Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

System Construction Course 2016,

Assignment 7

Felix Friedrich, ETH Zürich, 1.11.2016

Implementation of Priority Inheritance

Lessons to Learn

- Understand the problem of Priority Inverion and how to deal with it
- Get deeper insight into scheduling and process synchronisation algorithms of $\mathcal{A}_2$

Preceding Remark

- We are considering the A2 System with 3 user priorities (Low, Normal, High).
- The *development system* is Oberon running on windows or Linux, an emulation of A2 under Windows/Linux. Its scheduling is implemented using Windows/Linux threads and thus does not strictly follow the scheduling protocol of A2. Henceforth as *target system* we will use A2 running in a virtual machine. This A2 system does *not yet* comprise priority inversion handling.

Preparation

1. For this exercise you can continue using Bochs as used in the previous exercise. If you feel uncomfortable with this, you can use any virtualisation environment that emulates modern x86 hardware and that has a support for booting from raw disk images as IDE devices.
2. Open a console in directory assignments/assignment7
3. Use the script in file MakeA2.cmds in order to to compile and link the boot-file and to inject the files into a bootable HDD image by calling `oberon run MakeA2.txt` [1]

Task 1

Analyse the behavior of the test command `TestPrioInv.Start` by looking at the source code of module TestPrioInv.Mod and the log output of the started system. What implication does the scheduling policy have?

Task 2

Modify procedure `Lock` of module Objects.Mod in such a way that it produces some console output if priorities are supposed to be inherited. Make sure that your log output only appears when you need it, i.e. use TestPrioInv.Mod somehow to trigger the output. Otherwise you get ample logging already during booting of the kernel.

[1] or, equivalently, by executing `SystemTools.DoFile MakeA2.txt` within the Oberon shell.

Task 3

Implement the priority inversion handling protocol that was presented in the lecture.

- First make sure that priorities are set and reset correctly in procedures `Lock` and `Unlock`. In a second step the priority propagation should be implemented as a helper procedure `PropagatePrio`. Add respective calls in `Lock` and `Unlock`.
  Hint: No additional fields in the data structure are required. The counter fields are already present at the data structure of processes `Objects.Process.prioRequests` and object headers `Heaps.ProtRecBlockDesc.waitingPriorities`
- Update procedures `Await` and `TransferLock` accordingly.

Documents

- System Construction Lecture 7 slides from the course-homepage
  `http://lec.inf.ethz.ch/syscon`
- $\mathcal{A}_2$ Programming Quickstart Guide. File A2QuickStartGuide.pdf in folder documents/oberon