University of Karlsruhe
System Architecture Group
Gerd Liefländer

Teaching Assistant:
Philipp Kupferschmied

System Architecture 2008/09
Assignment 10

Question 10.1: Linux Scheduling Domains

1. Why might a scheduler that treats all CPUs of a multiprocessor-system equally not be well-suited for today's computers?
2. What is the purpose of *scheduling domains* [1]?
3. How are the `sched_domain` and `sched_group` structure related?
4. Try to visualize the hierarchy of `sched_domains` and the `sched_groups` within each domain for a NUMA system consisting of two nodes with two physical CPUs each, and two virtual CPUs per physical CPU.

Question 10.2: Priority Inheritance

Assume that in your single-processor RTOS you have $n > 1$ concurrent processes and $m > 1$ mutual exclusive resources. The only interaction of these n processes is competing for these m resources, which are controlled by special binary semaphores.

1. Extend the PCBs and the control structure of each resource (including these binary semaphore objects) according to the requirements of the basic priority inheritance model (see the original paper on priority inheritance [2], p. 1177).
2. What are the two major problems of this basic inheritance protocol?
3. What is the easiest way to *eliminate* one of these two inherent problems?

Question 10.3: Prerequisites for Deadlocks

Recall the necessary conditions for deadlocks. For each condition, give an example of how deadlocks can be prevented by breaking the condition.

Question 10.4: Spooling System

Figure 1 shows a simple spooling system which consists of an input thread t_i, a processing thread t_p, and an output thread t_o. The threads are connected via two buffers through which the threads exchange data blocks of equal size. These blocks are buffered in input and output buffers on a disk. The sizes of the input and output buffers depends of the threads' respective production and consumption rates. The communication primitives used ensure that the following resource constraints are satisfied:

$$i + o \leq \max$$

where "max" denotes the maximum number of blocks on the disk, "i" the number of blocks used for the input buffer, and "o" the number of disk blocks assigned to the output buffer.
The threads operate as follows:

- As long as the environment supplies data, thread t_i will eventually put it into the input buffer on disk (provided disk space is/becomes available).

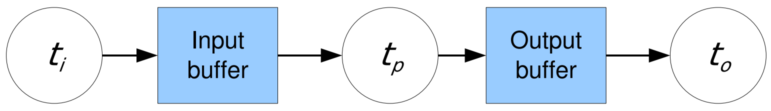

Abbildung 1: Spooling system with three threads connected by two buffers on the same hard disk.

- If data is available in the input buffer, thread t_p will eventually consume it and output a finite number of blocks into the output buffer instead (provided disk space is/becomes available).
- If data is available in the output buffer, thread t_o will eventually consume it.

Discuss whether this system can be deadlocked or not.

Question 10.5: Deadlock Prevention

1. Resource ordering is one method to prevent deadlocks. Explain how it works and give concrete examples of how it can be used advantageously.
2. Is linear ordering an appropriate strategy to prevent deadlocks on resources with multiple indistinguishable units such as buffers? Explain your answer!

Question 10.6: Resource Trajectory Graphs

1. All trajectories on the slides for the lecture on deadlocks are horizontal or vertical. Can you envision any circumstances in which diagonal trajectories are also possible?
2. Can a system be in a state that is neither (unavoidably) deadlocked nor safe? If so, give an example. If not, prove that all states are either (unavoidably) deadlocked or safe.
3. Resource trajectory graphs *could* be used to avoid deadlocks. By clever scheduling, the OS could avoid *unsafe regions*. What has to be assumed in order to avoid deadlocks with this method?

Question 10.7: Deadlock Avoidance (Banker's Algorithm)

Given a system of $n > 1$ threads competing for $m > 1$ types of multi-unit resources, where the total amount available of each type is given by $R = (r_1, r_2, \ldots, r_m)$. Suppose you had to implement a deadlock-avoidance policy.

1. What information do you need if there is one resource manager for all m types of resources?
2. Suppose a current resource allocation for x units of resource type i could be fulfilled (there are enough units available). Should such a request be granted in any case?
3. Assume $n = 5$, $m = 4$, $R = (3, 12, 19, 12)$ and $V = (1, 5, 2, 0)$, where $V[i]$ gives the number of currently available resource units of type i. Matrix A describes the currently allocated resource units and matrix C holds the maximum resource requirements for each resource type per thread. Both matrices provide a row per thread, the columns representing resource types.

$$A = \begin{pmatrix} 0 & 0 & 2 & 2 \\ 1 & 0 & 0 & 0 \\ 1 & 3 & 5 & 4 \\ 0 & 4 & 4 & 2 \\ 0 & 0 & 6 & 4 \end{pmatrix}, \quad C = \begin{pmatrix} 0 & 0 & 2 & 2 \\ 1 & 7 & 5 & 0 \\ 2 & 3 & 5 & 6 \\ 0 & 6 & 5 & 2 \\ 0 & 6 & 7 & 6 \end{pmatrix}$$

Analyze whether the system is in a safe, unsafe, or (unavoidably) deadlocked state.

References

[1] M. J. Bligh, M. Dobson, D. Hart, and G. Huizenga. Linux on NUMA systems. In *Proceedings of the Linux Symposium*, volume 1, pages 89–102.

[2] Lui Sha, Ragunathan Rajkumar, and John P. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Transactions on Computers*, 39(9):1175–1185, 1990.