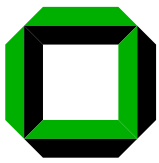

University of Karlsruhe
System Architecture Group
Gerd Liefländer

Teaching Assistant:
Philipp Kupferschmied

System Architecture 2008/09
Assignment 14

Question 14.1: Device Drivers

1. Many operating systems have interrupts enabled within the kernel (apart from some critical code paths). Assume the kernel is executing some function f when an interrupt occurs. Can the interrupt handler safely call f in any case?

2. Most modern operating systems split the handling of interrupts in two phases, a high-priority phase that is handled immediately, and a low(er) priority phase whose execution might be deferred to a later point in time (e.g., Linux' top-halves and bottom-halves or Windows' Deferred Procedure Calls). What is the motivation for such a design?

Question 14.2: Synchronous vs. Asynchronous I/O

Syscalls such as `write` that trigger device I/O can be either *blocking* (synchronous I/O) or *non-blocking* (asynchronous I/O).

1. Relate such syscalls to IPC syscalls.

2. The lecture slides provide pseudo-code for blocking/synchronous I/O operations. How would you implement non-blocking I/O in each of the three models (programmed I/O, interrupt-driven I/O, and DMA)?

3. If an application requests asynchronous I/O, may the system implement synchronous I/O instead without potentially causing the application to fail?

4. May the system implement asynchronous I/O although the user requested synchronous I/O?

Question 14.3: Byte-Level Access to Block-Devices

As stated in lecture, disks cannot access individual bytes but only blocks. If you write byte-granular data to a file, is it reasonable to assume that the data is physically on the disk after you have executed the `write` system call?

Question 14.4: Disk Scheduling

For the disk scheduling algorithms FCFS (FIFO), Scan, and SSTF, compute average track head movements given the following sequence of disk track requests: 129, 37, 31, 99, 89, 102, 15, 63, 130. Assume an initial head position over track 100, moving towards track 0.

Question 14.5: Anticipatory Disk Scheduling

Have a look at (some of) the information given at `http://www.cs.rice.edu/~ssiyer/r/antsched/`

1. Describe the problem of "classical" (i.e., non-anticipatory) disk scheduling algorithms.

2. How does anticipatory disk scheduling solve this problem?

Question 14.6: RAID

Compare a SLED and RAID systems of levels 0 to 5. Assume that each RAID uses four disks for actual data storage and that the RAID2 uses three additional bits for the error correcting code. For each setup, answer the following questions:

1. How many disks do you need?
2. You want to modify one byte of your data. How many blocks do you have to read/write?
3. One of the data disks fails. What has to be done to recover the data?