LAB 3

Anirudh Pal (pal5)

Part 3.2:

Output:

```
Testing CFS with 4 CPU Bound and 4 I/O Bound Procs.

PID: 4, proctype: 0, CLKTIMEMILLI: 5087ms, CPU Gross Time: 1280ms, CPU Wait Time: 3799ms,
CPU Wait Count: 101, CPU Avg Wait Time: 37ms, pvirtcpu: 1276ms, Prio: 31491

PID: 6, proctype: 0, CLKTIMEMILLI: 5116ms, CPU Gross Time: 1280ms, CPU Wait Time: 3803ms,
CPU Wait Count: 103, CPU Avg Wait Time: 36ms, pvirtcpu: 1297ms, Prio: 31470

PID: 10, proctype: 0, CLKTIMEMILLI: 5124ms, CPU Gross Time: 1280ms, CPU Wait Time: 3811ms,
CPU Wait Count: 99, CPU Avg Wait Time: 38ms, pvirtcpu: 1298ms, Prio: 31469

PID: 8, proctype: 0, CLKTIMEMILLI: 5132ms, CPU Gross Time: 1281ms, CPU Wait Time: 3818ms,
CPU Wait Count: 101, CPU Avg Wait Time: 37ms, pvirtcpu: 1299ms, Prio: 31468

PID: 5, proctype: 1, CLKTIMEMILLI: 15107ms, CPU Gross Time: 0ms, CPU Wait Time: 74ms, CPU
Wait Count: 301, CPU Avg Wait Time: 0ms, pvirtcpu: 0ms, Prio: 32767

PID: 9, proctype: 1, CLKTIMEMILLI: 15108ms, CPU Gross Time: 0ms, CPU Wait Time: 75ms, CPU
Wait Count: 301, CPU Avg Wait Time: 0ms, pvirtcpu: 0ms, Prio: 32767

PID: 7, proctype: 1, CLKTIMEMILLI: 15109ms, CPU Gross Time: 0ms, CPU Wait Time: 76ms, CPU
Wait Count: 301, CPU Avg Wait Time: 0ms, pvirtcpu: 1297ms, Prio: 31470

PID: 11, proctype: 1, CLKTIMEMILLI: 15133ms, CPU Gross Time: 0ms, CPU Wait Time: 100ms, CPU
Wait Count: 301, CPU Avg Wait Time: 0ms, pvirtcpu: 0ms, Prio: 32767
```

Compare with R3: Instead of the priority jumping between two levels based on I/O or CPU bound behaviour, we instead see that priority is dependent on CPU usage. I/O procs are getting higher priority since they use very little CPU, whereas the CPU bound procs have to lower their priority based on CPU usage. The priorities do still remain close since they are normalized by creation (max current usage) and wakeup (min current usage). The average wait time for I/O bound is far less than CPU bound.

Part 3.3:

Output:

```
Testing CFS with Dynamic 4 CPU Bound and 4 I/O Bound Procs.

PID: 12, proctype: 0, CLKTIMEMILLI: 26543ms, CPU Gross Time: 1280ms, CPU Wait Time: 225ms,
CPU Wait Count: 23, CPU Avg Wait Time: 9ms, pvirtcpu: 1273ms, Prio: 31494

PID: 14, proctype: 0, CLKTIMEMILLI: 28099ms, CPU Gross Time: 1280ms, CPU Wait Time: 731ms,
CPU Wait Count: 61, CPU Avg Wait Time: 11ms, pvirtcpu: 2319ms, Prio: 30448

PID: 16, proctype: 0, CLKTIMEMILLI: 29656ms, CPU Gross Time: 1281ms, CPU Wait Time: 1287ms,
CPU Wait Count: 63, CPU Avg Wait Time: 20ms, pvirtcpu: 3080ms, Prio: 29687

PID: 18, proctype: 0, CLKTIMEMILLI: 30162ms, CPU Gross Time: 1280ms, CPU Wait Time: 794ms,
CPU Wait Count: 82, CPU Avg Wait Time: 9ms, pvirtcpu: 3599ms, Prio: 29168

PID: 13, proctype: 1, CLKTIMEMILLI: 40612ms, CPU Gross Time: 0ms, CPU Wait Time: 24ms, CPU
Wait Count: 301, CPU Avg Wait Time: 0ms, pvirtcpu: 0ms, Prio: 32767

PID: 15, proctype: 1, CLKTIMEMILLI: 41613ms, CPU Gross Time: 0ms, CPU Wait Time: 25ms, CPU
Wait Count: 301, CPU Avg Wait Time: 0ms, pvirtcpu: 0ms, Prio: 32767

PID: 17, proctype: 1, CLKTIMEMILLI: 42636ms, CPU Gross Time: 0ms, CPU Wait Time: 48ms, CPU
Wait Count: 301, CPU Avg Wait Time: 0ms, pvirtcpu: 3573ms, Prio: 29194

PID: 19, proctype: 1, CLKTIMEMILLI: 43636ms, CPU Gross Time: 0ms, CPU Wait Time: 48ms, CPU
Wait Count: 301, CPU Avg Wait Time: 0ms, pvirtcpu: 0ms, Prio: 32767
```

Compare with Static Load: When compared to the static load we can see that the normalization during creation and wakeup is actually working. When the CPU bound processes are created, they get the max cpu usage currently in the readylist which leads to an increasing *pvirtcpu* for successive CPU bound creations leading to lower priority. This is done so as to prevent new procs from hogging the CPU. When it comes to I/O bound processes, they get the min cpu usage currently in the readylist (probably of *main()* proc) when they wakeup. This lets them have similar priorities. This is done to stop I/O Bound procs from hogging CPU when they change behaviour.

NOTE: For Part 4.5.1; XINUSCHED was set to 0, INITPRIO was set to 0x7FFF and create() had max priority restriction for TS procs removed. This was done in order to allow main() to be created with high priority so it can start all RT procs and then die.

Part 4.5.1:

Output:

```
Testing RMS with 4 Real Time Procs

PID: 7, x: 25, y: 800, Period Number: 0, Remaining Period: 775

PID: 6, x: 50, y: 900, Period Number: 0, Remaining Period: 850

PID: 4, x: 300, y: 999, Period Number: 0, Remaining Period: 699

PID: 5, x: 100, y: 1000, Period Number: 0, Remaining Period: 900

PID: 7, x: 25, y: 800, Period Number: 1, Remaining Period: 775

PID: 6, x: 50, y: 900, Period Number: 1, Remaining Period: 850

PID: 4, x: 300, y: 999, Period Number: 1, Remaining Period: 699

PID: 5, x: 100, y: 1000, Period Number: 1, Remaining Period: 900

PID: 7, x: 25, y: 800, Period Number: 2, Remaining Period: 775

PID: 6, x: 50, y: 900, Period Number: 2, Remaining Period: 850

PID: 4, x: 300, y: 999, Period Number: 2, Remaining Period: 699

PID: 7, x: 25, y: 800, Period Number: 3, Remaining Period: 775

PID: 5, x: 100, y: 1000, Period Number: 2, Remaining Period: 874

PID: 6, x: 50, y: 900, Period Number: 3, Remaining Period: 850

PID: 7, x: 25, y: 800, Period Number: 4, Remaining Period: 775

PID: 4, x: 300, y: 999, Period Number: 3, Remaining Period: 673

PID: 5, x: 100, y: 1000, Period Number: 3, Remaining Period: 900

PID: 6, x: 50, y: 900, Period Number: 4, Remaining Period: 850

PID: 7, CLKTIMEMILLI: 4007ms, CPU Gross Time: 130ms, CPU Wait Time: 0ms, CPU Wait Count: 6,
CPU Avg Wait Time: 0ms, Prio: 29200

PID: 4, x: 300, y: 999, Period Number: 4, Remaining Period: 699

PID: 5, x: 100, y: 1000, Period Number: 4, Remaining Period: 900

PID: 6, CLKTIMEMILLI: 4533ms, CPU Gross Time: 255ms, CPU Wait Time: 26ms, CPU Wait Count: 6,
```

```
CPU Avg Wait Time: 4ms, Prio: 29100

PID: 4, CLKTIMEMILLI: 5079ms, CPU Gross Time: 1505ms, CPU Wait Time: 103ms, CPU Wait Count:
7, CPU Avg Wait Time: 14ms, Prio: 29001

PID: 5, CLKTIMEMILLI: 5408ms, CPU Gross Time: 505ms, CPU Wait Time: 427ms, CPU Wait Count:
7, CPU Avg Wait Time: 61ms, Prio: 29000
```

Analysis: All the RT procs received the periods that they requested and their priority was inversely proportional to the assigned period. They also finished the computation within the defined period and this indicates that the admission control was appropriate for this workload.

NOTE: For Part 4.5.2; XINUSCHED was set to 1, INITPRIO was set back to 20 and create() had max priority restriction for TS procs added. This was done in order to allow R3 to work properly as the assigned priority is discarded and the behaviour determines the priority of a TS proc. This is also the state in which the Lab was turned in as there was no indication of integrating CFS and RMS together. Before submission all the testcode was also commented out of main().

Part 4.5.2:

Output:

```
Testing R3 with RMS

PID: 2, x: 300, y: 999, Period Number: 0, Remaining Period: 699

PID: 8, x: 100, y: 1000, Period Number: 0, Remaining Period: 900

PID: 9, x: 50, y: 900, Period Number: 0, Remaining Period: 850

PID: 10, x: 25, y: 800, Period Number: 0, Remaining Period: 775

PID: 2, x: 300, y: 999, Period Number: 1, Remaining Period: 699

PID: 8, x: 100, y: 1000, Period Number: 1, Remaining Period: 900

PID: 9, x: 50, y: 900, Period Number: 1, Remaining Period: 850

PID: 10, x: 25, y: 800, Period Number: 1, Remaining Period: 775
```

```
PID: 2, x: 300, y: 999, Period Number: 2, Remaining Period: 699

PID: 10, x: 25, y: 800, Period Number: 2, Remaining Period: 775

PID: 9, x: 50, y: 900, Period Number: 2, Remaining Period: 850

PID: 8, x: 100, y: 1000, Period Number: 2, Remaining Period: 849

PID: 10, x: 25, y: 800, Period Number: 3, Remaining Period: 775

PID: 9, x: 50, y: 900, Period Number: 3, Remaining Period: 850

PID: 2, x: 300, y: 999, Period Number: 3, Remaining Period: 622

PID: 8, x: 100, y: 1000, Period Number: 3, Remaining Period: 900

PID: 10, x: 25, y: 800, Period Number: 4, Remaining Period: 775

PID: 9, x: 50, y: 900, Period Number: 4, Remaining Period: 850

PID: 2, x: 300, y: 999, Period Number: 4, Remaining Period: 648

PID: 8, x: 100, y: 1000, Period Number: 4, Remaining Period: 900

PID: 10, CLKTIMEMILLI: 5011ms, CPU Gross Time: 130ms, CPU Wait Time: 0ms, CPU Wait
Count: 6, CPU Avg Wait Time: 0ms, Prio: 29200

PID: 2, CLKTIMEMILLI: 5253ms, CPU Gross Time: 1505ms, CPU Wait Time: 128ms, CPU Wait Count:
9, CPU Avg Wait Time: 14ms, Prio: 29001

PID: 9, CLKTIMEMILLI: 5360ms, CPU Gross Time: 255ms, CPU Wait Time: 26ms, CPU Wait Count: 7,
CPU Avg Wait Time: 3ms, Prio: 29100

PID: 8, CLKTIMEMILLI: 5659ms, CPU Gross Time: 505ms, CPU Wait Time: 128ms, CPU Wait Count:
8, CPU Avg Wait Time: 16ms, Prio: 29000

PID: 4, proctype: 0, CLKTIMEMILLI: 7275ms, CPU Gross Time: 1282ms, CPU Wait Time: 5985ms,
CPU Wait Count: 53, CPU Avg Wait Time: 112ms, pvirtcpu: 0ms, Prio: 20

PID: 5, proctype: 0, CLKTIMEMILLI: 7523ms, CPU Gross Time: 1281ms, CPU Wait Time: 6209ms,
CPU Wait Count: 55, CPU Avg Wait Time: 112ms, pvirtcpu: 0ms, Prio: 20

PID: 6, proctype: 0, CLKTIMEMILLI: 7533ms, CPU Gross Time: 1282ms, CPU Wait Time: 6168ms,
CPU Wait Count: 54, CPU Avg Wait Time: 114ms, pvirtcpu: 0ms, Prio: 20

PID: 7, proctype: 0, CLKTIMEMILLI: 7537ms, CPU Gross Time: 1282ms, CPU Wait Time: 6097ms,
CPU Wait Count: 53, CPU Avg Wait Time: 115ms, pvirtcpu: 0ms, Prio: 20
```

Analysis: All the RT procs received the periods that they requested and their priority was inversely proportional to the assigned period. They also finished the computation within the defined period. The TS procs only got a chance to run when the RT procs were all in a sleep state or when they all terminated. This means the RMS mechanism is working.

BONUS:

Response: One of the complications with EDF scheduler is that you need to monitor which RMS proc is reaching near its period end and change is priority dynamically. This can be done through the use of a complicated setup, but is more practical with something like a scheduler daemon that runs computations periodically similar to UNIX Solaris. Another problem is that EDF admission control does not account kernel overhead and timer accuracy which are both problems with XINU since the kernel would have to run periodically. There is also the problem of XINU dropping external interrupts due to its excessive use of Interrupt disable. All these problems would have to be addressed and then with an active scheduler and admission control that accounts for overhead it would be possible to implement a true EDF scheduler.