CS333 Autumn 2023
Lab6 clarifications

A new solutions file has been created for lab6 and attached here, with some changes.

Following are the corrections/clarifications in the solutions provided for lab6:

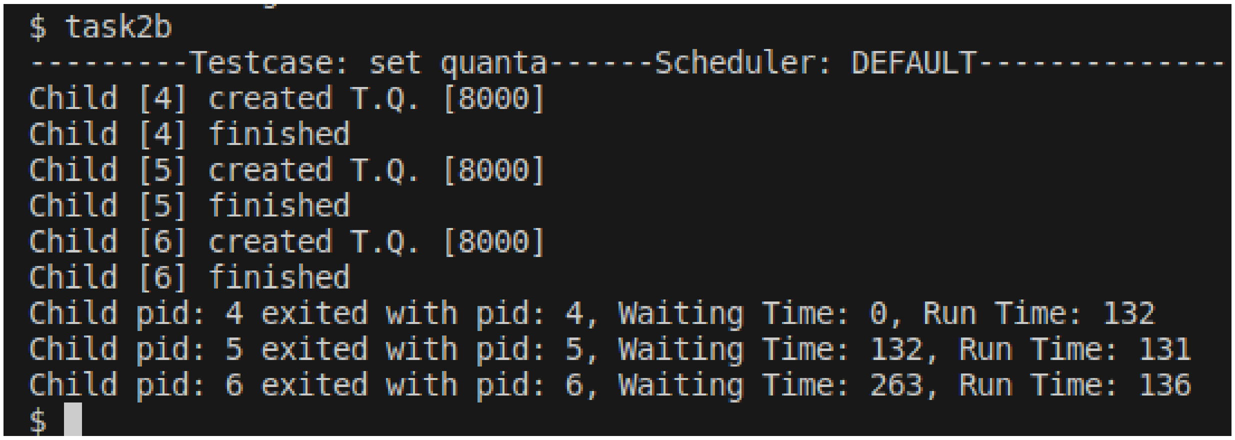
1. For **Task1b.** The `updatestatistics()` call was redundant and was leading to wrong outputs. As we are already updating the runtime whenever `sched()` is called and updating wait time in the scheduler() when the process is getting scheduled again. Hence, separately incrementing the ticks using `updatestatisitics()` is not necessary and its usage has been removed from `trap.c` and `proc.c` in the solution.

2. For **Task2b.** In `trap.c`, at line no. 149, the case was not handled when the current time slice of the process is equal to 1 . Hence, `current>1` has been changed to `current>=1.` Previously, the current slice was never reaching the value `0` and hence, the process was never calling `yield()` and hence running till completion. Hence, in the older output, the first process had a waiting time of `0`.

3. For **Task 2b.** Following are the corrected expected outputs for different scenarios of the time quanta values:

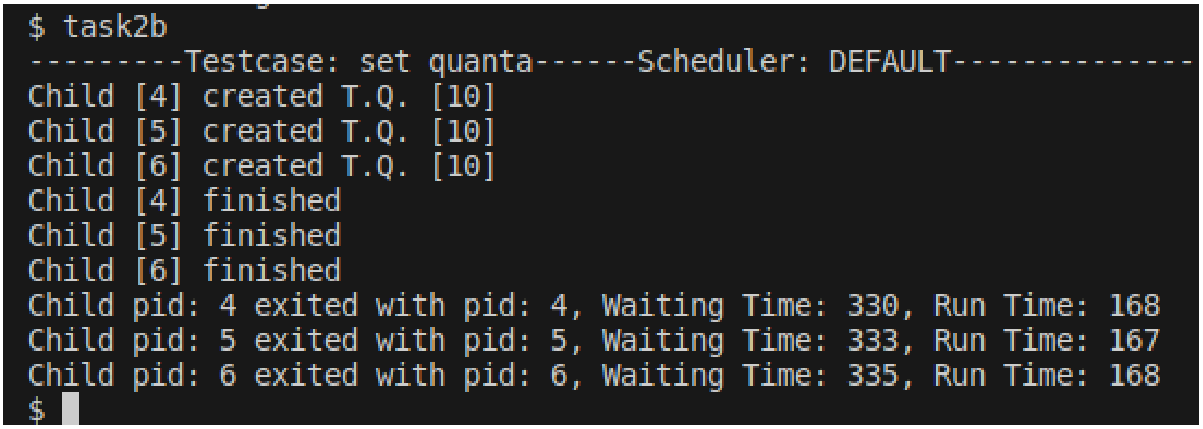
   Note: You can vary the time quanta values for each of the processes by modifying line 32-34 in `task2b.c`

   a. **Scenario 1:** When all processes have a large time quanta value, (time quanta > runtime), the scheduling works as FCFS instead of Round Robin and every process runs to completion. So, **wait time of ith process = wait time + run time of the i-1th process**



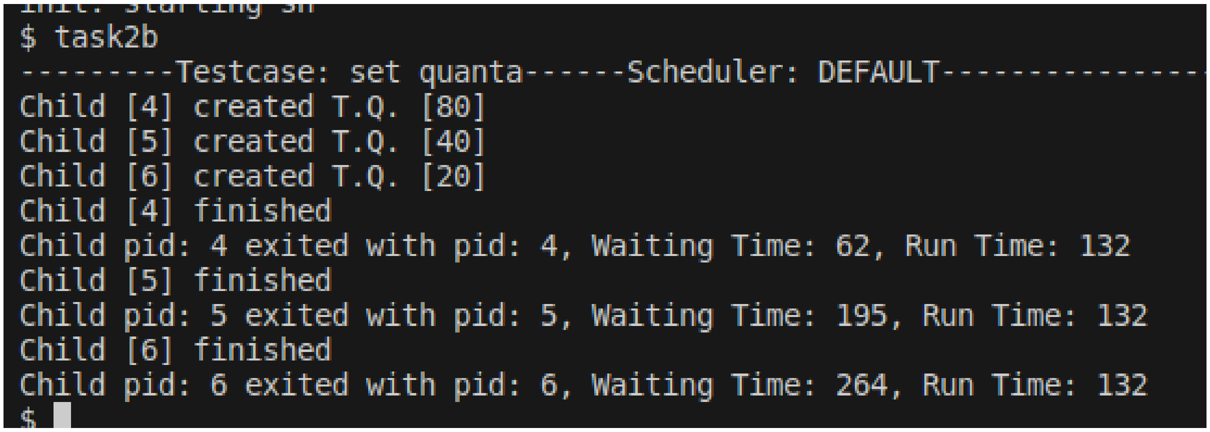
```
$ task2b
---------Testcase: set quanta------Scheduler: DEFAULT-------------
Child [4] created T.Q. [8000]
Child [4] finished
Child [5] created T.Q. [8000]
Child [5] finished
Child [6] created T.Q. [8000]
Child [6] finished
Child pid: 4 exited with pid: 4, Waiting Time: 0, Run Time: 132
Child pid: 5 exited with pid: 5, Waiting Time: 132, Run Time: 131
Child pid: 6 exited with pid: 6, Waiting Time: 263, Run Time: 136
$
```

b. **Scenario 2:** When all the processes have same time quanta value, all the processes get equal opportunity to run in a round-robin manner. Therefore, the wait time of all the processes is nearly identical. The slight difference in the wait times can be explained by the ticks taken by `sched()` when switching between the processes.



```
$ task2b
---------Testcase: set quanta------Scheduler: DEFAULT-------------
Child [4] created T.Q. [10]
Child [5] created T.Q. [10]
Child [6] created T.Q. [10]
Child [4] finished
Child [5] finished
Child [6] finished
Child pid: 4 exited with pid: 4, Waiting Time: 330, Run Time: 168
Child pid: 5 exited with pid: 5, Waiting Time: 333, Run Time: 167
Child pid: 6 exited with pid: 6, Waiting Time: 335, Run Time: 168
$
```

c. **Scenario 3:** When the processes have different time quanta values, the process having a higher time quanta gets the CPU for a longer period of time in one go, in a round-robin manner. Hence, the waiting time of a process is inversely proportional to its time quanta.



```
init: starting sh
$ task2b
---------Testcase: set quanta------Scheduler: DEFAULT----------------
Child [4] created T.Q. [80]
Child [5] created T.Q. [40]
Child [6] created T.Q. [20]
Child [4] finished
Child pid: 4 exited with pid: 4, Waiting Time: 62, Run Time: 132
Child [5] finished
Child pid: 5 exited with pid: 5, Waiting Time: 195, Run Time: 132
Child [6] finished
Child pid: 6 exited with pid: 6, Waiting Time: 264, Run Time: 132
$
```