NTNU | Norwegian University of Science and Technology

Operating Systems

Lecture 23: Semester review

Michael Engel

First things first – what's relevant?

- **The most urgent question: which parts of the material in the course is relevant for the exam?**

- **Everything except:**
  - Lecture 8 (From Source Code to Process)
  - Lecture 13 (Real-time Scheduling)
    - but note that lecture 12 on scheduling is ***very*** relevant!
  - Lectures 19 and 20 (embedded systems part 1 and 2)
  - Lecture 22 (Security 2)
- **Especially these topics are relevant:**
  - Topics of the practical exercises (!)
  - The material from the lectures as rehearsed in the theoretical exercises

What about…?

- The C crash course?
  - The exam will certainly require knowledge of C on an advanced level, but we won't ask specific questions about details of C programming
  - Nevertheless, you must be able to read and understand C programs and also to find and fix errors
- Unix?
  - We demonstrated OS concepts using Unix system calls and libc functionality. Knowledge of their use and functionality is expected and was fundamental for the practical exercises!
  - You will not be asked details of a specific Unix/Linux implementation in the exam
    - e.g. how things are realized in MacOS X or Linux – ***exception:*** different file system implementations

The rest of this lecture

This is a quick walk-through summary of the ***most relevant*** topics we discussed in the run of this course

It gives an overview of the **minimum** extent of (sub)areas I expect you to have knowledge about

…but it is certainly not exhaustive

The takeaways from each lecture are given as a number of ***questions*** – you can figure out if you can answer these and, especially if not, read up on these topics

As mentioned before, some lectures are not relevant for the exam and are thus not summarized here

1. Introduction to operating systems

Introduction to operating systems and their history

Important questions:

- Why were operating systems developed initially?
- How did the features of operating systems evolve along with the development of the available hardware?
- How do we define the term "operating system" today?

You do *not* need to know details about the historical computers and operating systems discussed in that lecture

2. Resources and computer architecture

Interaction of computer architecture and the OS, resources and their management

- What are the building blocks of a computer system?
- Which resources are represented by these building blocks?
- How does code (in the OS) interact with hardware resources?
- What are the most relevant developments in computer architecture of the last decades and which problems/ benefits are related to these developments?

3. Challenges and tasks of operating systems

Discussion about abstractions provided by an OS and the related tasks an OS has to perform as well as the resulting problems and challenges

- Which abstractions does a modern OS provide?
  - CPUs, processes, memory, file systems, security, …
- What is a process?
  - How do processes interact with each other and the OS?
    - Synchronization and deadlock fundamentals
  - What different view of processes exists and why?
  - Which states can a process have and what characterizes the different states?

4. Processes

Details about the process abstraction and its use in Unix

- What is the definition of a process and what is the difference to a *program?*
- What is a process hierarchy and why does it exist?
  - Parent/child processes, orphans, zombies and PID 1 (init)
- How can processes perform I/O, how can it be (re)configured?
  - Relation of the I/O concept to the Unix philosophy?
- How do processes interact with the OS: system calls
- How can processes be created/controlled/terminated?
  - Which Unix syscalls are used for process management?
  - Pros and cons of the Unix fork/exec model
- Optimizations for process creation in Unix: copy-on-write
- What are details of the extended process state model?

5. Threads

Threads as a lightweight abstraction and process alternative

- What is the overhead of Unix processes and their creation?
- What are the differences between address spaces for processes and threads?
- What are the thread models in Unix and Windows?
- What are fibers (user-level threads)?
    - Can you discuss pros and cons of threads vs. fibers?
- Cooperative multithreading
    - Can you describe the ideas behind Duff's device and protothreads? (you don't have to know the details of their implementations)

6. Concurrency: Mutual Exclusion and Synchronization

Interaction of parallel activities, the resulting problems and solutions

- What is shared data/memory communication, why is it problematic?
  - Can you give an example of a problematic situation?
  - Can you understand multithreaded code using shared data?
- What is a race condition (can you give examples)?
  - Why are race conditions hard to detect and debug?
- What is synchronization used for, which options for synchronization exist?
  - Can you define the term "*critical section*"?
- What are locks and how are they used?
  - Can you give details on lock implementations (atomic operations, suppressing interrupts, semaphores)?
- What is a semaphore, which operations exists on semaphores?
  - Can you define the use and *implementation* of semaphores?
  - Can you describe problems (e.g. reader/writer) solved using semaphores?
- What are monitors and how to they differ from semaphore solutions?

7. Concurrency: Deadlocks and Starvation

Problems using synchronization and possible solutions

- Can you define the terms "deadlock" and "livelock"?
  - Explain situations leading to both problems
- What are the *necessary conditions* for deadlocks to occur?
  - What is the *additional condition* that is required for a deadlock to occur?
- Which types of resources exists related to synchronization?
- What are the components of a resource allocation graph, how do you construct it?
  - How can you detect a deadlock in this graph?
- What is the dining philosophers problem?
  - Why do deadlocks occur here?
  - Can you describe a solution to solve the problem?
  - Can you discuss the efficiency of different solutions?
- How can deadlocks be prevented and what are safe/unsafe states?
- Which methods exist to resolve a deadlock and what are their pros/cons?

8. From source code to process

Details of processes and binaries in Unix, VM basics, process start

- What are the sections of an ELF executable, what is stored in each section?
- What are the tasks of compiler, assembler, linker, loader?
- What is symbol resolution, which tool performs this task?
- Why is virtual memory useful?
  - How does the translation from virtual to physical addresses work?
  - What is a typical virtual memory layout of a Unix process and how is this related to ELF executable sections?
- What is a shared library and why is it useful?
  - Can you give differences between linker and (dynamic) loader?
- How are Unix processes created?
  - Can you describe the effects on physical and virtual memory spaces?
  - How do fork and exec interact?
- You *do not* need to know exact details of process creation (init, libc start etc.)

9. Memory management

Main memory as a resource and its management

- Requirements for memory management for multiprogramming systems?
- Which policies and strategies are relevant for memory management?
- Can you describe the basic problem of memory allocation?
- How does dynamic memory allocation work?
  - Can you describe different approaches, describe pros/cons?
- Can you name and describe different placement strategies?
- What is memory fragmentation?
  - Which kinds of fragmentation exist, what are their properties?
  - Where are different allocation methods typically used?
- Can you describe differences between swapping, segmentation, paging?
  - How does paging as an OS concept interact with the MMU?
  - How can paging be optimized using hardware or software approaches?

10. Virtual memory

Details on concepts and implementation of virtual memory

- What is the locality principle in computers?
  - Which kinds of locality exist, can you describe their properties?
  - How can locality be used to optimize performance?
- What is the idea behind virtual memory, which abstraction/illusion is created by virtual memory?
- How does demand paging work?
  - What is a page fault and how is it handled?
  - What are tasks of the OS and hardware when handling page faults?
- Can you name different page replacement strategies and discuss their pros/cons? (FIFO, optimal, LRU, second chance)
  - Can you simulate different strategies given an access sequence?
- Can you define *thrashing* and name causes and possible solutions?
- What is the working set of a process and how can you determine it?

11. Inter-process communication

Communication between processes

- Which approaches to inter-process communication exist?
  - Can you give their pros/cons?
- What are the primitives for message-based communication?
  - Which synchronization methods exist here?
  - How can processes be addressed?
  - Which message formats exist?
- Which IPC methods exist in Unix?
- Can you describe the concepts and use (programming) of…
  - Signals, unnamed pipes, named pipes, Unix message queues, sockets
- What is RPC and what is the fundamental difference to IPC?

12. Uniprocessor scheduling

Processors and processor time as a resource and its management

- Can you define the terms "*dispatching" and* "*scheduling*"?
- Which dispatch states exist, which level of scheduling are they related to?
    - Can you describe details of short/medium/long term scheduling?
    - Which process state transitions are related to which scheduling level?
- Can you explain *preemptive scheduling* and its advantages?
- Can you give examples for scheduling strategies, explain how they work?
    - Can you determine scheduling orders for a given strategy?
    - Can you discuss pros/cons of the different scheduling strategies?
- What is multi-level scheduling and how is this related to priorities?
- Can you give details of scheduling strategies in Unix and Windows?

14. I/O management and disk scheduling

I/O devices as resources and their management

- How do devices and the OS interact? Can you name different methods?
- Which classes of devices exist and what are their properties?
- How do interrupts and DMA work and what are their pros/cons?
- How can I/O devices be addressed by the OS?
- What is a device driver, in which ways can it interact with the hardware?
- What are the various tasks of the OS related to devices?
- How are devices represented and abstracted in Unix?
  - Name properties of/differences between character/block/other devices
  - How does the OS implement the relation device special file ⇔ device driver?
- How can devices be used in user processes, what are related syscalls/libc functions?
  - Why is buffering important, can you discuss the pros/cons?
  - How does a *ring buffer* work, where is it typically used?
- How does I/O scheduling for disk drives work
  - What are the pros/cons of the different scheduling approaches?

15. File systems 1

Files as an abstraction of disk space and their management

- What is the file abstraction and why is it useful?
- What are the syscalls/libc functions in Unix to handle files?
- What is a *virtual file system* and how does this work?
  - What is mounting/unmounting, what is their effect on the directory tree of a Unix system?
- Which methods exist to map a file to disk blocks?
  - Describe problems of the approaches/pros/cons
- Which methods exist to manage free space?
- What are the directory and inode structures for typical file systems?
  - Unix System V, BSD FFS, Linux ext2/3/4

16. File systems 2

Modern file systems (FS)

- What are the challenges for file systems today?
- How can the reliability of disk storage be improved?
- How can the performance of disk storage be improved?
- What is the Unix block buffer cache and how does it work?
- What is logical volume management and why is it useful?
- What is RAID?
  - Which different RAID levels exist and how do they work?
  - Can you discuss the pros/cons of the different levels?
- What is a journaling/log structured file system?
  - How do they work, what are differences to traditional FS?

17. Virtual machines and microkernels

Operating system architectures and abstractions

- Can you define monolithic kernels, microkernels, hypervisors?
  - Differences between these, pros/cons?
- What problem did first-generation *microkernels* have?
  - How was this solved in second-generation microkernels?
  - What is an *exokernel*?
- What is virtualization, can you define its functionality?
  - What is a *virtual machine monitor* or *hypervisor*?
  - What are the differences between *type 1 and 2 hypervisors*?
  - Which hardware support was introduced to support virtualization?
  - What is *paravirtualization* and what are its pros/cons?
  - What is a *hypercall*?

18. Cloud, Unikernels, single-address space OS

From virtualization to the Cloud and its OS approaches

- Which service models exist for Cloud systems?
  - What are their properties, pros and cons?
  - Which provisioning models exist?
- What does the architecture for a Cloud OS look like?
  - What are differences to a "regular" OS?
  - Which strategic decisions have to be taken by a Cloud OS?
- What is a container and how are containers related to virtualization?
  - What is virtualized in containers?
- How does virtual memory management interact with virtualization for the Cloud?
  - Which optimization approaches exist, can you describe them?
- How can I/O be virtualized for the Cloud?
  - Which I/O virtualization approaches exist, can you name pros/cons?

21. OS Security 1

Security and its management in the OS

- Can you define *safety* and *security*?
- What is the task of OS security?
- Can you give examples for *malware*?
  - What is the difference to *social engineering*?
- Which types of malware exist, can you define them?
- What is permission management and what are the related requirements?
  - What is the *principle of least privilege*?
- Define the *access matrix* and describe the ways to use it
  - File/process attributes in Unix
  - ACLs, capabilities, mandatory access control
- How do the MMU and the CPU *privilege levels* contribute to security?
- What is *software-based protection*, can you give an example?
- Which typical *software bugs* contribute to security problems, can you give examples?

That’s all…

If you are *still* interested in operating system topics 😉, get in touch for a student exchange in Bamberg, Germany!

I have started my new position at Bamberg University last Friday…

Bamberg: Old city hall – CC BY-SA 4.0 by Ermell