Operating Systems
Jacobs University Bremen
Dushan Terzikj

Course: CO20-320202
October 25, 2018

Problem Sheet 3

Problem 3.1

Solution:

The answers to these questions are referenced from the following article from Linux Journal.

(a) Fairness in CFS context means that the CPU processing power is shared between processes **equally**. That means that if there is only 1 process running, then the processing power of the CPU given to that process will be 100%. If there are 2 processes, then each gets 50%, if there are 4 processes each gets 25% and so on.

(b) The data structure that CFS uses is an RBTree (red-black tree). CFS uses the fair clock and wait runtime to keep all the runnable tasks sorted by the *rq→fair_clock - p→wait_runtime* key in the RBtree. So, the leftmost task in the tree is the one with the gravest CPU need, and CFS selects the leftmost task.
The reason that RBTrees were chosen for CFS is that as the system progresses forward, newly awakened tasks are put into the tree farther to the right, giving every task a chance to become the leftmost task and, thus, get on the CPU within a deterministic amount of time.

(c) No, it does not use time-slices.
There are two parameters which affect time calculations for CFS:

1. System-wide *fair_clock* variable. This fair clock runs at a fraction of real time, so that it runs the ideal pace for single task when there are N runnable tasks in the system.
2. Each process has a wait time. It is the time each process takes for waiting while the CPU was assigned to the currently running task. The wait time is accumulated in the variable *wait_runtime*.

(d) Each task is assigned a weight based on its static priority. The task with lower weight (or lower-priority) will see time elapse at a faster rate than that of higher-priority task. This leads to its wait_runtime exhausting more quickly than that of higher-priority task, so lower-priority task get less CPU time compared to higher-priority tasks.

Problem 3.2

Solution:
Please check *runners.c*. Run *make* in order to compile. Run *make clean* in order to cleanup.