Topic: Watching the dynamic priority adjustments of Windows and their effect on the performance of applications.

In Exercise 1, you will first observe the behavior and performance of a process with CPU-bound threads (1a) and that of an I/O bound process (1b) when either is running on the system alone. Then you will observe how the performances are affected when both processes are run together (1c). Finally, you will simulate the situation of a system which is not doing dynamic priority adjustments, and again observe the performances of the two applications.

In Exercise 2, you will run a program demonstrating a special case of dynamic priority adjustment in Windows, using yet another Windows tool to observe the behavior of the program’s threads.

Let me have a look at your observations during the lab sessions to make sure that they are the expected ones before you start finding the explanations for them!

Turn in a hardcopy of your solutions by the due date announced in Moodle and in class. As always, do not only describe your observations and results, but also give sufficient explanations for them, and give proof of your practical work.

Exercise 1

Preface: For this exercise, all involved processes must be restricted to using the same single CPU by adjusting their affinity in Task Manager or Process Explorer. Preferably, run the programs on a physical machine, which otherwise is as idle as possible. For example, a started Firefox heavily influences some of the measurements even when you are not actively using it.

Use Process Explorer to observe running threads (in particular their current priority, CPU usage [user and kernel mode], and context switch delta [only available for the process, not for the individual threads]), and the interrupt activity on the system (number of interrupts, and CPU time used for interrupt handling).

a) Running several CPU-bound threads:

The program CPUSTRES.EXE can run up to 4 worker threads. You can specify the base priority of the threads, as well as an “Activity Level”. At level “Low”, the thread is sleeping 75% of the time, at “Maximum” it executes an infinite loop.

Observe the context switch delta of CPUSTRES.EXE, and the CPU usage and state of the worker threads with varying numbers of worker threads (2-4, the case of one thread is slightly exceptional), all at activity level “Maximum” and with the same base priority. How and why do the values for context switch delta change (or don’t change) with the number of active threads? Do the values tell anything about quantum size?

Which counter is the best indicator for the performance of the program?
b) Running an I/O-intensive thread:

End CPUSTRES.EXE, and run IO_intensive.exe. This program continuously writes to a file that you have to specify when the program starts. Make sure to specify a file on a local magnetic disk (not on a file server, and not an SSD). Observe the behavior of the program (priorities, context switch delta, CPU time in user mode and in kernel mode), also paying attention to whether the thread gets a dynamic priority boost. Also watch the interrupt activity on the system.

Describe and explain your observations, including an explanation for the relationship between the program activity (I/Os), and the context switch delta. Also describe the observed relationship between those two and the increase in the number of interrupts, and explain why the number of interrupts increases because of the disk I/Os. (The quantitative change in the number of interrupts depends on the Windows version, and I have no explanation for it.)

Which counter is the best indicator for the performance of the program?


c) Running a CPU-bound thread and an I/O-intensive thread simultaneously with the normal dynamic priority adjustment of Windows:

Simultaneously run CPUSTRES.EXE with any number of threads at activity maximum and IO_intensive.exe, both with the same base priority (all on the same single CPU, as already mentioned above). Again observe the relevant thread parameters. How and why does running IO_intensive.exe change the observed counters and/or the performance of the CPUSTRES.EXE threads? Do the counters and/or the performance of IO_intensive.exe also change (significantly)?

Remark: Be more specific than just saying “slightly slower”, or “much faster”. Evaluate the performance with the “best indicators” you identified before.

d) Running a CPU-bound thread and an I/O-intensive thread simultaneously without dynamic priority adjustment:

You cannot switch off the dynamic priority adjustments done by Windows. However, by cleverly adjusting the settings of the CPUSTRES threads, you can emulate the behavior as if those adjustments didn’t exist. Often, the behavior changes after some time (tens of seconds, or minutes) to that to be observed in Exercise 2. You need not explain why this happens, but make sure that you describe the behavior before this change occurs!

Describe and explain the thread counters and the performance of both programs in this case. Also compare the results with those from part c), and evaluate the effect of the priority adjustment done by Windows for such a mixed workload of CPU-bound and I/O-intensive threads, which, of course, is very common on real systems.
Exercise 2

The program `Scheduler_Test.exe` creates several threads, two of which increment and output an integer in an infinite loop at different base priorities (6 and 8).

Observe and describe the output of the program. Identify the two counting threads in the process, and then observe the current priority of the lower priority one with the Windows performance monitor. Use `PERFMON4.EXE`, which is the performance monitor from Windows NT 4.0. Contrary to more recent versions, it allows to set the update interval to values smaller than 1 second (use a value of 0.01).

Document, describe, and explain your observations.