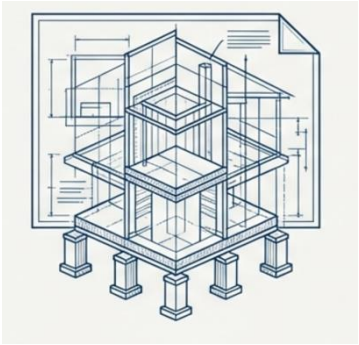
IN2140:
Introduction to Operating Systems and Data Communication

An overview of IN2140

Özgü Alay
ozgua@ifi.uio.no
Room: 4159

Tuesday, January 20, 2026

IN2140 Course Content



- an introduction to **operating systems** (OS), viewed from both the user's and machine's point of view
  - have basic knowledge of the most important components of an operating system
  - be able to develop small applications that call OS services
  - know algorithms, don't program the operating system kernel (that is IN3000)



- insight into how **data communication** is taking place
  - be able to choose between algorithms that enable today's data communication in the Internet
  - be able to develop programs with processes that communicate – within the computer and over a network



- understanding theory through practice by **programming**
  - experience implementing algorithms for operating systems and data communication without the protective cover provided by runtimes and interpreters
  - experience programming in C

Learning Outcome: OS





- **See beyond the code:** How operating systems hide hardware complexity for you? What are kernel space and user space, and how they interact?



- **Learn different algorithms:** What are the typical algorithms for the most important operations performed by the operating system for you? Which algorithm to choose for different workloads?



- **Build smarter systems:** When does software waste performance unnecessarily? What is the right choice of hardware, services and algorithms when you build complex systems?



- **Enable Future knowledge:** Prerequisite for advanced courses in operating systems, compilers, distributed systems, parallel programming, HPC, ...

Learning Outcome: Datacom





- **Map the internet:** How are data and information moved over a network? How are networks structured? What parts of the network you can change yourself?



- **Build distributed applications:** How to write simple networked and distributed applications? What are the right choices when programming networked and distributed applications?



- **Performance matters**: What is the cost of distributing applications? What impacts the performance of complex distributed applications?



- **Enable Future knowledge:** Prerequisite for advanced courses in networks, distributed systems, parallel programming, HPC, security, ...

Learning Outcome: C





- **Direct memory control:** How memory and memory addresses work in practice?
- **Better system understanding:** How operating systems are related to computer architecture? Do networks know about data structures and program structures?
- **Closer to the architecture and OS:**
  - C allows you to easily **interpret every** (accessible and readable) **byte** in the computer **as you like**
  - Most programming languages make that very hard or impossible
  - You cannot write an OS without re-interpreting memory





- C is the (grand-) parent of C++, Objective C, Java, C#, Scala, Python, PHP, Perl, Ruby, ...
- C is not the final word in OS writing, but so far, all others failed. Will **Rust** succeed?

People

- Lecturers
  - Pål Halvorsen
    paalh @ ifi
  - Michael Welzl
    michawe @ ifi
  - Kristjon Ciko
    kristjoc@uio.no
  - Carsten Griwodz
    griff @ ifi
  - Özgü Alay
    ozgua@ifi
  - Cagri Erdem
    cagrie @ ifi








- Group teachers

Andreas Marelius Stachnik Hansen
Birk Bergerud-Nygård
Alexander Gran Makarov
Sondre Sundby Rosslund
William Henrik Behn
Lars Nøvik

Administrative questions by email

Teaching structure

Theory



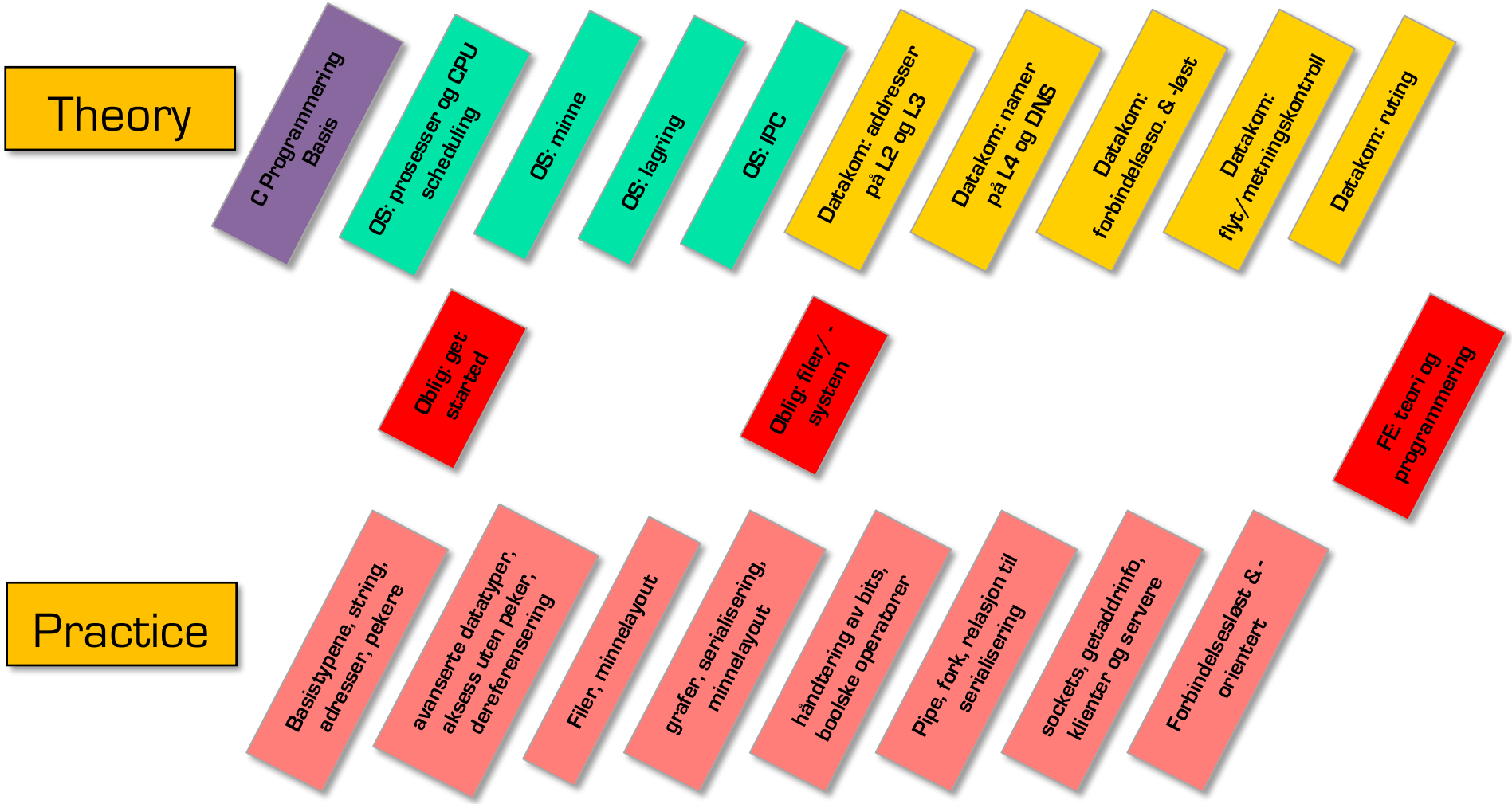
- Input
  - Weekly lectures
  - Weekly slides
  - Tanenbaum's books

Practice



- Input
  - C programming education
  - Assignment help
  - Weekly exercises

Connecting the dots



Teaching structure

Theory

Voluntary

Attend «lecture» session

Ask theory questions

Practice

Voluntary

Attend «group» session

Solve programming tasks

Ask theory & programming questions

Ask for help

Mandatory

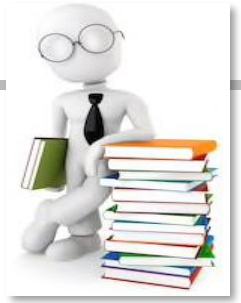
2x semester mandatory programming exercise

deliver individually

Examination

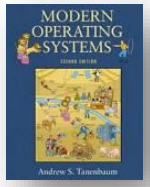
4-hour exam, both theory (mostly no-free text questions) and programming (mostly comprehension) questions

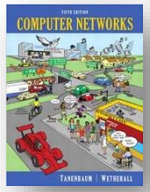
Pensum



- Topics taught in the videos, slides and live sessions together with the given slides and exercises constitute the "pensum"

- Supporting books
  - Modern Operating Systems:
    Andrew S. Tanenbaum
  - Computer Networks:
    Andrew S. Tanenbaum
  - C Notes for Professionals book:
    editor: web@petercv.com
  - Other useful C books:
    - The C programming Language:
      Brian Kernighan and Dennis Ritchie
    - C: A Reference Manual:
      Samuel P. Harbison and Guy L. Steele







Information

- Course homepage in Vortex
  - your starting point
  - messages & news
  - schedule
  - videos in Panopto



- Devilry
  - for submitting your responses to mandatory assignment

- Inspera
  - for final exam (theory + programming)

Exams and Mandatory Exercises

- Mandatory exercises
  - two C programming exercises
    - getting started with C
    - Learning the basic concepts
  - tentative dates: delivery 09.02. + 09.03.

- Final exam
  - focus on the concepts as well as programming
  - date 27.05.
  - 4 hours, 15:00-19:00



Evaluation and grading

- Passing the mandatory assignments is a prerequisite for admission to the **final** exam
  - Passing mandatory assignments in early years qualifies for taking the exam.

- Grading:
  - **No homeexam in 2026!**
  - Final exam: around 70% theory and 30% programming

Questions?

