University of Karlsruhe
System Architecture Group
Gerd Liefländer

Teaching Assistant:
Philipp Kupferschmied

System Architecture 2008/09
Assignment 2 Update 1

For some questions of this assignment, the "Hardware (P)Review" and "History of Operating Systems" slides that can be found on the course website might be helpful.

Question 2.1: Potpourri

Mark each statement as *correct* if it is *always* true, otherwise mark it as *incorrect*:

1. A system call is an invocation that crosses protection domains.
2. Unix was developed in response to the complexity of Multics.
3. OS/360 was the first PC OS.
4. Address translation is necessary but not sufficient to support virtual memory.
5. The Intel x86 architecture supports three-level page tables in hardware.

Question 2.2: Insights into Hardware Design

1. What are some of the differences between a processor running in *privileged mode* (also called kernel mode) and *user mode*? Why are the two modes needed?
2. Describe the principle and the benefits of a memory hierarchy. How can memory hierarchies provide both fast access and large capacity? What typical program behavior coincides with the benefits of a memory hierarchy?
3. Cache memory is divided into (and loaded in) blocks (also called cache lines). Why is a cache divided into these cache lines? What might limit the size of a cache line?
4. You have the choice of buying two different computers A and B. A has a 3 GHz processor and a 512 kB cache, whereas B has a 2 GHz processor with a 3 MB cache. Both machines have the same kind of RAM (same bandwidth, same latency). Which of the two computers would you prefer?
5. Given that disks could stream quite fast (e.g., 1 block in tens of microseconds), why are *average access times* for a disk block still in the range of milliseconds?

Question 2.3: Interrupt Notions

1. Define the terms "interrupt", "exception", and "trap" and discriminate them from each other.
2. What is an interrupt vector?
   - array that contains addresses of all interrupt handlers?
   - address of an interrupt handler?
   - linear diversity in the gap?
3. What is an interrupt service routine (ISR)?

- The update function of an OS?
- System code that has to be called in case of an interrupt?
- ISRs are only used by software interrupts?
- Quick online support?

Question 2.4: History

1. Why was time sharing not widespread on early so-called “batched” systems?
2. On Unix, which of the following are considered system calls? Why?
   (a) `read()`
   (b) `lseek()`
   (c) `sprintf()`
   (d) `memcpy()`
   (e) `open()`
   (f) `strncpy()`
3. In what respects do multi-user systems differ from single-user systems? What additional requirements must a multi-user operating system fulfill?

Question 2.5: System Call Basics

1. What events can lead to an invocation of the kernel?
2. When a system call is executed, parameters might need to be passed to the kernel. How can this be achieved?
3. What problem exists when a system call expects a pointer to a user-level buffer to which the kernel has to write data? Is there also a problem when the user-level buffer is only read?

Question 2.6: Design Goals

1. Some system designers and vendors claim compatibility to be the most important design goal of an operating system. Do you agree? Discuss the pros and cons.
2. Suppose you have to design (and implement) the on-board computer for your next trip to Mars. What system goals would you try to achieve with your design?
3. Assume you are running a small server with one CPU with an operating system that is said to be *scalable*. What system behaviour would you expect when
   - the number of programs running on the server doubles?
   - you exchange the CPU with a modern quad-core CPU, while system load remains the same?

   How would a non-scalable system behave in these cases?