CS 367 Project #0 - Fall 2020:
CPU Process Scheduler
Sample Runs of Provided Traces

1 Reading the Output

Output from the scheduler, shown on the right, shows the state of the OS simulator during each one of the loops.

Each block shows five pieces of information.

1. Starting Time for that block.
2. Which, if any, process is taken off of the CPU and returned back to the scheduler (using your `scheduler_add` function).
3. Which action was processed for that time.
4. Fourth, it shows the state of all three lists as they are after the action finishes, but before a process is selected to be run.
5. Finally, it shows which process, if any, were selected to run on the CPU.

```
.=============
| Starting Time:  1
+------------
| Nothing to Return to Scheduler
+------------
| Process Starting
|      PID:   1 Time Remaining:  1 Priority (Cur: 130, Base:
130) Command: ls  Flags:[C     ]
+------------
| Ready Queue:    1 Processes
|      PID:   1 Time Remaining:  1 Priority (Cur: 130, Base:
130) Command: ls  Flags:[ R    ]
| Stopped Queue:  0 Processes
|      None
| Defunct Queue:  0 Processes
|      None
+------------
| Selecting to Run on the CPU
|      PID:   1 Time Remaining:  1 Priority (Cur: 130, Base:
130) Command: ls  Flags:[ R    ]
\=============

.=============
| Starting Time:  2
```

The process information within any of these informational areas shows you the PID for that process, how much time is remaining, the time it was last run, and the name of the command itself.

The flags are also listed for each process in the list, using the following codes:

- C – Created
- R – Ready
- T – Stopped
- Z – Defunct
- S – Sudo (Super User) Privileges

Example: The process selected in the sample on this page is in the Ready state.

Description of the Simulator Steps, Showing the Order the Output Prints in.

1. Starting Time is Printed
2. Process is removed from the CPU and Printed.
3. Process that was removed from the CPU is added using your **`scheduler_add`** (nothing printed)
4. Action is read from the Trace File and Printed.
5. Action is parsed and executed by calling one of your functions. (nothing printed)
6. Current contents of each queue is Printed.
7. **`scheduler_select`** is called to remove a selected process from ready queue. (nothing printed)
8. **`scheduler_select`** changes the **`cur_priority`** back to **`base_priority.`** (nothing printed)
9. **`scheduler_select`** decrements **`cur_priority`** of each ready queue process. (nothing printed)
10. Selected process is then dispatched to the CPU for execution and Printed.

```
.=============
| Starting Time:  2
+------------
| Returning Process (PID: 1) to Scheduler
|      PID:   1 Time Remaining:  8 Priority (Cur: 105, Base: 105) Command: ls  Flags:[ R    ]
+------------
| Process Starting
|      PID:   2 Time Remaining: 10 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[C     ]
+------------
| Ready Queue:    2 Processes
|      PID:   2 Time Remaining: 10 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R    ]
|      PID:   1 Time Remaining:  8 Priority (Cur: 105, Base: 105) Command: ls  Flags:[ R    ]
| Stopped Queue:  0 Processes
|      None
| Defunct Queue:  0 Processes
|      None
+------------
| Selecting to Run on the CPU
|      PID:   2 Time Remaining: 10 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R    ]
\=============
```

Example: Reading this sample output above using those same steps.

1. This sample output above shows all of the events that happen for Time 2
2. The process with PID 1 ran last time, so it is removed from the CPU and printed to the screen.
    a. It has a Time Remaining of 8, so it needs to run on the CPU for 8 more times to finish.
3. The simulator calls your **`scheduler_add`** on that process (PID 1).
    a. Since it has a Time Remaining greater than 0, it will go back to the Ready Queue.
4. The next section says, 'Process Starting', meaning the next action on the trace file was to create a new process. The simulator will prepare all of the information from the tracefile to use.
5. The simulator calls **`scheduler_generate`** to create the process struct, followed by **`scheduler_add`**.
    a. Since it is created with state **`CREATE`**, it will go in to your Ready Queue as a **`READY`** state process.
    b. All inserts are to the front, so this will be in front of the other process. *(head -> PID2 -> PID1)*
6. The simulator then prints the status of all queues.
    a. You can see PID 2 as the first process, followed by PID 1 in the Ready Queue.
7. The simulator calls your **`scheduler_select`** function to get the next process to run.
    a. Your function will remove the process PID 2 because it's Current Priority (Cur:) is lowest.
8. Your **`scheduler_select`** will then reset PID 2's Current Priority to its Base Priority of 100.
9. Your **`scheduler_select`** will decrement PID 1's Current Priority to 104, then return the selected one.
10. Finally, the simulator prints the information on PID 2, which you just returned from **`scheduler_select`**.

Note: The queues are printed in Step 6 before the Current Priorities are modified by your **`scheduler_select`**.

2 traces/trace1.dat

Notes: This just exists on the first loop, at time 1. No processes are created, so all of the Queues should be empty.

2.1 Tracefile

```
exit
```

2.2 Sample Output from Scheduler

```
kandrea@zeus-1:handout$ ./scheduler traces/trace1.dat
.=============
| Starting Time:  1
+------------
| Nothing to Return to Scheduler
+------------
| Exit Selected
+------------
| Ready Queue:    0 Processes
|      None
| Stopped Queue:  0 Processes
|      None
| Defunct Queue:  0 Processes
|      None
+------------
| Selecting to Run on the CPU
|      None
\=============

We're done!
```

3 traces/trace2.dat

Notes: This starts a new process at time 1 (ls) and starts it with PID = 1, `time_remaining` = 1, `base_priority` = 130, and `exit_code` = 1.

At Time 1, there's nothing currently running to be returned. For the action, it shows...

- A new process with PID 1 is starting up (initialized to `CREATED`).
- For the linked lists, you can see that this new process was moved into the Ready Queue and you changed its status to `READY`).
- You can finally see that your `scheduler_select` function properly chose the Process with PID 1 to run next because it had the lowest `cur_priority` (Shown in the output as Cur:).

This shows that there is 1 time remaining at the end of the scheduling phase.

At Time 2, you can see...

- The process you previously selected with PID 1 has run for 1 time unit (it has 0 time remaining) and is unloaded and returned to the scheduler through your `scheduler_add` function.
- The action for time 2 is exit, so that is helpfully printed in the action area.
- Following this, you can see that when the process was handled by your `scheduler_add` function, you noticed there was 0 `time_remaining`, so you changed its state to `DEFUNCT` and put it in the Defunct List. Finally, nothing is in the ready list, so nothing is scheduled to run.

At this point, the scheduler ends.

< Tracefile and Expected Output on the Next Page >

3.1 Tracefile

```
ls [1,1,130,1]
exit
```

3.2 Sample Output from Scheduler

```
kandrea@zeus-1:handout$ ./scheduler traces/trace2.dat
.=============
| Starting Time:  1
+------------
| Nothing to Return to Scheduler
+------------
| Process Starting
|      PID:   1 Time Remaining:  1 Priority (Cur: 130, Base: 130) Command: ls  Flags:[C     ]
+------------
| Ready Queue:    1 Processes
|      PID:   1 Time Remaining:  1 Priority (Cur: 130, Base: 130) Command: ls  Flags:[ R    ]
| Stopped Queue:  0 Processes
|      None
| Defunct Queue:  0 Processes
|      None
+------------
| Selecting to Run on the CPU
|      PID:   1 Time Remaining:  1 Priority (Cur: 130, Base: 130) Command: ls  Flags:[ R    ]
\=============

.=============
| Starting Time:  2
+------------
| Returning Process (PID: 1) to Scheduler
|      PID:   1 Time Remaining:  0 Priority (Cur: 130, Base: 130) Command: ls  Flags:[ R    ]
+------------
| Exit Selected
+------------
| Ready Queue:    0 Processes
|      None
| Stopped Queue:  0 Processes
|      None
| Defunct Queue:  1 Processes
|      PID:   1 Time Remaining:  0 Priority (Cur: 130, Base: 130) Command: ls  Flags:[   Z  ]
+------------
| Selecting to Run on the CPU
|      None
\=============

We're done!
```

4 traces/trace3.dat

Notes: This starts a new process at time 1 (ls) and starts it with PID = 1, `time_remaining` = 1, `base_priority` = 130, and `exit_code` = 1.

At Time 1, there's nothing currently running to be returned. For the action, it shows...

- A new process with PID 1 is starting up (initialized to `CREATED`).
- For the linked lists, you can see that this new process was moved into the Ready Queue and you changed its status to `READY`).
- You can finally see that your `scheduler_select` function properly chose the Process with PID 1 to run next because it had the lowest `cur_priority` (Shown in the output as Cur:).

This shows that there is 1 time remaining at the end of the scheduling phase.

At Time 2, you can see...

- The process you previously selected with PID 1 has run for 1 time unit (it has 0 time remaining) and is unloaded and returned to the scheduler through your `scheduler_add` function.
- The action for time 2 is reap on PID 1, so this will reap the process in the Defunct Queue.
- Unlike the previous trace, since it was reaped, when the time 2 ends and the output is printed, there is no process in the Defunct Queue! So, from this point forward, all queues will be empty.

At Time 3, exit is the selected action, so at this point, the scheduler ends.

4.1 Tracefile

```
ls [1,1,130,1]
reap 1
exit
```

< Expected Output on the Next Page >

4.2 Sample Output from Scheduler

```
kandrea@zeus-1:handout$ ./scheduler traces/trace3.dat
.=============
| Starting Time:  1
+------------
| Nothing to Return to Scheduler
+------------
| Process Starting
| PID:   1 Time Remaining:  1 Priority (Cur: 130, Base: 130) Command: ls  Flags:[C     ]
+------------
| Ready Queue:    1 Processes
| PID:   1 Time Remaining:  1 Priority (Cur: 130, Base: 130) Command: ls  Flags:[ R    ]
| Stopped Queue:  0 Processes
| None
| Defunct Queue:  0 Processes
| None
+------------
| Selecting to Run on the CPU
| PID:   1 Time Remaining:  1 Priority (Cur: 130, Base: 130) Command: ls  Flags:[ R    ]
\=============

.=============
| Starting Time:  2
+------------
| Returning Process (PID: 1) to Scheduler
| PID:   1 Time Remaining:  0 Priority (Cur: 130, Base: 130) Command: ls  Flags:[ R    ]
+------------
| Reaping Process (PID:  1). Exited with code: 1
+------------
| Ready Queue:    0 Processes
| None
| Stopped Queue:  0 Processes
| None
| Defunct Queue:  0 Processes
| None
+------------
| Selecting to Run on the CPU
| None
\=============

.=============
| Starting Time:  3
+------------
| Nothing to Return to Scheduler
+------------
| Exit Selected
+------------
| Ready Queue:    0 Processes
| None
| Stopped Queue:  0 Processes
| None
| Defunct Queue:  0 Processes
| None
+------------
| Selecting to Run on the CPU
| None
\=============

We're done!
```

5 traces/trace4.dat

Notes: This trace demonstrates the starvation protection using aging we have in scheduling. The scheduling algorithm you implemented chooses the process with the lowest `cur_priority` from the ready queue (with the process closest to the head of the list to break any ties). Once that occurs and the process is removed from the queue, then all remaining processes have 1 decremented from their `cur_priority` to simulate aging.

So, for this trace, two processes are created. PID 1 (ls) will run for 9 time units, and PID 2 (sudo pwd) will run for 10 time units. Since PID 1 is the only process created in Time 1, it will be selected to run.

After that, since PID 2 has the smallest `cur_priority`, it will be selected to run during Times 2, 3, 4, 5, 6, and 7. Each time PID 1 is not chosen and left behind in the Ready Queue, its `cur_priority` decreases by 1. So, when the Ready Queue is printed in Time 8, you have PID 1 with `cur_priority` equal to 99, which is lower than PID 2, so finally PID 1 will get to run again. However, as soon as it is selected to run, it's `cur_priority` is reset to equal its `base_priority.`

At time 8, PID 1 finally runs, which you can see at the end of the output, with its `cur_priority` now back up to 105, which is its `base_priority`. Note that when PID 1 is selected, PID 2 will have its priority decrement by 1 now to 99! You can see this in Time 9.

At time 9, we go through selection again and see that PID 2 has the lowest `cur_priority` again at 99, so it gets selected and has its `cur_priority` reset to its `base_priority` of 100.

Time 10 and 11 continue from here, where you see PID 1 slowly getting its `cur_priority` decremented whenever PID 2 is selected, so it will slowly wait its turn until its `cur_priority` is low enough to run again.

This trace demonstrates that even when you have a process with a much better priority, the other processes can still make sure they get some occasional access to the CPU too!

5.1 Tracefile

```
ls [1,9,105,1]
sudo pwd [2,10,100,2]
pass
pass
pass
pass
pass
pass
pass
pass
pass
exit
```

5.2 Sample Output from Scheduler

```
kandrea@zeus-1:handout$ ./scheduler traces/trace4.dat
.=============
| Starting Time:  1
+------------
| Nothing to Return to Scheduler
+------------
| Process Starting
|      PID:   1 Time Remaining:  9 Priority (Cur: 105, Base: 105) Command: ls  Flags:[C     ]
+------------
| Ready Queue:    1 Processes
|      PID:   1 Time Remaining:  9 Priority (Cur: 105, Base: 105) Command: ls  Flags:[ R    ]
| Stopped Queue:  0 Processes
|      None
| Defunct Queue:  0 Processes
|      None
+------------
| Selecting to Run on the CPU
|      PID:   1 Time Remaining:  9 Priority (Cur: 105, Base: 105) Command: ls  Flags:[ R    ]
\=============

.=============
| Starting Time:  2
+------------
| Returning Process (PID: 1) to Scheduler
|      PID:   1 Time Remaining:  8 Priority (Cur: 105, Base: 105) Command: ls  Flags:[ R    ]
+------------
| Process Starting
|      PID:   2 Time Remaining: 10 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[C    S]
+------------
| Ready Queue:    2 Processes
|      PID:   2 Time Remaining: 10 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
|      PID:   1 Time Remaining:  8 Priority (Cur: 105, Base: 105) Command: ls  Flags:[ R    ]
| Stopped Queue:  0 Processes
|      None
| Defunct Queue:  0 Processes
|      None
+------------
| Selecting to Run on the CPU
|      PID:   2 Time Remaining: 10 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
\=============

.=============
| Starting Time:  3
+------------
| Returning Process (PID: 2) to Scheduler
|      PID:   2 Time Remaining:  9 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
+------------
| Passing (No Action)
+------------
| Ready Queue:    2 Processes
|      PID:   2 Time Remaining:  9 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
|      PID:   1 Time Remaining:  8 Priority (Cur: 104, Base: 105) Command: ls  Flags:[ R    ]
| Stopped Queue:  0 Processes
|      None
| Defunct Queue:  0 Processes
|      None
+------------
| Selecting to Run on the CPU
```

```
|      PID:   2 Time Remaining:  9 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
\=============

.=============
| Starting Time:  4
+------------
| Returning Process (PID: 2) to Scheduler
|      PID:   2 Time Remaining:  8 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
+------------
| Passing (No Action)
+------------
| Ready Queue:    2 Processes
|      PID:   2 Time Remaining:  8 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
|      PID:   1 Time Remaining:  8 Priority (Cur: 103, Base: 105) Command: ls  Flags:[ R    ]
| Stopped Queue:  0 Processes
|      None
| Defunct Queue:  0 Processes
|      None
+------------
| Selecting to Run on the CPU
|      PID:   2 Time Remaining:  8 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
\=============

.=============
| Starting Time:  5
+------------
| Returning Process (PID: 2) to Scheduler
|      PID:   2 Time Remaining:  7 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
+------------
| Passing (No Action)
+------------
| Ready Queue:    2 Processes
|      PID:   2 Time Remaining:  7 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
|      PID:   1 Time Remaining:  8 Priority (Cur: 102, Base: 105) Command: ls  Flags:[ R    ]
| Stopped Queue:  0 Processes
|      None
| Defunct Queue:  0 Processes
|      None
+------------
| Selecting to Run on the CPU
|      PID:   2 Time Remaining:  7 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
\=============

.=============
| Starting Time:  6
+------------
| Returning Process (PID: 2) to Scheduler
|      PID:   2 Time Remaining:  6 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
+------------
| Passing (No Action)
+------------
| Ready Queue:    2 Processes
|      PID:   2 Time Remaining:  6 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
|      PID:   1 Time Remaining:  8 Priority (Cur: 101, Base: 105) Command: ls  Flags:[ R    ]
| Stopped Queue:  0 Processes
|      None
| Defunct Queue:  0 Processes
|      None
+------------
| Selecting to Run on the CPU
|      PID:   2 Time Remaining:  6 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
```

```
\=============

.=============
| Starting Time:  7
+------------
| Returning Process (PID: 2) to Scheduler
|      PID:   2 Time Remaining:  5 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
+------------
| Passing (No Action)
+------------
| Ready Queue:    2 Processes
|      PID:   2 Time Remaining:  5 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
|      PID:   1 Time Remaining:  8 Priority (Cur: 100, Base: 105) Command: ls  Flags:[ R    ]
| Stopped Queue:  0 Processes
|      None
| Defunct Queue:  0 Processes
|      None
+------------
| Selecting to Run on the CPU
|      PID:   2 Time Remaining:  5 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
\=============

.=============
| Starting Time:  8
+------------
| Returning Process (PID: 2) to Scheduler
|      PID:   2 Time Remaining:  4 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
+------------
| Passing (No Action)
+------------
| Ready Queue:    2 Processes
|      PID:   2 Time Remaining:  4 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
|      PID:   1 Time Remaining:  8 Priority (Cur: 99, Base: 105) Command: ls  Flags:[ R    ]
| Stopped Queue:  0 Processes
|      None
| Defunct Queue:  0 Processes
|      None
+------------
| Selecting to Run on the CPU
|      PID:   1 Time Remaining:  8 Priority (Cur: 105, Base: 105) Command: ls  Flags:[ R    ]
\=============

.=============
| Starting Time:  9
+------------
| Returning Process (PID: 1) to Scheduler
|      PID:   1 Time Remaining:  7 Priority (Cur: 105, Base: 105) Command: ls  Flags:[ R    ]
+------------
| Passing (No Action)
+------------
| Ready Queue:    2 Processes
|      PID:   1 Time Remaining:  7 Priority (Cur: 105, Base: 105) Command: ls  Flags:[ R    ]
|      PID:   2 Time Remaining:  4 Priority (Cur: 99, Base: 100) Command: pwd  Flags:[ R   S]
| Stopped Queue:  0 Processes
|      None
| Defunct Queue:  0 Processes
|      None
+------------
| Selecting to Run on the CPU
|      PID:   2 Time Remaining:  4 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
\=============
```

```
.=============
| Starting Time: 10
+------------
| Returning Process (PID: 2) to Scheduler
|      PID:   2 Time Remaining:  3 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
+------------
| Passing (No Action)
+------------
| Ready Queue:    2 Processes
|      PID:   2 Time Remaining:  3 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
|      PID:   1 Time Remaining:  7 Priority (Cur: 104, Base: 105) Command: ls  Flags:[ R    ]
| Stopped Queue:  0 Processes
|      None
| Defunct Queue:  0 Processes
|      None
+------------
| Selecting to Run on the CPU
|      PID:   2 Time Remaining:  3 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
\=============

.=============
| Starting Time: 11
+------------
| Returning Process (PID: 2) to Scheduler
|      PID:   2 Time Remaining:  2 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
+------------
| Exit Selected
+------------
| Ready Queue:    2 Processes
|      PID:   2 Time Remaining:  2 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
|      PID:   1 Time Remaining:  7 Priority (Cur: 103, Base: 105) Command: ls  Flags:[ R    ]
| Stopped Queue:  0 Processes
|      None
| Defunct Queue:  0 Processes
|      None
+------------
| Selecting to Run on the CPU
|      PID:   2 Time Remaining:  2 Priority (Cur: 100, Base: 100) Command: pwd  Flags:[ R   S]
\=============
We're done!
```

6 traces/trace5.dat

Notes: This trace demonstrates stopping and continuing a process.

For this trace, you have ls created with PID 1 and a `time_remaining` of 2. After Time 1, this process has been selected and run once. At Time 2, it has `time_remaining` of 1 and is moved back to the Ready Queue. The action now is to STOP PID 1, which causes the simulator to call your `scheduler_stop` function. Your function will go into the ready queue, remove PID 1, change its state to `STOPPED`, then move it to the stopped queue.

At the end of Time 2, there are no processes in the ready queue, so nothing is scheduled. Likewise with Time 3, which leaves the CPU idle.

At Time 4, CONT is sent, which calls your `scheduler_continue` function. PID 1 is removed from the stopped queue, has its state changed to `READY`, and moved into the ready queue. At this point, the scheduler will see it in the ready queue and schedule it to run.

At Time 5, it finishes running and moves to the defunct queue as normal.

6.1 Tracefile

```
ls [1,2,130,1]
kill -STOP 1
pass
kill -CONT 1
exit
```

6.2 Sample Output from Scheduler

```
kandrea@zeus-1:handout$ ./scheduler traces/trace5.dat
.=============
| Starting Time:  1
+------------
| Nothing to Return to Scheduler
+------------
| Process Starting
| PID:   1 Time Remaining:  2 Priority (Cur: 130, Base: 130) Command: ls  Flags:[C     ]
+------------
| Ready Queue:    1 Processes
| PID:   1 Time Remaining:  2 Priority (Cur: 130, Base: 130) Command: ls  Flags:[ R    ]
| Stopped Queue:  0 Processes
| None
| Defunct Queue:  0 Processes
| None
+------------
| Selecting to Run on the CPU
| PID:   1 Time Remaining:  2 Priority (Cur: 130, Base: 130) Command: ls  Flags:[ R    ]
\=============

.=============
| Starting Time:  2
+------------
```

```
| Returning Process (PID: 1) to Scheduler
| PID:   1 Time Remaining:  1 Priority (Cur: 130, Base: 130) Command: ls  Flags:[ R    ]
+------------
| Sending STOP to Process (PID:  1)
+------------
| Ready Queue:    0 Processes
| None
| Stopped Queue:  1 Processes
| PID:   1 Time Remaining:  1 Priority (Cur: 130, Base: 130) Command: ls  Flags:[  T   ]
| Defunct Queue:  0 Processes
| None
+------------
| Selecting to Run on the CPU
| None
\=============

.=============
| Starting Time:  3
+------------
| Nothing to Return to Scheduler
+------------
| Passing (No Action)
+------------
| Ready Queue:    0 Processes
| None
| Stopped Queue:  1 Processes
| PID:   1 Time Remaining:  1 Priority (Cur: 130, Base: 130) Command: ls  Flags:[  T   ]
| Defunct Queue:  0 Processes
| None
+------------
| Selecting to Run on the CPU
| None
\=============

.=============
| Starting Time:  4
+------------
| Nothing to Return to Scheduler
+------------
| Sending CONT to Process (PID:  1)
+------------
| Ready Queue:    1 Processes
| PID:   1 Time Remaining:  1 Priority (Cur: 130, Base: 130) Command: ls  Flags:[ R    ]
| Stopped Queue:  0 Processes
| None
| Defunct Queue:  0 Processes
| None
+------------
| Selecting to Run on the CPU
| PID:   1 Time Remaining:  1 Priority (Cur: 130, Base: 130) Command: ls  Flags:[ R    ]
\=============

.=============
| Starting Time:  5
+------------
| Returning Process (PID: 1) to Scheduler
| PID:   1 Time Remaining:  0 Priority (Cur: 130, Base: 130) Command: ls  Flags:[ R    ]
+------------
| Exit Selected
+------------
| Ready Queue:    0 Processes
```

```
|None
| Stopped Queue:  0 Processes
|None
| Defunct Queue:  1 Processes
|PID:   1 Time Remaining:  0 Priority (Cur: 130, Base: 130) Command: ls  Flags:[   Z  ]
+------------
| Selecting to Run on the CPU
|None
\=============

We're done!
```