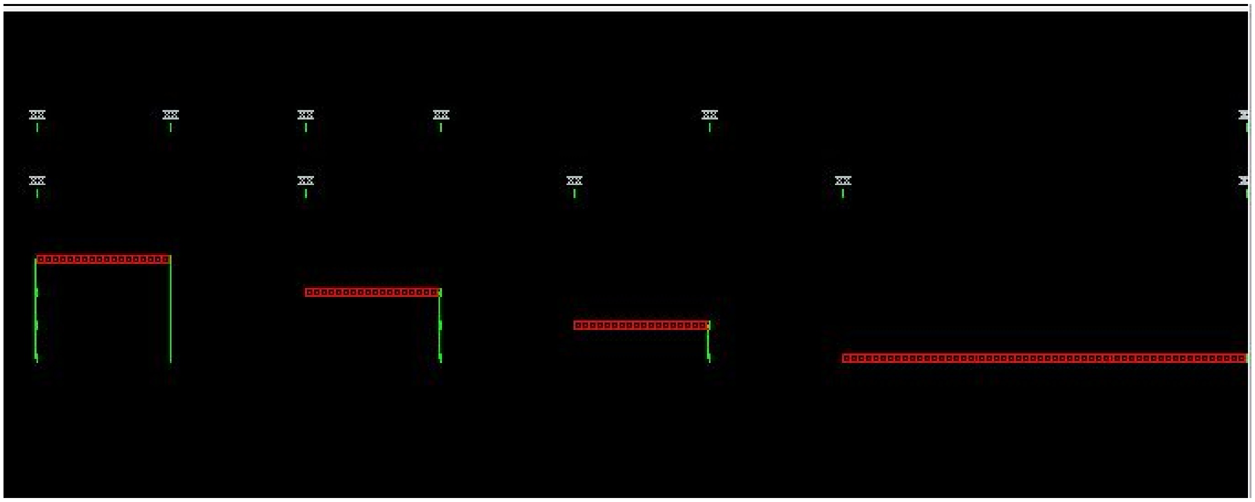
Second part of lab 2 dealt with understanding tasking priority and execution order, inside functions. The three following commands *sp*, *repeat* and *period* all deal with executing a function with arguments with different types of implementation for the times the function is run. *sp(function, arg)* spawns a task for the inserted function and the argument for the function, and only runs the duration of the function called. *repeat(repeatCount, function, arg)* repeats the inputted function for the inputted times to repeat unless 0, which is an infinite run, and the argument for the function. *period(period, function, arg)* makes the inputted function repeat infinitely with a period defined by the first argument, and the argument for the function being the third input (B1). Running the command *spawn_tasks(50)* which spawns 50 tasks and prints the associated information for the tasks, allows us to make use of the information command *i* which shows a detailed table of the processes currently in queue for the CPU with their status and priority along with other information (B2). The next functions all deal with controlling task execution, with *ts(taskId)* suspending the inputted task, *tr(taskId)* resuming the inputted the task, and *td(taskId)* deleting the inputted task (B3). The next task was to make use of the debugger provided by the vxWorks program, which as debuggers go it is not very intuitive because of the multi-step process just to get the debugger set up and when trying to get the actual function we are interested to evaluate, the break point we put into the function did not actually get reached and it was not entirely obvious how to reach it (B4). The final task was to basically invert the outputs for the provided output samples, and to do the the printing, which could be achieved by simply reverse the print order, which acted as “changing” the priority rather than actually changing the priority (B5).

Section 4: Observation, Comments, and Lessons Learned
We found out that it’s rather confusing to read the System Viewer graph to find the delays and that it’s also rather difficult to find the information needed to populate our table for A8.
Next time we will spend more time planning and preparing for the lab to make sure that we don’t run into any problems during lab execution.

Appendix



System Viewer graph 1: Captured timing diagram.



System Viewer graph 2: Execution of the spawned tasks.

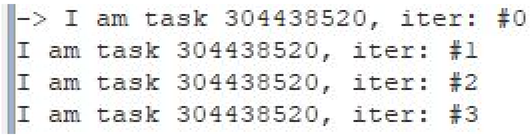
Labels Used in System Viewer	What was the print-out the tasks executed?	How long the task was in a running state? (micro seconds)

t1	Yes	9.1 us
t2	Yes	9.4 us
t3	Yes	12.5 us
t4	Yes	12.8 us

Table 1.) This table showcases the System Viewer information we gathered for the A8.

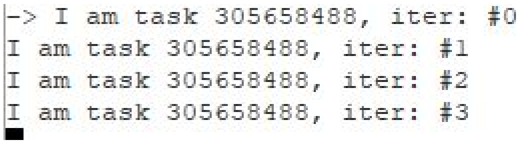
B1:

sp(printing, 4)



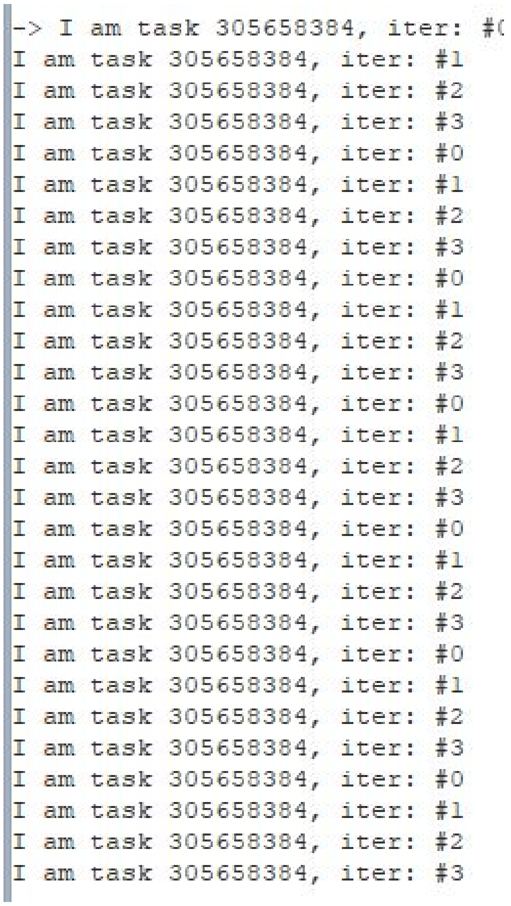
```
-> I am task 304438520, iter: #0
I am task 304438520, iter: #1
I am task 304438520, iter: #2
I am task 304438520, iter: #3
```

repeat(1, printing, 4)



```
-> I am task 305658488, iter: #0
I am task 305658488, iter: #1
I am task 305658488, iter: #2
I am task 305658488, iter: #3
```

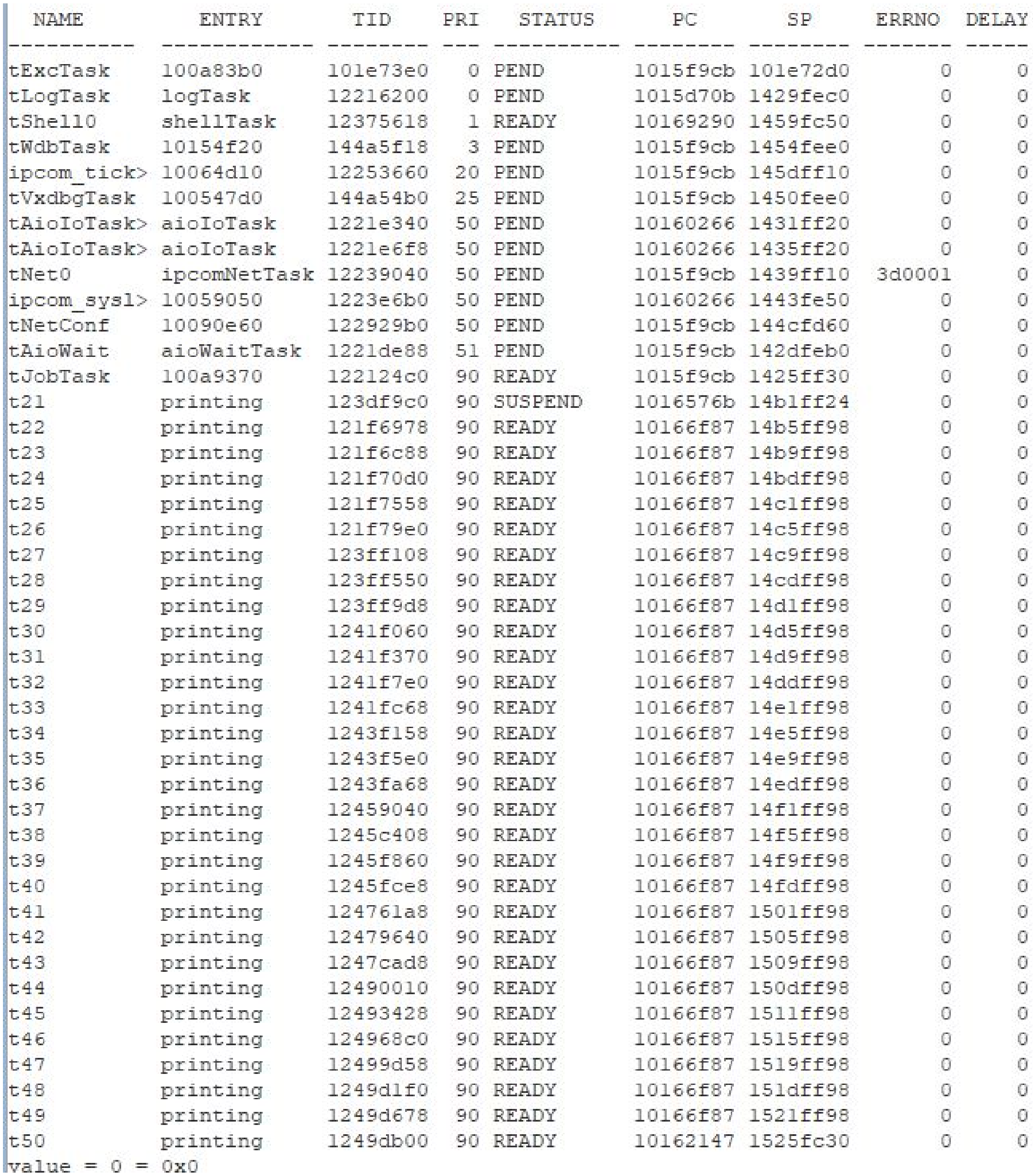
period(1, printing, 4)



```
-> I am task 305658384, iter: #0
I am task 305658384, iter: #1
I am task 305658384, iter: #2
I am task 305658384, iter: #3
I am task 305658384, iter: #0
I am task 305658384, iter: #1
I am task 305658384, iter: #2
I am task 305658384, iter: #3
I am task 305658384, iter: #0
I am task 305658384, iter: #1
I am task 305658384, iter: #2
I am task 305658384, iter: #3
I am task 305658384, iter: #0
I am task 305658384, iter: #1
I am task 305658384, iter: #2
I am task 305658384, iter: #3
I am task 305658384, iter: #0
I am task 305658384, iter: #1
I am task 305658384, iter: #2
I am task 305658384, iter: #3
I am task 305658384, iter: #0
I am task 305658384, iter: #1
I am task 305658384, iter: #2
I am task 305658384, iter: #3
I am task 305658384, iter: #0
I am task 305658384, iter: #1
I am task 305658384, iter: #2
I am task 305658384, iter: #3
```

B2:

spawn_tasks(50)



```
  NAME         ENTRY       TID    PRI   STATUS      PC       SP     ERRNO  DELAY
---------- ------------ -------- --- ---------- -------- -------- ------- -----
tExcTask   100a83b0     101e73e0   0 PEND       1015f9cb 101e72d0       0     0
tLogTask   logTask      12216200   0 PEND       1015d70b 1429fec0       0     0
tShell0    shellTask    12375618   1 READY      10169290 1459fc50       0     0
tWdbTask   10154f20     144a5f18   3 PEND       1015f9cb 1454fee0       0     0
ipcom_tick> 10064d10    12253660  20 PEND       1015f9cb 145dff10       0     0
tVxdbgTask 100547d0     144a54b0  25 PEND       1015f9cb 1450fee0       0     0
tAioIoTask> aioIoTask   1221e340  50 PEND       10160266 1431ff20       0     0
tAioIoTask> aioIoTask   1221e6f8  50 PEND       10160266 1435ff20       0     0
tNet0      ipcomNetTask 12239040  50 PEND       1015f9cb 1439ff10  3d0001     0
ipcom_sysl> 10059050    1223e6b0  50 PEND       10160266 1443fe50       0     0
tNetConf   10090e60     122929b0  50 PEND       1015f9cb 144cfd60       0     0
tAioWait   aioWaitTask  1221de88  51 PEND       1015f9cb 142dfeb0       0     0
tJobTask   100a9370     122124c0  90 READY      1015f9cb 1425ff30       0     0
t21        printing     123df9c0  90 SUSPEND    1016576b 14b1ff24       0     0
t22        printing     121f6978  90 READY      10166f87 14b5ff98       0     0
t23        printing     121f6c88  90 READY      10166f87 14b9ff98       0     0
t24        printing     121f70d0  90 READY      10166f87 14bdff98       0     0
t25        printing     121f7558  90 READY      10166f87 14c1ff98       0     0
t26        printing     121f79e0  90 READY      10166f87 14c5ff98       0     0
t27        printing     123ff108  90 READY      10166f87 14c9ff98       0     0
t28        printing     123ff550  90 READY      10166f87 14cdff98       0     0
t29        printing     123ff9d8  90 READY      10166f87 14d1ff98       0     0
t30        printing     1241f060  90 READY      10166f87 14d5ff98       0     0
t31        printing     1241f370  90 READY      10166f87 14d9ff98       0     0
t32        printing     1241f7e0  90 READY      10166f87 14ddff98       0     0
t33        printing     1241fc68  90 READY      10166f87 14e1ff98       0     0
t34        printing     1243f158  90 READY      10166f87 14e5ff98       0     0
t35        printing     1243f5e0  90 READY      10166f87 14e9ff98       0     0
t36        printing     1243fa68  90 READY      10166f87 14edff98       0     0
t37        printing     12459040  90 READY      10166f87 14f1ff98       0     0
t38        printing     1245c408  90 READY      10166f87 14f5ff98       0     0
t39        printing     1245f860  90 READY      10166f87 14f9ff98       0     0
t40        printing     1245fce8  90 READY      10166f87 14fdff98       0     0
t41        printing     124761a8  90 READY      10166f87 1501ff98       0     0
t42        printing     12479640  90 READY      10166f87 1505ff98       0     0
t43        printing     1247cad8  90 READY      10166f87 1509ff98       0     0
t44        printing     12490010  90 READY      10166f87 150dff98       0     0
t45        printing     12493428  90 READY      10166f87 1511ff98       0     0
t46        printing     124968c0  90 READY      10166f87 1515ff98       0     0
t47        printing     12499d58  90 READY      10166f87 1519ff98       0     0
t48        printing     1249d1f0  90 READY      10166f87 151dff98       0     0
t49        printing     1249d678  90 READY      10166f87 1521ff98       0     0
t50        printing     1249db00  90 READY      10162147 1525fc30       0     0
value = 0 = 0x0
```