OS Overview *

Jens Lechtenbörger

Computer Structures and Operating Systems 2023

Contents

1 Introduction

1.1 Recall: Big Picture of CSOS

- Computer Structures and Operating Systems (CSOS)
  - CS: How to build a computer from logic gates?

Figure 1: "NAND" under CC0 1.0; from GitLab

    * Von Neumann architecture
    * CPU (ALU), RAM, I/O

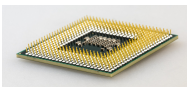

Figure 2: "CPU" under CC0 1.0; cropped and converted from Pixabay

  - OS: What **abstractions** do Operating Systems provide for applications?
    * Processes and threads with scheduling and concurrency, virtual memory

*This PDF document is an inferior version of an OER HTML page; free/libre Org mode source repository.

Figure 3: "Pong in TECS VM" under GPLv2; screenshot of VM of TECS software suite

* What is currently executing why and where, using what resources how?

1.1.1 OS Responsibilities

Warning! External figure **not** included: "What does your OS even do?" © 2016 Julia Evans, all rights reserved from julia's drawings. Displayed here with personal permission.
(See HTML presentation instead.)

1.1.2 Definition of Operating System

- Definition from [Hai19]: **Software**
  - that **uses hardware** resources of a computer system
  - to provide support for the **execution of other software**.

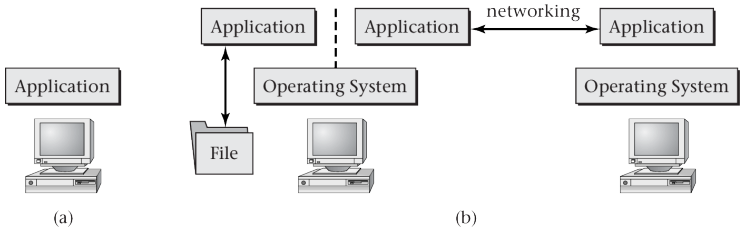

Figure 4: "Figure 1.1 of [Hai17]" by Max Hailperin under CC BY-SA 3.0; converted from GitHub

1.2 Prerequisite Knowledge

- Be able to write, compile, and execute small *Java* programs
  - What is an object? What is the meaning of `this` in Java?
  - How do you execute a program that requires a command line argument?
- Be able to explain basic *data structures* (stacks, queues, trees) and *algorithms* (in particular, hashing)
- Being able to explain the database *transaction concept* and *update anomalies*

2 OS Teaching

2.1 OS Part

- Sharp cut ahead
  - So far, we followed a book with projects to build a computer bottom up
  - Unfortunately, nothing similar exists to build an Operating System (OS) bottom up
    - (Or is far too technical for our purposes)
- Thus, the OS part presents major concepts and techniques of modern OSs
  - Some topics build upon each other
  - Some are more isolated
  - Goal: Understand and control your devices
    - Performance
    - Trust

2.2 OS Exercises

- While the CS part had weekly projects, this will **not** be the case for the OS part
- Exercise sheets continue, though

2.3 Textbook(s)

- Major source (adopted in 2017)
  - [Hai19] Max Hailperin: Operating Systems and Middleware: Supporting Controlled Interaction
    - Distributed under the free/libre and open license CC BY-SA 3.0
      - Source code at GitHub
    - PDF (also) in Learnweb
- Additions
  - [Sta14] Stallings: Operating Systems – Internals and Design Principles, 8/e, 2014
  - [TB15] Tanenbaum, Bos: Modern operating systems, 4th edition, 2015

2.4 Presentations (1/2)

- HTML presentations with audio for self-study
  - Read usage hints first
- HTML generated from simple text files in Org mode

  - Source code available on GitLab
    * You may want to check out those for your annotations
  - Generated presentations; HTML and two PDF variants
    * Those are draft versions until you see a link in Learnweb
  - Offline use possible
    * Generate yourself or download from latest GitLab pipeline
  - Open Educational Resources (OER), your contributions are very welcome!

2.5 Presentations (2/2)

- As usual, (pointers to) presentations are available in Learnweb
- Presentations include
  - Learning objectives
  - Explanations for class topics
  - Some slides contain audio explanations, e.g., the one starting with this bullet point

Starting with the final bullet point of this slide, audio controls should have appeared on the lower left.

The folder icon on the upper right indicates that a transcript of the audio is available.

You read the usage hints for this type of presentations as instructed on the previous slide, didn't you?

3 OS Plan

3.1 Big Picture of OS Sessions

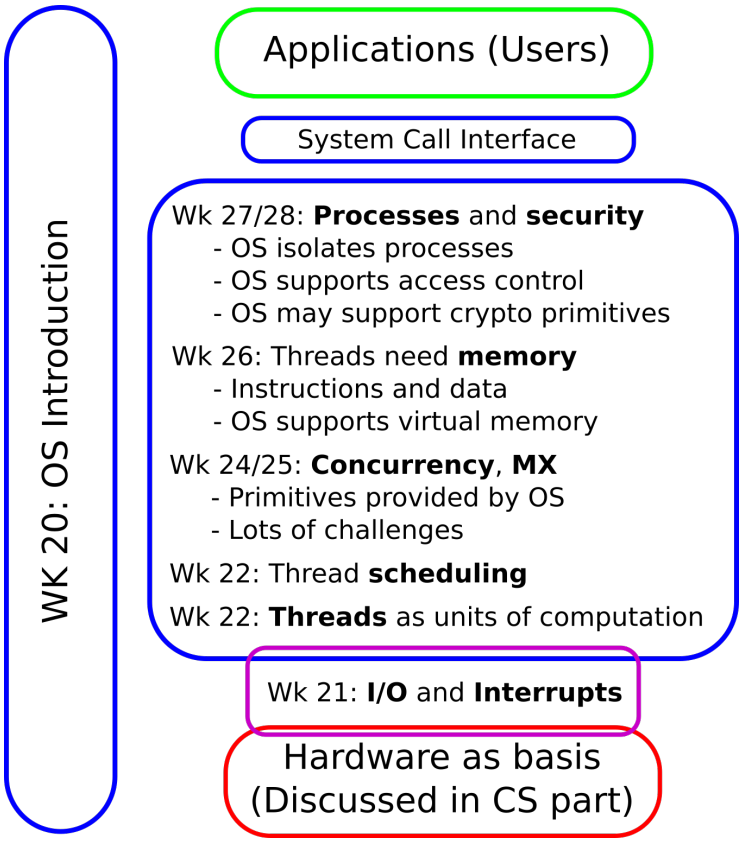

Figure 5: OS course plan, summer 2022

(Audio for this slide is split into several audio files, one for each step of the animation. In contrast, these notes contain a transcript of all animation steps.)

Although the Hack computer does not have an OS, it will help to recall briefly how you interact with that machine. On Hack, you are able to run a single program, where you access hardware directly. E.g., reading from the keyboard requires access to a special memory location that represents the state of the underlying hardware device.

With OSs, applications no longer have direct access to hardware. Instead OSs manage applications and their use of hardware. You will learn that the core part of OSs is called kernel, and each vendor's kernel comes with a specific interface to provide functionality to applications, usually in the form of so-called system calls. E.g., when you use `System.in` in Java to read keyboard input, the Java runtime executes a system call to ask the OS for keyboard input.

Starting from system calls, we will look into techniques for input/output processing (I/O for short) such as access to the keyboard. Recall that in Hack you programmed a loop to wait for keyboard input. Clearly, such a loop keeps the CPU busy even if no key is pressed, wasting the resource CPU if other application could perform useful computations. Hence, I/O is usually paired with interrupt processing, which does not exist in Hack. Briefly, interrupts indicate the occurrence of external events, and OSs come with interrupt handlers. Then, for example, keyboard processing code only needs to be executed when a key was pressed.

In contrast to Hack, OSs manage the execution of several applications, and each application might contain several so-called threads of execution. It is up to application programmers to decide how many threads to use for a single application (and you will create threads in Java).

The OS manages all those threads and their scheduling for execution on the CPU (or their parallel execution on multiple CPU cores). Usually, scheduling mechanisms involve time slicing, which means that each thread runs for a brief amount of time before the OS schedules a different thread for execution. Such scheduling happens in intervals of about 10-50ms, creating the illusion of parallelism even if just a single CPU core exists. Such time-sliced or parallel executions are also called concurrent executions.

If shared resources are accessed in concurrent executions, subtle bugs may arise, a special case of which are update anomalies in database systems. The notion of mutual exclusion (MX) generalizes several synchronization mechanisms to overcome concurrency challenges, and we will look at typical related OS mechanisms and their use in Java.

Just as in Hack, instructions and code of applications need to be stored in memory. Differently from Hack with its Harvard architecture, code and instructions are stored uniformly in RAM in our Von Neumann machines, and the OS manages the allocation of RAM. Importantly, mainstream OSs provide support for virtual memory, which does not exist in Hack, but for which we will see advantages such as isolation and flexibility.

Furthermore, mainstream OSs offer a process concept as abstraction for applications, under which several related and cooperating threads share resources (such as virtual memory or files). Finally, OSs offer various security mechanisms for processes and applications, a selection of which will be topics for the final presentation.

To sum up, this figure visualizes

- what OS topics will be discussed when and
- how topics build upon each other.

Note that some parts of this figure are hyperlinked to other presentations, which the mouse pointer should indicate.

3.2 A Quiz

Bibliography

[Hai17] Max Hailperin. *Operating Systems and Middleware – Supporting Controlled Interaction*. revised edition 1.3, 2017. URL: https://gustavus.edu/mcs/max/os-book/.

[Hai19] Max Hailperin. *Operating Systems and Middleware – Supporting Controlled Interaction*. revised edition 1.3.1, 2019. URL: https://gustavus.edu/mcs/max/os-book/.

[Sta14] William Stallings. *Operating Systems – Internals and Design Principles*. 8th ed. Pearson, 2014.

[TB15] Andrew S. Tanenbaum and Herbert Bos. *Modern Operating Systems*. 4th ed. Pearson, 2015.

License Information

In particular, trademark rights are *not* licensed under this license. Thus, rights concerning third party logos (e.g., on the title slide) and other (trade-) marks (e.g., "Creative Commons" itself) remain with their respective holders.