Operating Systems
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: CO20-320202
Date: 2016-10-11
Deadline: 2016-10-18

Problem Sheet #3

Problem 3.1: *the effect of being nice* (3+3 = 6 points)

Unix processes can lower their priority by setting their “nice” value (the idea is to be nice to others by lowering the priority of compute intensive processes, e.g., a simulation performed in the background).

a) Write a program that creates several child processes. Every child process executes a long (potentially endless) computationally intensive loop with a different `nice()` value n, with $n \in \{0..20\}$.

b) Watch the execution of the program using a utility like `top` using a long update interval. What is the resulting CPU distribution among the processes? Provide a table with the data you have collected and produce a suitable plot. Provide relevant information about the hardware and software configuration you used for the experiment (number of cores, operating system, scheduling algorithm, . . .).

Problem 3.2: *Completely Fair Scheduler and Brain Fuck Scheduler* (1+1+1+1 = 4 points)

With the Linux kernel version 2.6.23, the Completely Fair Scheduler (CFS) was introduced. Read about the CFS scheduler used by mainline Linux kernels (see the CFS documentation in the Linux kernel source tree[1]). An alternative to CFS is the Brain Fuck Scheduler (BFS). Read also about the BFS scheduler (contained in the BFS kernel patch[2]). Answer the following questions:

a) What does fairness mean in the context of the CFS scheduler?

b) Are there other definitions of fairness? Hint: Search for the term ’fair-share scheduling’.

c) How does the CFS scheduler select tasks to run? What is the main data structure used?

d) How does the BFS design differ from the CFS design?

[1] `https://www.kernel.org/doc/Documentation/scheduler/sched-design-CFS.txt`
[2] `http://ck.kolivas.org/patches/bfs/bfs-faq.txt`