CS 503 Lab 2 Answers

Prashant Ravi — ravi18@purdue.edu

February 19, 2016

Files modified/added

lab2q3.c ; lab2q4.c ; lab2q5.c; iobound.c ; cpubound.c ;resched.c; ready.c; prototypes.h; initialize.c; process.h

Testing Methodology

The following script in the main.c is an example for carrying out the tests. No additional parameters are needed for the lab test functions. Please take a look at exeisting main.c for example execution of tests for each lab section.

```
/*  main.c  - main */

#include <xinu.h>
#include <stdio.h>
int lab2flag=4; // set this!!!
process main(void)
{
        resume(create(lab2q4t1, 1024, 51, "lab2q4t1", 0, NULL));
        sleepms(6000);
        resume(create(lab2q4t2,1024,52, "lab2q4t2",0, NULL));
        sleepms(6000);
        resume(create(lab2q4t3,1024,52, "lab2q4t3",0, NULL));
        sleepms(3000);
        resume(create(shell, 8192, 50, "shell", 1, CONSOLE));
        /* Wait for shell to exit and recreate it */
        return OK;
}
```

The global variable lab2flag needs to be set prior to testing so as to enable the scheduling policy required to test the corresponding section of the lab, i.e set to 3/4/5. The sleeps in between resumes above are enabled for easy viewing(printing) to the console. To simulate these results I have 4 worker processes of priority of 54 and 1 monitor process of priority 57. Worker processes classify as either workerproceesTypeA or workerprocessTypeB. Type A processes execute an infinite loop, and Type B is similar to Type A execept that it will sleep on having counted upto a certain number(say 200), it will invoke a sleep instruction for 20ms and relinquish the processor.

Next up, we explain the monitor process. The monitor process is responsible for spawning the worker processes and then calling sleepms for a designated time. Since, monitor process has the highest priority it will take up the CPU exclusively. On relinquishing the processor with sleep for the designated time, the 4 worker processes get a chance to execute and we get a chance to test. But when the monitor process wakes up, it executes the process status shell xsh-ps code to print the status of the processes and then kills the other 4 worker processes. Hence, this is the general methodology used to simulate the below results and by this we can visibly reason/ make inferences about the process execution.

Question 3: Monitoring CPU usage of processes

1 Test for fairness

In the image above, the upper half that describes lab2q3t2, I have set four workerprocessTypeA processes to have equal priority and they all share the CPU time, so as printed by the column CPU ms, they share the CPU time equitably. With worker process 1 being the last to execute right before the monitor process wakes up and takes over the cpu(after workerProc 1 time quantum completes of course).

2 Test for relinquishing CPU

In the middle part of the image above, I have set 4 processes to have equal priority. However, worker proce 1-3 are workerprocessTypeA and worker proc 4 is of type workerprocessTypeB. As mentioned earler, workerprocessTypeB is set to have a sleepms() invocation after a certain duration(after it counts up to 2000000). Hence, we can assume this process sleeps a lot. Until then, however it whirs in its infinite loop. So, as we can tell, process 4 has received significantly lesser time as compared to the other 3 processes.

3 Test for highest priority process hogging CPU

In this test I have set the 3 workerprocessTypeA processes to have same priority 54 however worker proc 4 has a higher priority 55 (yet its priority is lower than that of monitor process whose priority is 57). As we can see for the duration that the monitor process sleeps, the worker proc 4 hogs the CPU without allowing the other 3 worker processes to execute and takes up all of the 375 ms allotted for worker processes to execute. In other words, monitor process was let to sleep only for 375 ms, and then after it wakes up it takes over the cpu and prints the above information for us to visually validate.

```
 -------------------------
 In lab2q3t1 test | Note: sleeping for 5 seconds .. zzzzz ...
 -------------------------
 Clock time in seconds since boot is 5
 Clock time in milliseconds since boot is 5053
 -------------------------
 In lab2q3t2 test | Note: 4 (workerProcessTypeA) equal CPU usage seen by  monitorProcess
 (Monitor process will have higher priority than workerProcessTypeA)
 -------------------------
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size CPU time      ReadyQ Posn
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ---------- -----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEF1C     8192          0           6
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCAAC    16384          2
  3 Main process     ready   20    0 0x0EFD8FFC 0x0EFD8F44    65536          1           5
  5 cputimetest1     ready   52    3 0x0FDEFFFC 0x0FDEFF40     1024         19           4
  6 Worker Proc1     ready   54    5 0x0FDEFBFC 0x0FDEFB3C     1024         66           3
  7 Worker Proc2     ready   54    5 0x0FDEF7FC 0x0FDEF76C     1024         61           0
  8 Worker Proc3     ready   54    5 0x0FDEF3FC 0x0FDEF36C     1024         61           1
  9 Worker Proc4     ready   54    5 0x0EFC8FFC 0x0EFC8F6C     1024         61           2
 10 Monitor Proc     curr    57    5 0x0EFC8BFC 0x0EFC8B5C     1024         61

 -------------------------
 In lab2q3t3 test | Note: 3 (workerProcessTypeA) and 1 sleeper workerProcessTypeB
 (Monitor process will have higher priority than workerProcessTypeA/B)
 -------------------------
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size CPU time      ReadyQ Posn
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ---------- -----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEF1C     8192          0           6
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCAAC    16384          2
  3 Main process     ready   20    0 0x0EFD8FFC 0x0EFD8F44    65536          1           5
 11 cputimetest2     ready   52    3 0x0FDEFFFC 0x0FDEFF40     1024         19           4
 12 Worker Proc1     ready   54   11 0x0FDEFBFC 0x0FDEFB3C     1024         91           1
 13 Worker Proc2     ready   54   11 0x0FDEF7FC 0x0FDEF76C     1024        121           2
 14 Worker Proc3     ready   54   11 0x0FDEF3FC 0x0FDEF33C     1024        106           3
 15 Worker Proc4     ready   54   11 0x0EFC8FFC 0x0EFC8F80     1024         61           0
 16 Monitor Proc     curr    57   11 0x0EFC8BFC 0x0EFC8B5C     1024         61

 -------------------------
 In lab2q3t4 test | Note: 3 (workerProcessTypeA) and 1 higher priority workerProcessTypeA
 (Monitor process will have higher priority than workerProcessTypeA)
 -------------------------
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size CPU time      ReadyQ Posn
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ---------- -----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEF1C     8192          0           6
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCAAC    16384          2
  3 Main process     ready   20    0 0x0EFD8FFC 0x0EFD8F44    65536          1           5
 17 cputimetest3     ready   52    3 0x0FDEFFFC 0x0FDEFF40     1024         19           4
 18 Worker Proc1     ready   54   17 0x0FDEFBFC 0x0FDEFBCC     1024          1           1
 19 Worker Proc2     ready   54   17 0x0FDEF7FC 0x0FDEF7CC     1024          1           2
 20 Worker Proc3     ready   54   17 0x0FDEF3FC 0x0FDEF3CC     1024          1           3
 21 Worker Proc4     ready   55   17 0x0EFC8FFC 0x0EFC8F3C     1024        376           0
 22 Monitor Proc     curr    57   17 0x0EFC8BFC 0x0EFC8B5C     1024         62
```

Figure 1: Experiment results using lab2q3.c tests

Question 4.2: Dynamic priority scheduling to achieve "fairness"

I used the same tests as for part 3 and these were the results. Right off the bat, we notice that the CPU time used by the CPU hogging processes and IO hogging processes is nearly indistinguishable in all three cases and all the values are 30ms within the range of each other. This shows that our dynamic priority scheduling strategy shares the CPU equitably among IO hogging and CPU hogging applications.

```
-- LAB 2.4.2 ---

 ------------------------
 In lab2q4t1 test | Note: 4 (workerProcessTypeA) equal CPU usage seen by  monitorProcess
 (Monitor process will have higher priority than workerProcessTypeA)
 ------------------------
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size CPU time      ReadyQ Posn
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ---------- -----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEF50     8192 2147483647          4
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCAAC    16384          2
  3 Main process     sleep   20    0 0x0EFD8FFC 0x0EFD8F74    65536          2
  5 Worker Proc1     ready   54    4 0x0FDEFBFC 0x0FDEFB3C     1024         96          3
  6 Worker Proc2     ready   54    4 0x0FDEF7FC 0x0FDEF76C     1024         91          0
  7 Worker Proc3     ready   54    4 0x0FDEF3FC 0x0FDEF36C     1024         91          1
  8 Worker Proc4     ready   54    4 0x0EFC8FFC 0x0EFC8F6C     1024         91          2
  9 Monitor Proc     curr    57    4 0x0EFC8BFC 0x0EFC8B5C     1024         61

 ------------------------
 In lab2q4t2 test | Note: 3 (workerProcessTypeA) and 1 sleeper workerProcessTypeB
 (Monitor process will have higher priority than workerProcessTypeA/B)
 ------------------------
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size CPU time      ReadyQ Posn
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ---------- -----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEF1C     8192 2147483647          4
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCAAC    16384          2
  3 Main process     sleep   20    0 0x0EFD8FFC 0x0EFD8F74    65536          2
 11 Worker Proc1     ready   54   10 0x0FDEFBFC 0x0FDEFB6C     1024        120          1
 12 Worker Proc2     ready   54   10 0x0FDEF7FC 0x0FDEF73C     1024        130          3
 13 Worker Proc3     ready   54   10 0x0FDEF3FC 0x0FDEF33C     1024        118          0
 14 Worker Proc4     ready   54   10 0x0EFC8FFC 0x0EFC8F80     1024        121          2
 15 Monitor Proc     curr    57   10 0x0EFC8BFC 0x0EFC8B5C     1024         61

 ------------------------
 In lab2q4t3 test | Note: 3 (workerProcessTypeA) and 1 higher priority workerProcessTypeA
 (Monitor process will have higher priority than workerProcessTypeA)
 ------------------------
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size CPU time      ReadyQ Posn
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ---------- -----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEF1C     8192 2147483647          4
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCAAC    16384          2
  3 Main process     sleep   20    0 0x0EFD8FFC 0x0EFD8F74    65536          2
 17 Worker Proc1     ready   54   16 0x0FDEFBFC 0x0FDEFB3C     1024        136          3
 18 Worker Proc2     ready   54   16 0x0FDEF7FC 0x0FDEF76C     1024        121          0
 19 Worker Proc3     ready   54   16 0x0FDEF3FC 0x0FDEF36C     1024        121          1
 20 Worker Proc4     ready   55   16 0x0EFC8FFC 0x0EFC8F6C     1024        121          2
 21 Monitor Proc     curr    57   16 0x0EFC8BFC 0x0EFC8B5C     1024         61

 -- LAB 2.4.3 ---
```

Figure 2: Dynamic priority scheduling

Question 4.3: Performance evaluation

1 All processes are CPU-bound

1.1 Nulluser runs only if no other process is ready

My design solution to this problem is to set the nulluser's prcpumsec to be MAXINT32 when it's instantiated, as can be viewed in the image above the value for prnull's CPU time usage is MAXINT32 = 2147483647. In addition, its prcpumsec field should not be updated when it is the oldprocess to be context switched out as we can't increment beyond the max value of the int32. This ensures that within the TS scheduling strategy the nulluser is always at the end of the readylist, and hence only executes if no other process is ready. The only caveat is that we can't deem this to be true for long running processes that may be running for more than 24 days. Since, 24 days is roughly MAXINT32 milliseconds. In this lab since we are not dealing with long running instances, as specified in the document specifications, we can ignore this warning for now. As can be seen from the figure below, if we call ps as we execute the code the amount of time spent by each process is within 30ms of each other. So they work in lock-step where no process is left behind.

2 All processes are iobound

As can been from the figure below, if we call ps as we execute the code, all the processes work in lock-step. They are 30ms within the bound of each other. So this is the expected result. This is assuming the values for loop1 and loop2 are the same for all 6 processes. If they were different for different processes then time alloted to each process could be unpredicatable since it could be possible that a process that has a high prcpumsec time could execute while processes with the lower prcpumsec time are sleeping, and since the higher process is the only ready process(apart from null) it will execute and further the gap between itself and the lower processes. If this is the case then it could be very difficult for these lower prcpumsec time processes to catch up to the higher one. This can be the case only for different values of loop1 loop2 for the 6 iobound processes. Intuitively it makes sense.

3 Half-half

As show in image we go back to the last answer where we experimented with different values of loop1 and loop2 for the all processes are iobound case and notice that the same happens for cpubound processes. Every time all iobound processes are sleeping, the cpubound process takes over and gets more clock cycles and then it always stays in the lead an eventually the cpubound processes finish faster than the iobound processes. This is normal. Actually, it is the preferred way an OS must execute. What is the alternative to this? A cPU bound process not executing while all such IO bound processes are sleeping. Why waste the clock cycles. However, as soon as an iobound process wakes up it is given priority since it was given lower cpu clock cycles over the cpubound processes.

Question 5: Dynamic workloads

My solution to this problem involves a very intuitive yet effective method to manage dynamic workloads. To revisit the problem we have long standing process who gets very heavily demoted by a newly entered process. So the process would be allowed to execute until its cputime alloted rises up to the long standing process. But this is clearly leading to starvation. So the solution is to demote priority of process for cputime used and promote priority for the amount of time it has spent on the readylist. Hence, every time resched is

invoked all the processes on the ready list are given 6ms of promotion. Here we are using the same ready list as provided originally but the key of a process when inserted is the negation of its cputime used. Hence, the lower the cputime used by processes the better the chance of it to be executed next. By adding 6ms on each resched call to ready list processes that haven't won the ongoing time quantum, they slowly move up the process queue to get a chance to execute next, a chance they could've never got without this strategy. Hence, a priority of process is defined by = wait-time - cpu-time used, where wait time is 6ms * (number of times resched was invoked while it was on ready list). The results are shown in the image below.
As we can see that process with pid 4 has CPU time 690 while processes with pid 8 and 9 are supposed to demote the other processes by a lot and hog the cpu until they get a similar amount of time as them 690ms. But after a while process 4's ReadyQ key is decremented because it has been rewarded for waiting. And even though pid 9 has cpu time of 481 and pid 4 process has 690 cpu time the position of process with pid 4 on readylist is higher than process with pid 9. Hence, we have shown that dynamic workloads can be scheduled fairly, to prevent starvation.

```
 69 ps               curr    20   28 0x0EFD8FFC 0x0EFD8DF8     8192          2
xsh $ ps
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size   CPU Time
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEF1C     8192 2147483647
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCAAC    16384          2
 22 cpubound 1       ready    1    3 0x0FDEFFFC 0x0FDEFCF0     1024       7062
 23 cpubound 2       ready    1    3 0x0FDEFBFC 0x0FDEF980     1024       7084
 24 cpubound 3       ready    1    3 0x0FDEF7FC 0x0FDEF584     1024       7085
 25 cpubound 4       ready    1    3 0x0FDEF3FC 0x0FDEF0EC     1024       7069
 26 cpubound 5       ready    1    3 0x0EFC8FFC 0x0EFC8CF0     1024       7071
 27 cpubound 6       ready    1    3 0x0EFC8BFC 0x0EFC88F0     1024       7072
 28 shell            recv    50    3 0x0EFC87FC 0x0EFC848C     8192          2
 70 ps               curr    20   28 0x0EFD8FFC 0x0EFD8DF8     8192          2
xsh $ ps
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size   CPU Time
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEF1C     8192 2147483647
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCAAC    16384          2
 22 cpubound 1       ready    1    3 0x0FDEFFFC 0x0FDEFCF0     1024       7176
 23 cpubound 2       ready    1    3 0x0FDEFBFC 0x0FDEF8EC     1024       7176
 24 cpubound 3       ready    1    3 0x0FDEF7FC 0x0FDEF4F0     1024       7177
 25 cpubound 4       ready    1    3 0x0FDEF3FC 0x0FDEF160     1024       7180
 26 cpubound 5       ready    1    3 0x0EFC8FFC 0x0EFC8CF0     1024       7180
 27 cpubound 6       ready    1    3 0x0EFC8BFC 0x0EFC88EC     1024       7180
 28 shell            recv    50    3 0x0EFC87FC 0x0EFC848C     8192          2
 71 ps               curr    20   28 0x0EFD8FFC 0x0EFD8DF8     8192          3
xsh $ ps
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size   CPU Time
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEF1C     8192 2147483647
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCAAC    16384          2
 22 cpubound 1       ready    1    3 0x0FDEFFFC 0x0FDEFCEC     1024       7252
 23 cpubound 2       ready    1    3 0x0FDEFBFC 0x0FDEF960     1024       7268
 24 cpubound 3       ready    1    3 0x0FDEF7FC 0x0FDEF4EC     1024       7256
 25 cpubound 4       ready    1    3 0x0FDEF3FC 0x0FDEF184     1024       7274
 26 cpubound 5       ready    1    3 0x0EFC8FFC 0x0EFC8CF0     1024       7261
 27 cpubound 6       ready    1    3 0x0EFC8BFC 0x0EFC8980     1024       7274
 28 shell            recv    50    3 0x0EFC87FC 0x0EFC848C     8192          2
 72 ps               curr    20   28 0x0EFD8FFC 0x0EFD8DF8     8192          3
xsh $ ps
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size   CPU Time
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEF1C     8192 2147483647
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCAAC    16384          2
 22 cpubound 1       ready    1    3 0x0FDEFFFC 0x0FDEFD84     1024       7389
 23 cpubound 2       ready    1    3 0x0FDEFBFC 0x0FDEF980     1024       7389
 24 cpubound 3       ready    1    3 0x0FDEF7FC 0x0FDEF580     1024       7390
 25 cpubound 4       ready    1    3 0x0FDEF3FC 0x0FDEF0EC     1024       7387
 26 cpubound 5       ready    1    3 0x0EFC8FFC 0x0EFC8CF0     1024       7391
 27 cpubound 6       ready    1    3 0x0EFC8BFC 0x0EFC88EC     1024       7391
 28 shell            recv    50    3 0x0EFC87FC 0x0EFC848C     8192          2
 73 ps               curr    20   28 0x0EFD8FFC 0x0EFD8DF8     8192          3
xsh $ ps
```

Figure 3: All processes are CPU bound

```
uter loop 36 | process priority is 1 | preempt is 30
iority is 1 | preempt is 30
ps
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size   CPU Time
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEEB8     8192 2147483647
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCAAC    16384          2
 22 shell            recv    50    3 0x0EFC8FFC 0x0EFC8C8C     8192          4
 23 iobound 1        sleep    1    3 0x0FDEFFFC 0x0FDEFF58     1024        307
 24 iobound 2        sleep    1    3 0x0FDEFBFC 0x0FDEFB58     1024        302
 25 iobound 3        sleep    1    3 0x0FDEF7FC 0x0FDEF758     1024        304
 26 iobound 4        sleep    1    3 0x0FDEF3FC 0x0FDEF358     1024        288
 27 iobound 5        sleep    1    3 0x0EFC6FFC 0x0EFC6F58     1024        306
 28 iobound 6        sleep    1    3 0x0EFC6BFC 0x0EFC6B58     1024        302
 50 ps               curr    20   22 0x0EFD8FFC 0x0EFD8DF8     8192          2
xsh $
 In iobound 4 |The PID is 26 | outer loop 37 | process priority is 1 | preempt is 30

 In iobound 6 |The PID is 28 | outer loop 37 | process priority is 1 | preempt is 30

 In iobound 2 |The PID is 24 | outer loop 37 | process priority is 1 | preempt is 30

 In iobound 3 |The PID is 25 | out
 In iobound 5 |The PID is 27 | outer loop 37 | process priority is 1 | preempt is 30

 In iobound 1 |The PID ier loop 37 | process priority is 1 | preempt is 30
s 23 | outer loop 37 | process priority is 1 | preempt is 30
ps
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size   CPU Time
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEEB8     8192 2147483647
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCAAC    16384          2
 22 shell            recv    50    3 0x0EFC8FFC 0x0EFC8C8C     8192          4
 23 iobound 1        sleep    1    3 0x0FDEFFFC 0x0FDEFF58     1024        316
 24 iobound 2        sleep    1    3 0x0FDEFBFC 0x0FDEFB58     1024        309
 25 iobound 3        sleep    1    3 0x0FDEF7FC 0x0FDEF758     1024        311
 26 iobound 4        ready    1    3 0x0FDEF3FC 0x0FDEF1D0     1024        300
 27 iobound 5        sleep    1    3 0x0EFC6FFC 0x0EFC6F58     1024        313
 28 iobound 6        sleep    1    3 0x0EFC6BFC 0x0EFC6B58     1024        309
 51 ps               curr    20   22 0x0EFD8FFC 0x0EFD8DF8     8192          2
xsh $ he PID is 26 | outer loop 38 | process priority is 1 | preempt is 30
```

Figure 4: All processes are IO bound

```
ps
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size   CPU Time
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEF1C       8192 2147483647
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCAAC      16384          2
  3 Main process     sleep   20    0 0x0EFD8FFC 0x0EFD8F74      65536          3
 22 iobound 4        sleep    1    3 0x0FDEFFFC 0x0FDEFF58       1024        151
 23 iobound 5        sleep    1    3 0x0FDEFBFC 0x0FDEFB58       1024         15
 24 iobound 6        sleep    1    3 0x0FDEF7FC 0x0FDEF758       1024          8
 25 cpubound 4       ready    1    3 0x0FDEF3FC 0x0FDEF0F0       1024        478
 26 cpubound 5       ready    1    3 0x0EFC8FFC 0x0EFC8D54       1024        479
 27 cpubound 6       ready    1    3 0x0EFC8BFC 0x0EFC88F0       1024        478
 28 shell            recv    50    3 0x0EFC87FC 0x0EFC848C       8192          1
 34 ps               curr    20   28 0x0EFC67FC 0x0EFC65B8       8192          2
xsh $ e PID is 24 | outer loop 1 | process priority is 1 | preempt is 30

 In iobound 5 |The PID is 23 | outer loop 2 | process priority is 1 | preempt is 30

 In iobound 4 |The PID is 22 | outer loop 3 | process priority is 1 | preempt is 30

 In iobound 4 |The PID is 22 | outer loop 4 | process priority is 1 | preempt is 30

 In iobound 5 |The PID is 23 | outer loop 3 | process priority is 1 | preempt is 30

 In iobound 6 |The PID is 24 | outer loop 2 | process priority is 1 | preempt is 30
p
 In iobound 4 |The PID is 22 | outer loop 5 | process priority is 1 | preempt is 30
s
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size   CPU Time
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEF1C       8192 2147483647
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCAAC      16384          2
  3 Main process     sleep   20    0 0x0EFD8FFC 0x0EFD8F74      65536          3
 22 iobound 4        sleep    1    3 0x0FDEFFFC 0x0FDEFF58       1024        173
 23 iobound 5        sleep    1    3 0x0FDEFBFC 0x0FDEFB58       1024         29
 24 iobound 6        sleep    1    3 0x0FDEF7FC 0x0FDEF758       1024         63
 25 cpubound 4       ready    1    3 0x0FDEF3FC 0x0FDEF0F0       1024        896
 26 cpubound 5       ready    1    3 0x0EFC8FFC 0x0EFC8CF0       1024        897
 27 cpubound 6       ready    1    3 0x0EFC8BFC 0x0EFC88EC       1024        900
 28 shell            recv    50    3 0x0EFC87FC 0x0EFC848C       8192          1
 35 ps               curr    20   28 0x0EFC67FC 0x0EFC65B8       8192          2
```

Figure 5: Half -Half

```
 -- LAB 2.5 ---
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size CPU time ReadyQ Posn ReadyQ Key
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ---------- ----------- -----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEF40       8192 2147483647           6  -2147483647
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCA9C      16384          2
  3 Main process     curr    20    0 0x0EFD8FFC 0x0EFD8AD0      65536         32
  4 cpubound 1       ready    1    3 0x0FDEFFFC 0x0FDEFD74       1024        690          3       -624
  5 cpubound 2       ready    1    3 0x0FDEFBFC 0x0FDEF970       1024        691          2       -619
  6 cpubound 3       ready    1    3 0x0FDEF7FC 0x0FDEF544       1024        702          5       -648
  7 cpubound 4       ready    1    3 0x0FDEF3FC 0x0FDEF174       1024        691          4       -631
  8 cpubound 5       ready    1    3 0x0EFC8FFC 0x0EFC8D44       1024        131          1       -119
  9 cpubound 6       ready    1    3 0x0EFC8BFC 0x0EFC8974       1024         91          0        -31
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size CPU time ReadyQ Posn ReadyQ Key
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ---------- ----------- -----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEF40       8192 2147483647           6  -2147483647
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCA9C      16384          2
  3 Main process     curr    20    0 0x0EFD8FFC 0x0EFD8AF0      65536        122
  4 cpubound 1       ready    1    3 0x0FDEFFFC 0x0FDEFD74       1024        690          3       -582
  5 cpubound 2       ready    1    3 0x0FDEFBFC 0x0FDEF970       1024        691          2       -577
  6 cpubound 3       ready    1    3 0x0FDEF7FC 0x0FDEF544       1024        702          5       -606
  7 cpubound 4       ready    1    3 0x0FDEF3FC 0x0FDEF174       1024        691          4       -589
  8 cpubound 5       ready    1    3 0x0EFC8FFC 0x0EFC8D74       1024        190          1       -160
  9 cpubound 6       ready    1    3 0x0EFC8BFC 0x0EFC8974       1024        181          0       -115
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size CPU time ReadyQ Posn ReadyQ Key
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ---------- ----------- -----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEF40       8192 2147483647           6  -2147483647
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCA9C      16384          2
  3 Main process     curr    20    0 0x0EFD8FFC 0x0EFD8AD0      65536        212
  4 cpubound 1       ready    1    3 0x0FDEFFFC 0x0FDEFD74       1024        690          3       -534
  5 cpubound 2       ready    1    3 0x0FDEFBFC 0x0FDEF970       1024        691          2       -529
  6 cpubound 3       ready    1    3 0x0FDEF7FC 0x0FDEF544       1024        702          5       -558
  7 cpubound 4       ready    1    3 0x0FDEF3FC 0x0FDEF174       1024        691          4       -541
  8 cpubound 5       ready    1    3 0x0EFC8FFC 0x0EFC8D74       1024        250          1       -220
  9 cpubound 6       ready    1    3 0x0EFC8BFC 0x0EFC8974       1024        301          0       -205
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size CPU time ReadyQ Posn ReadyQ Key
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ---------- ----------- -----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEF40       8192 2147483647           6  -2147483647
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCA9C      16384          2
  3 Main process     curr    20    0 0x0EFD8FFC 0x0EFD8AD0      65536        302
  4 cpubound 1       ready    1    3 0x0FDEFFFC 0x0FDEFD74       1024        690          3       -486
  5 cpubound 2       ready    1    3 0x0FDEFBFC 0x0FDEF970       1024        691          2       -481
  6 cpubound 3       ready    1    3 0x0FDEF7FC 0x0FDEF544       1024        702          5       -510
  7 cpubound 4       ready    1    3 0x0FDEF3FC 0x0FDEF174       1024        691          4       -493
  8 cpubound 5       ready    1    3 0x0EFC8FFC 0x0EFC8D70       1024        340          0       -274
  9 cpubound 6       ready    1    3 0x0EFC8BFC 0x0EFC8974       1024        391          1       -301
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size CPU time ReadyQ Posn ReadyQ Key
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ---------- ----------- -----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEF40       8192 2147483647           6  -2147483647
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCA9C      16384          2
  3 Main process     curr    20    0 0x0EFD8FFC 0x0EFD8AD0      65536        392
  4 cpubound 1       ready    1    3 0x0FDEFFFC 0x0FDEFD74       1024        690          2       -414
  5 cpubound 2       ready    1    3 0x0FDEFBFC 0x0FDEF974       1024        721          1       -409
  6 cpubound 3       ready    1    3 0x0FDEF7FC 0x0FDEF544       1024        702          5       -438
  7 cpubound 4       ready    1    3 0x0FDEF3FC 0x0FDEF174       1024        721          3       -421
  8 cpubound 5       ready    1    3 0x0EFC8FFC 0x0EFC8D74       1024        430          0       -394
  9 cpubound 6       ready    1    3 0x0EFC8BFC 0x0EFC8974       1024        481          4       -421
Pid Name             State Prio Ppid Stack Base Stack Ptr  Stack Size CPU time ReadyQ Posn ReadyQ Key
--- ---------------- ----- ---- ---- ---------- ---------- ---------- ---------- ----------- -----------
  0 prnull           ready    0    0 0x0EFDEFFC 0x0EFDEF40       8192 2147483647           6  -2147483647
  1 rdsproc          wait   200    0 0x0EFDCFFC 0x0EFDCA9C      16384          2
  3 Main process     curr    20    0 0x0EFD8FFC 0x0EFD8AD0      65536        482
  4 cpubound 1       ready    1    3 0x0FDEFFFC 0x0FDEFD70       1024        720          3       -666
  5 cpubound 2       ready    1    3 0x0FDEFBFC 0x0FDEF974       1024        721          2       -661
  6 cpubound 3       ready    1    3 0x0FDEF7FC 0x0FDEF574       1024        731          5       -701
  7 cpubound 4       ready    1    3 0x0FDEF3FC 0x0FDEF174       1024        721          4       -673
  8 cpubound 5       ready    1    3 0x0EFC8FFC 0x0EFC8D74       1024        550          0       -484
```

Figure 6: Dynamic Workloads