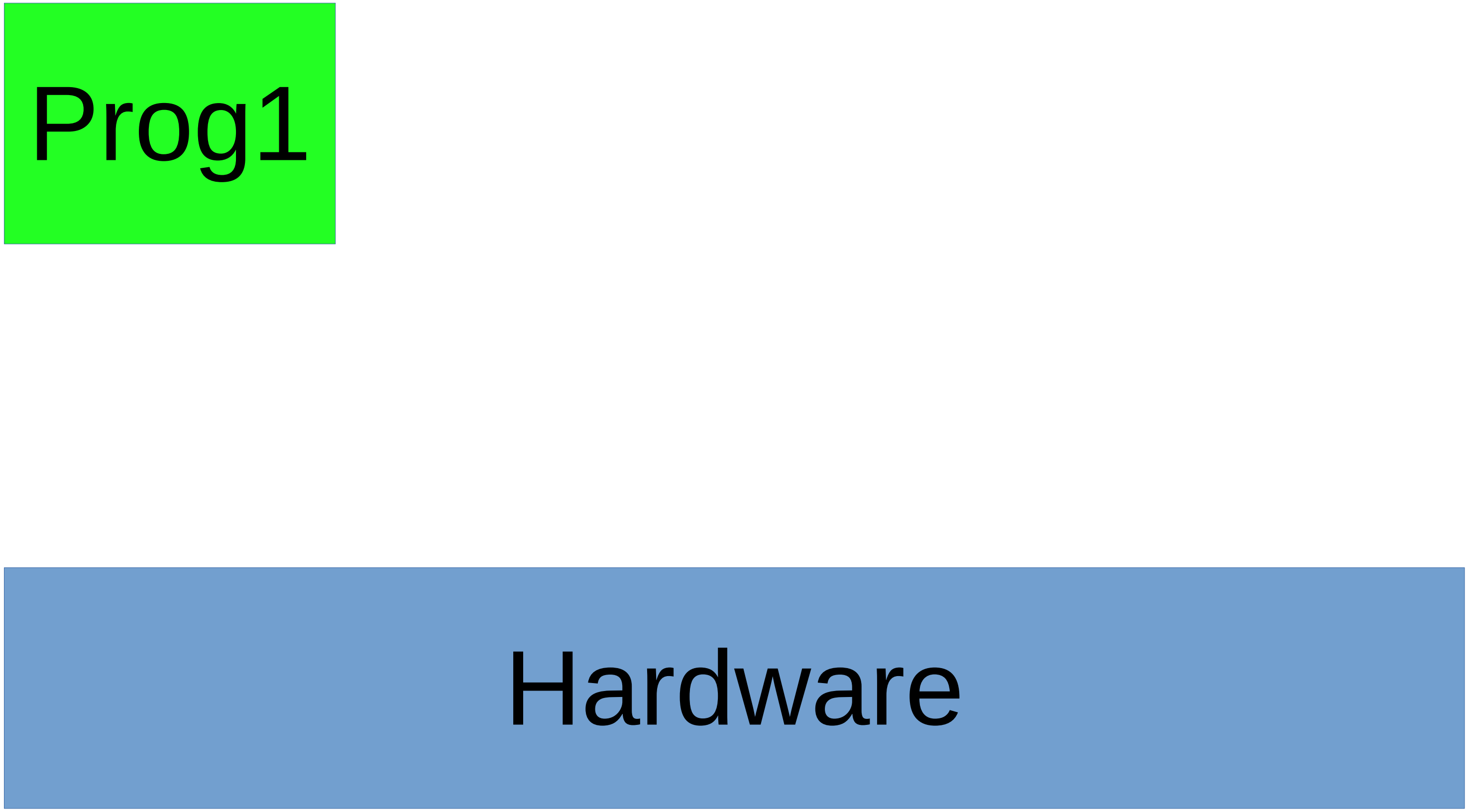
# TDDE68/TDDE47
# Concurrent Programming and Operating Systems

## Lecture 1:
## Introduction, interrupts and system calls

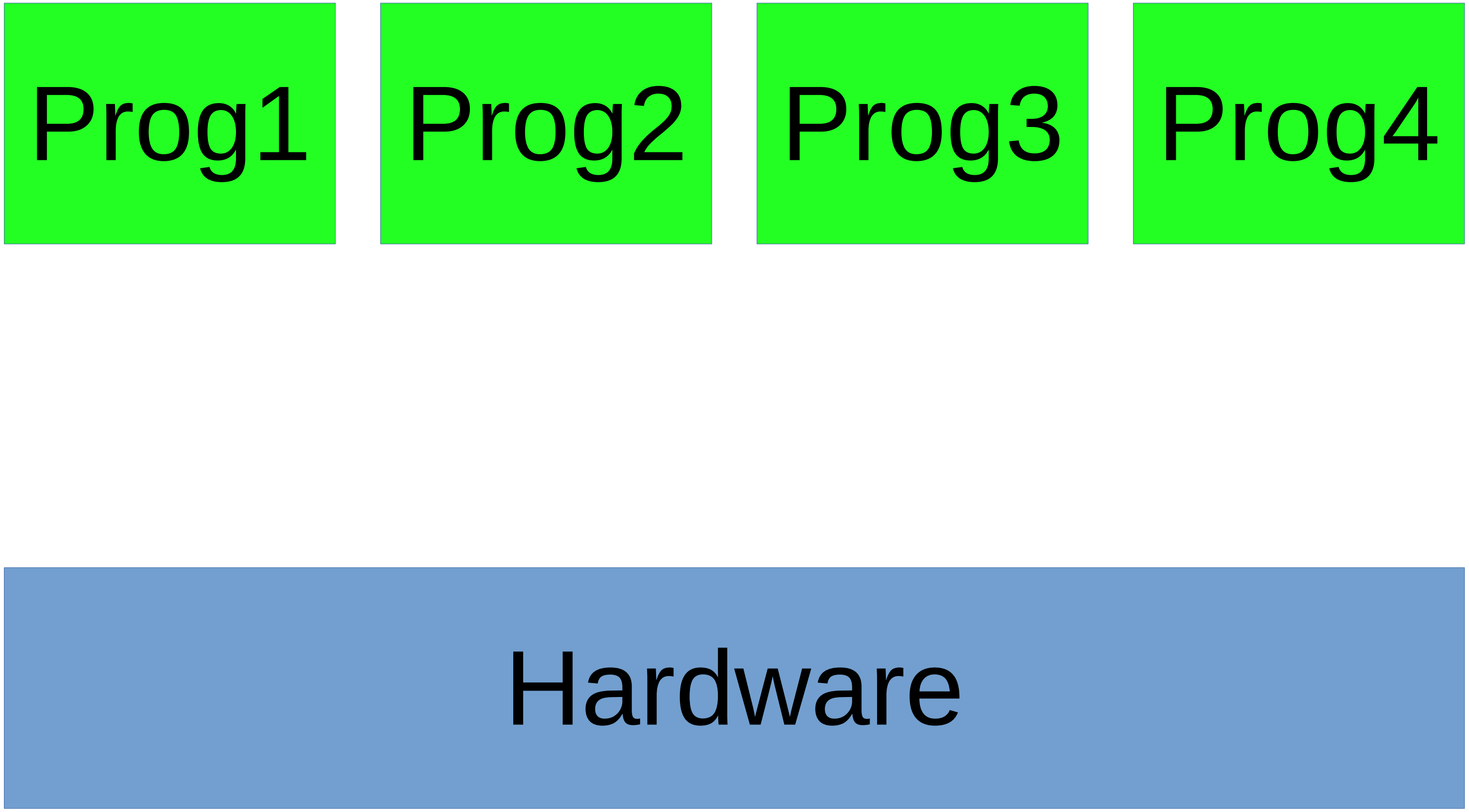
Mikael Asplund

Hardware



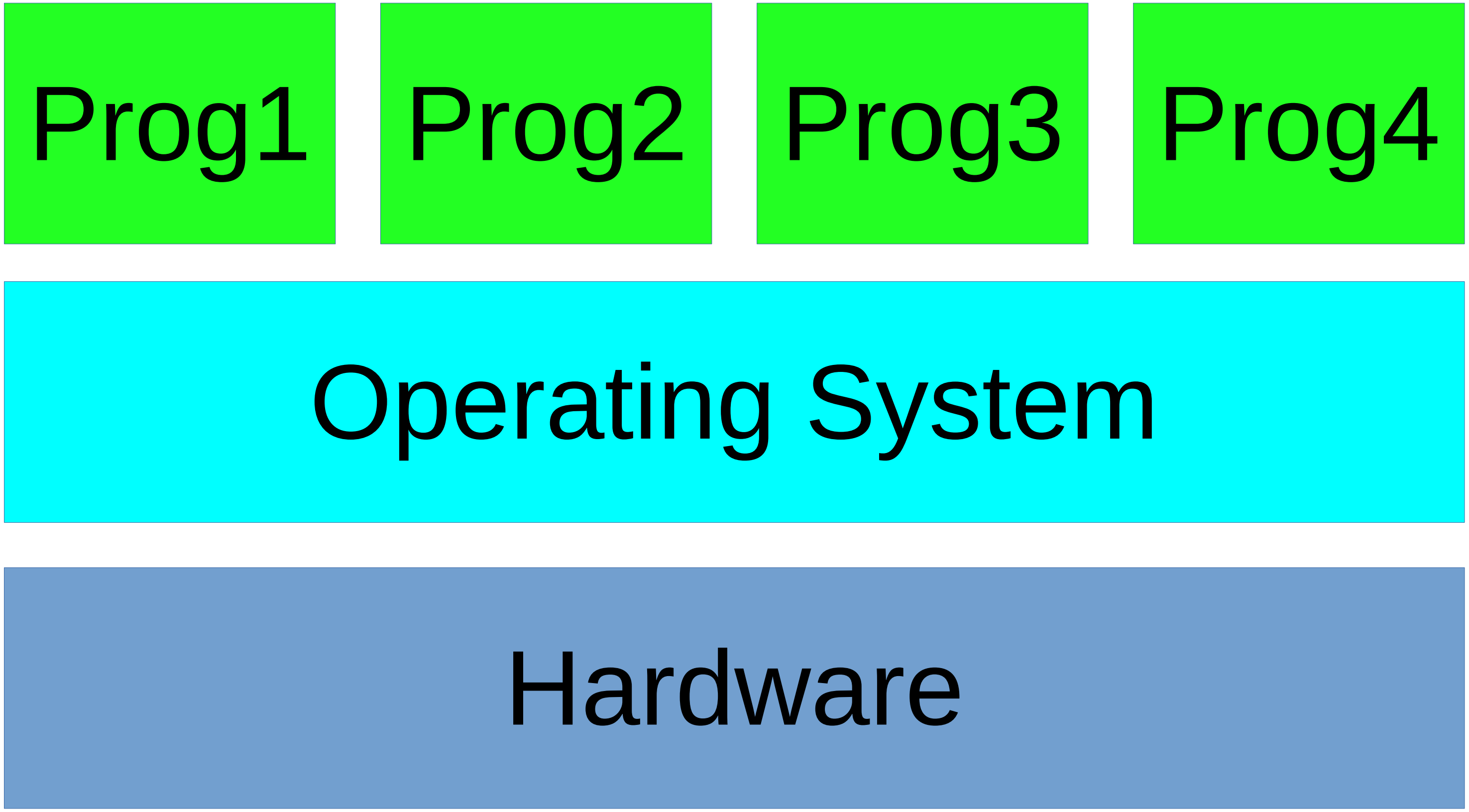
Prog1

Hardware



Prog1 Prog2 Prog3 Prog4

Hardware

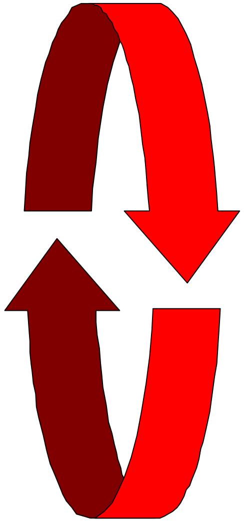


Prog1 Prog2 Prog3 Prog4

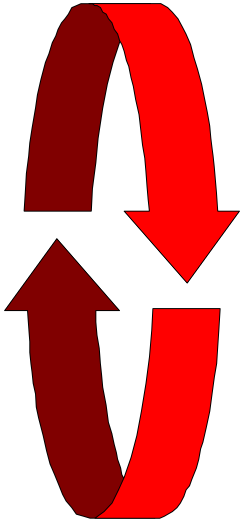
Operating System

Hardware

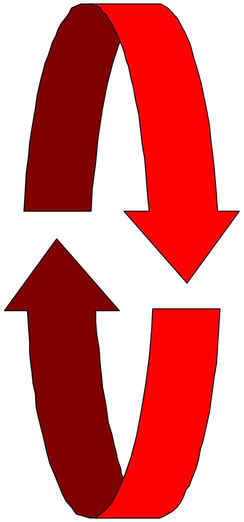
# Concurrency



1. Do this
2. Do that
3. Wait for X
4. Send Y
5. Do something
6. Wait a while
7. Do something else



1. Send Z
2. Wait for X
3. Do nothing
4. Send something else
5. Do this



1. Wait for query
2. Connect to database
3. Send db-query
4. Wait for results
5. Create web page
6. Reply to request

In this course:

Focus on basic principles

# Courses

- TDDE68: 6hp course for D + U + I + M.Sc. + Erasmus
  - C/C++ with pointers a prerequisite
- TDDE47: 8hp course for IT
  - More time and some PBL sessions for learning about memory allocation and pointers
- TDDE47 and TDDE68 have the same exam and labs.

# Course web

- TDDE68:
  - https://www.ida.liu.se/~TDDE68/

- TDDE47
  - https://www.ida.liu.se/~TDDE47/

- Same content – report if anything is strange

# People

- Examiner: Mikael Asplund
- Course leader: Klas Arvidsson
- Course assistant: Dag Jönsson
- Lab assistants:
  - TDDE68 Group A: Christopher Wåtz
  - TDDE68 Group B: Ahmad Usman
  - TDDE68 Group C: Reyhane Falanji
  - TDDE68 Group D: Navya Sivaraman
  - TDDE68: Group E: Malte Nygren
  - TDDE68: Group F: Dag Jönsson
  - TDDE47: Jon Svensson Magnusson

# Communication

1. Use the Teams channel
   a) Lab-related questions in the channel ”Lab-related questions”
   b) Other general things in the general channel

2. Talk/mail with your lab assistant

3. Send an email to cpos-course@groups.liu.se

# Examination

- Labs 3hp for TDDE68, 5hp for TDDE47
- Written exam 3hp
- Bonus assignment 0hp
  - 3p bonus points on the exam
  - Pass all labs no later than **2024-03-08**
  - Need to be first-time registered in VT1 2024

# Lectures vs. Labs

- Intentionally different scope
- Lectures provide an overview of a vast area
- Labs provide hands-on experience with a particular educational OS.

# Labs

- Pintos
  - Stanford educational OS
- Challenging!
- Register in webreg today
  - Only new students!

# Deadlines

- **2024-01-19** Sign up in webreg (new students)
- **2024-03-08** Lab deadline to get the bonus points on the exam
- **2024-03-26** Last chance to pass the labs in VT1
- After VT1, labs can be passed in June or August-September

# Lab feedback

- Our ambition**:**
  - No later than 2w from the soft deadline
  - Lab6 at least one chance to correct if you hand in on the 8th

- Soft deadlines:
  - Lab 0: January 23
  - Labs 1 and 2: February 6
  - Labs 3 and 4: February 20
  - Labs 5 and 6: March 8

# Time management

- Lectures 8*2h + 1*4h
- Lessons 3*2h (to prepare for labs)
- Labs 36h (42 for TDDE47)
- Self-study TDDB68: ~100h, TDDE47: ~145h

# Time management

- Le
- Le
- La
- Self-study TDDB68: ~100h, TDDE47: ~145h

If you’re not spending **10+ hours/week** of free time doing labs and studying for this course,
you will be in trouble.

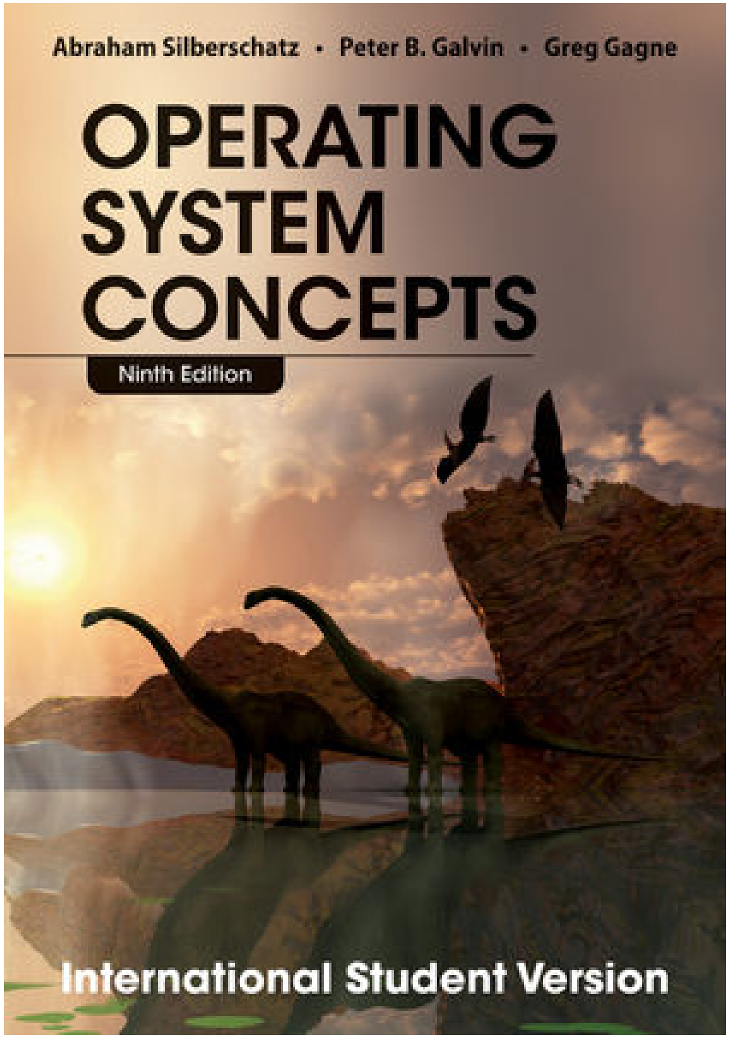
# Do not cheat!

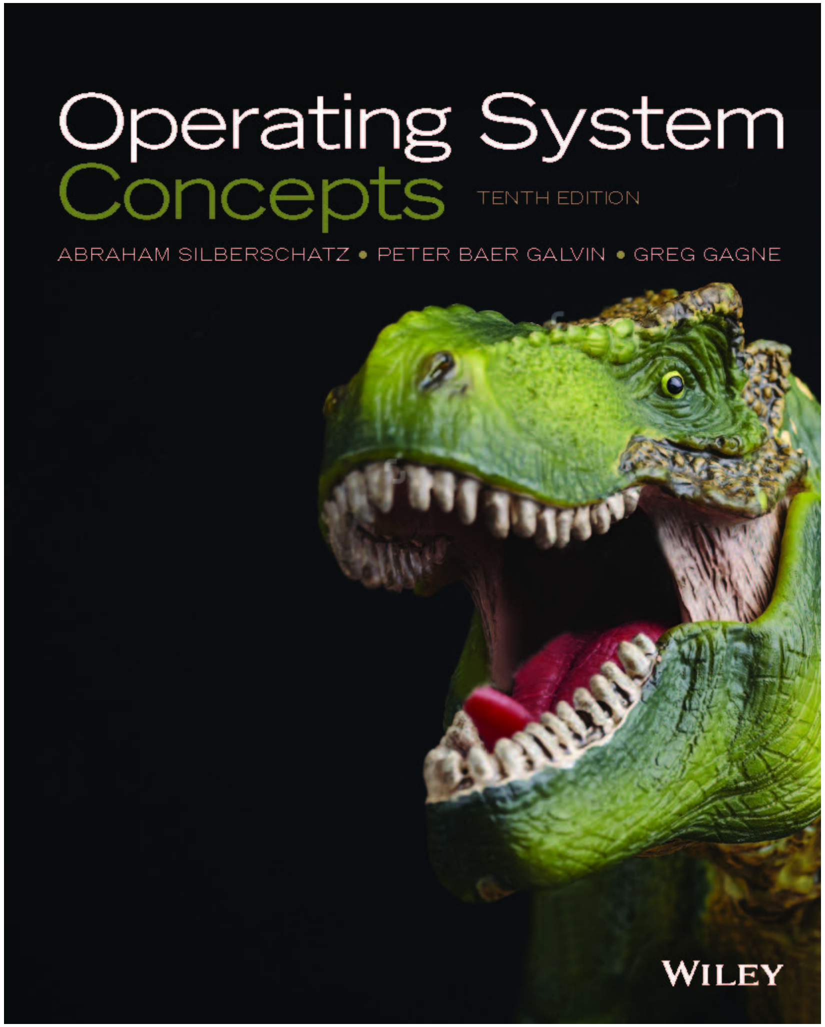
- Labs are hard and you might find solutions online

- It is **not allowed** to copy anyone else’s solution

- We are obliged to report suspected plagiarism to the disciplinary board

# Course book


Abraham Silberschatz • Peter B. Galvin • Greg Gagne
OPERATING SYSTEM CONCEPTS
Ninth Edition
International Student Version


Operating System Concepts
TENTH EDITION
ABRAHAM SILBERSCHATZ • PETER BAER GALVIN • GREG GAGNE
WILEY

# Two comments:

”Put more emphasis on telling the students to read the book. I didn't understand much during the lectures because I didn't read the book until the exam period. Had I read the pages before each lecture, the lectures would have been more productive.”

”... And the book was actually really good - I wish I had started reading it before the exam period.”

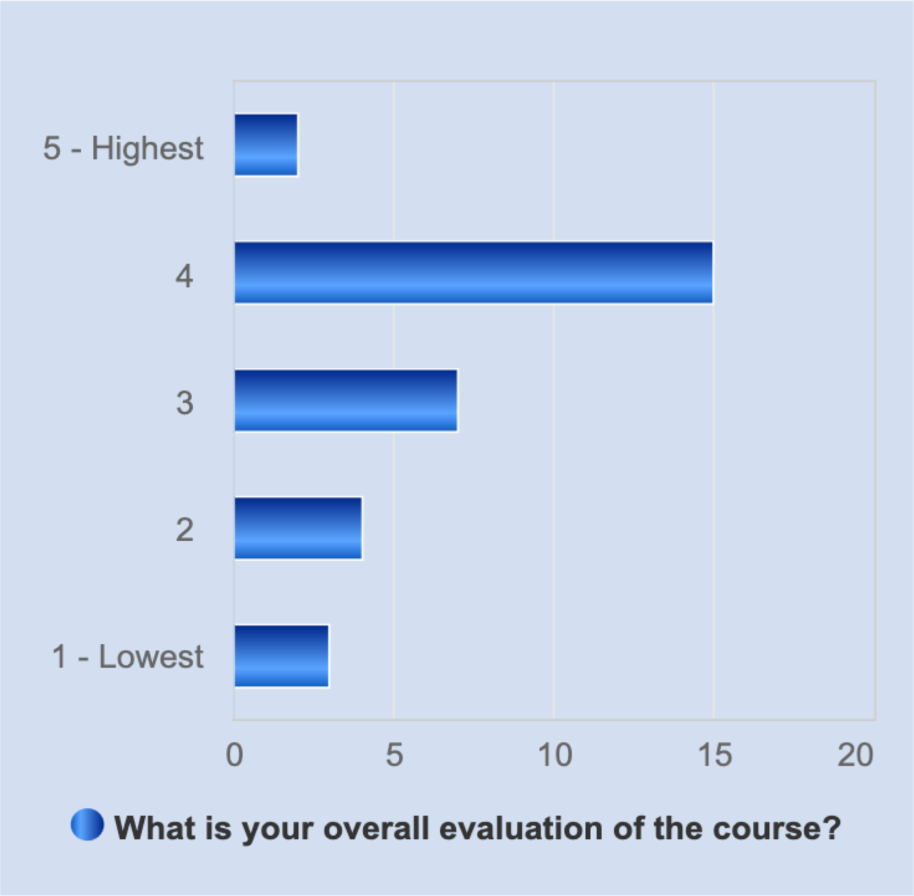
# Exam

- Previous exams available
  - Solutions **not available**

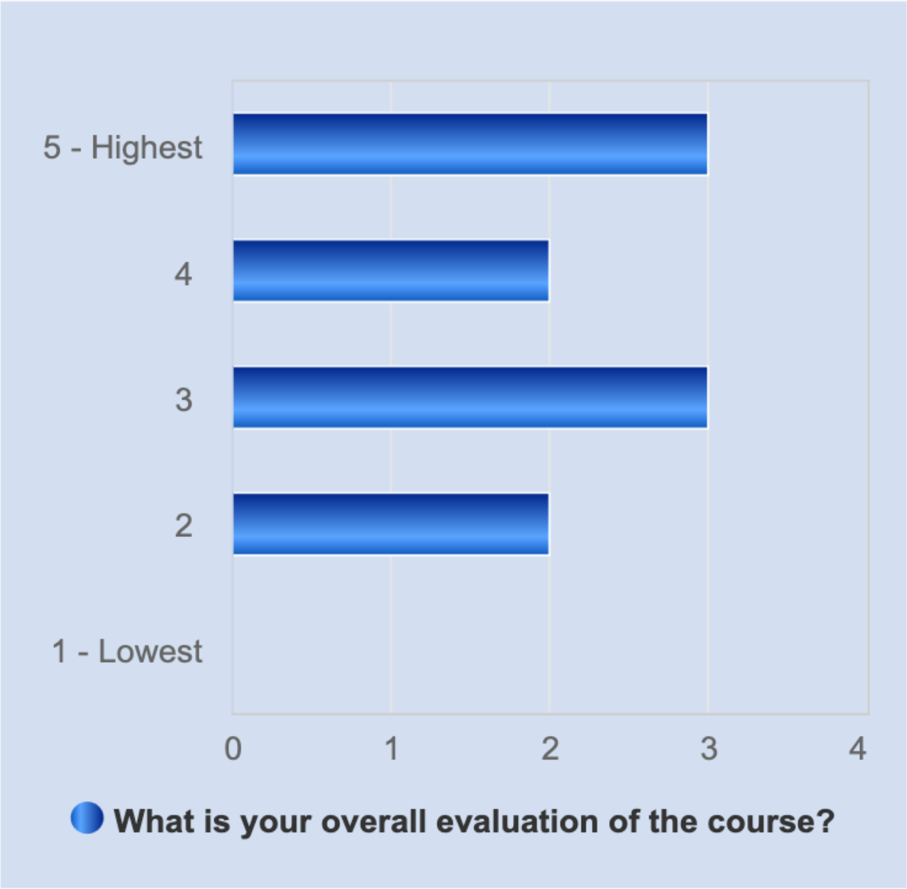
- Two parts
  - Part A: Basic questions, enough for grade 3
  - Part B: Ability to analyze and compare, used for grades 4 and 5

- Grading criteria
  - Grade 3: At least 12 points on part A.
  - Grade 4: At least 28 points in total.
  - Grade 5: At least 34 points in total.

# Results from Evaliuate



TDDB68



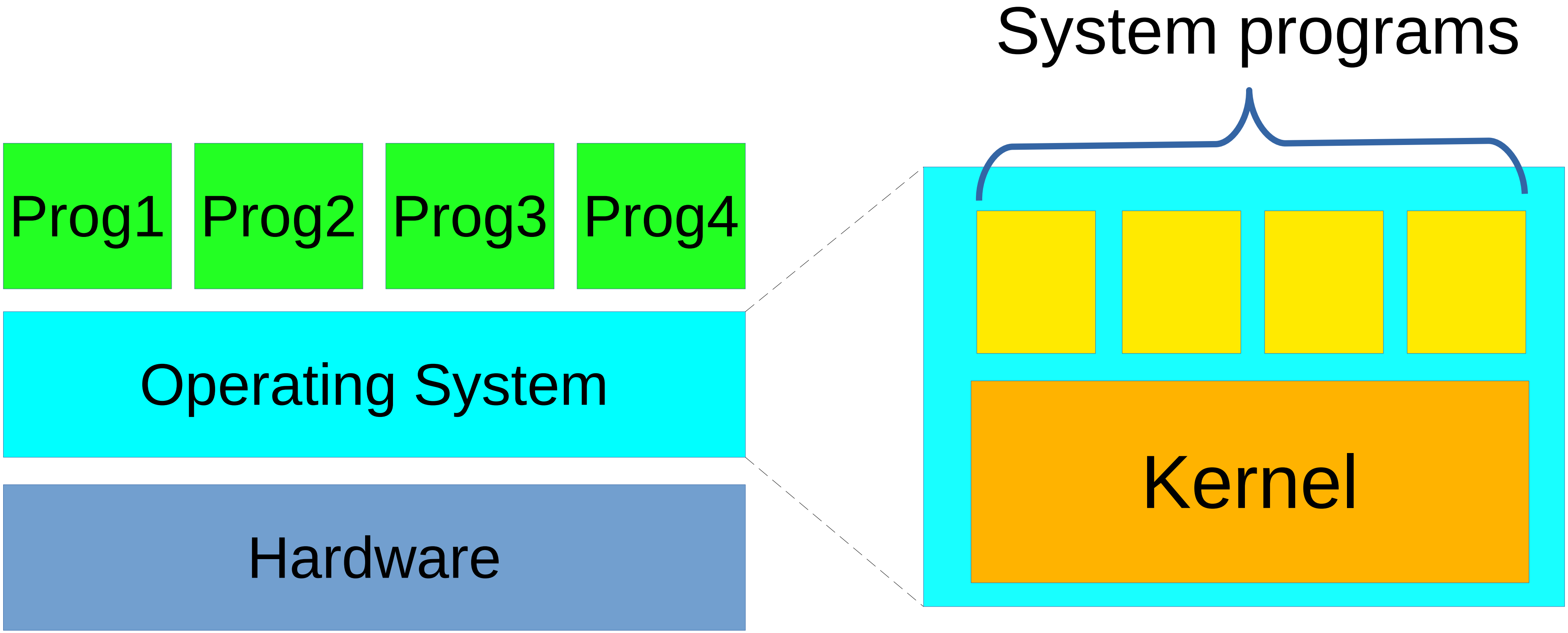
TDDE47

# Planned changes

- New course leader (and lecturer)
- Some reordering in the labs
- Smaller fixes

What does an operating system do?

# OS basic structure



# System programs vs kernel

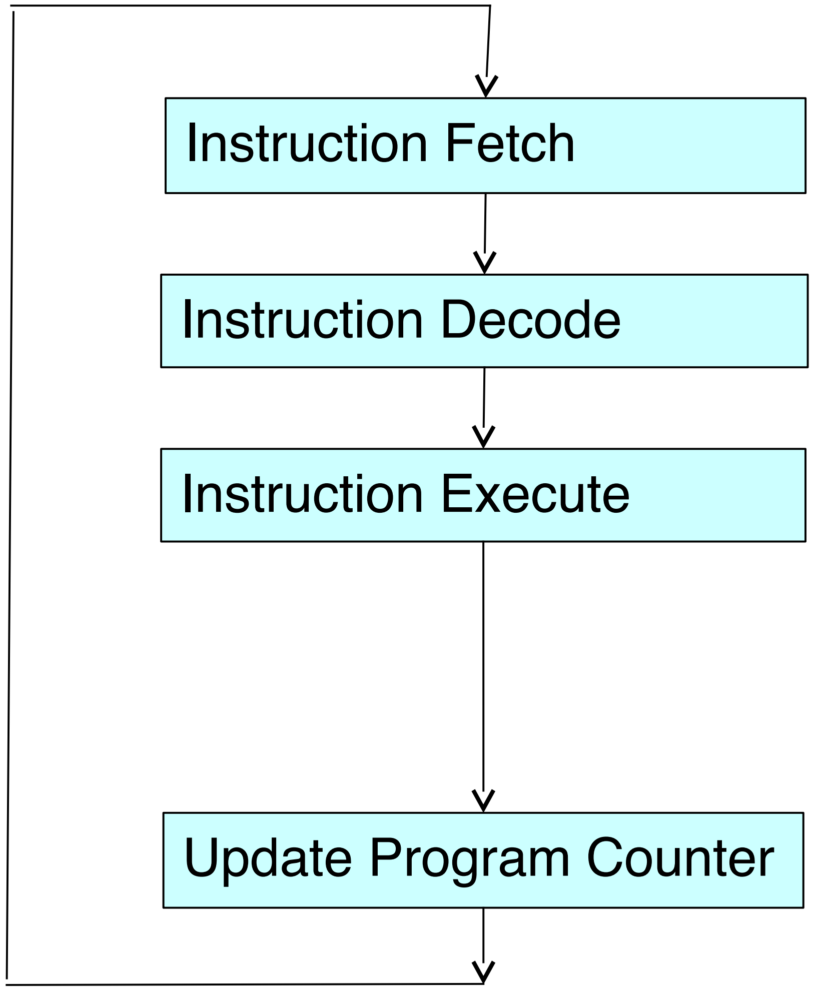
**System programs:**

- User's view of OS
- Provide a convenient environment
  - File management
  - Program development
  - Graphical user interface
  - ...

**Kernel:**

- Low-level management
  - Less visible, more important
- Controls and manages resources
  - Process management
  - Scheduling
  - Memory management
  - File system control
  - ...

# Program execution (von Neumann cycle) by a processor



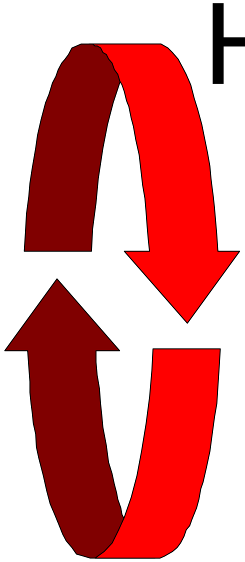
Instruction Fetch

Instruction Decode

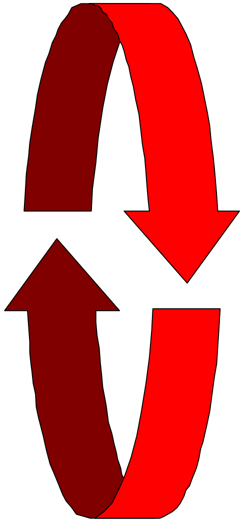
Instruction Execute

Update Program Counter

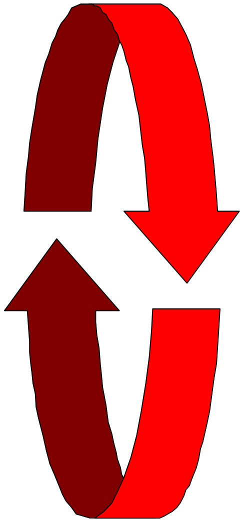
# How to do concurrency?



1. Do this
2. Do that
3. Wait for X
4. Send Y
5. Do something
6. Wait a while
7. Do something else

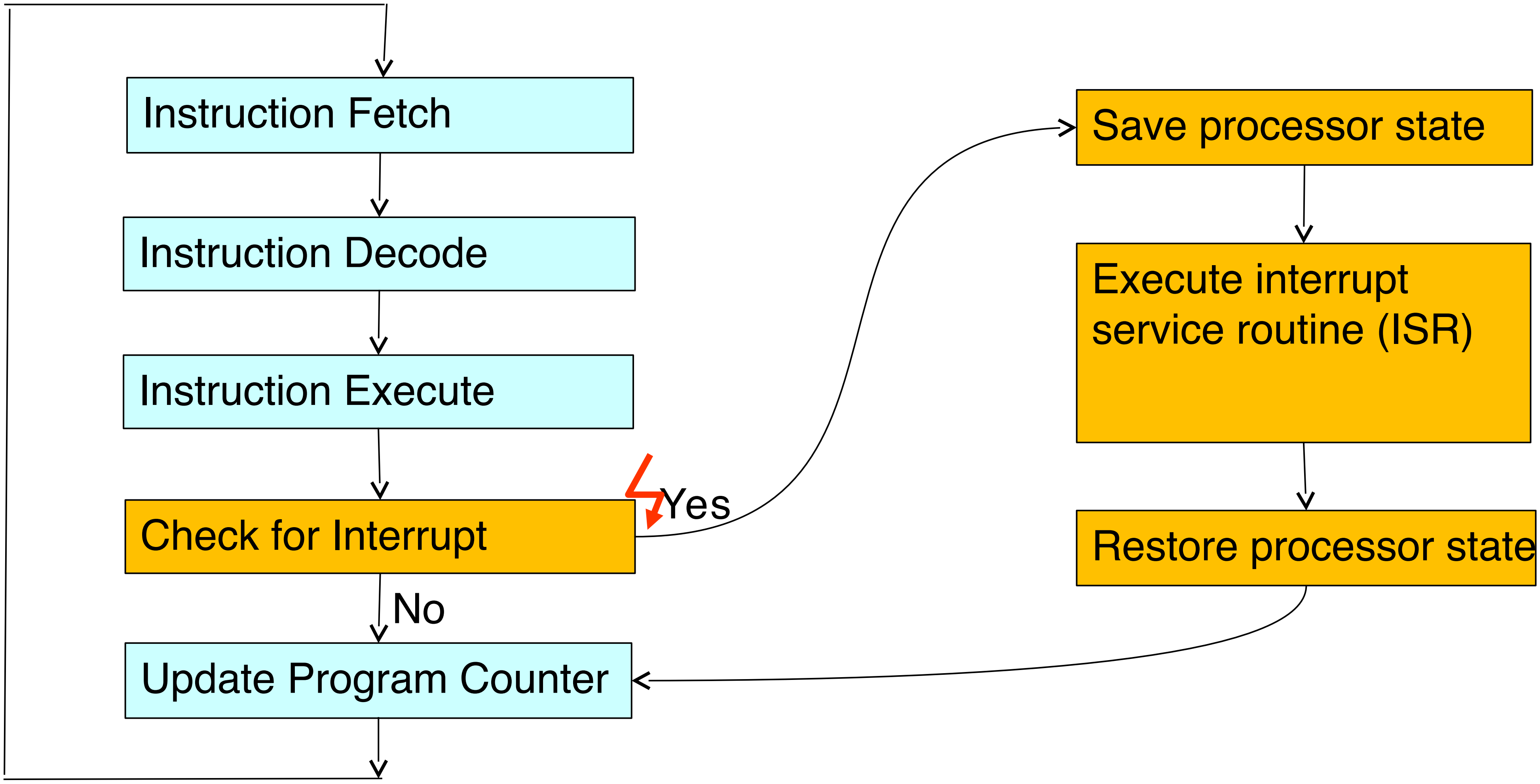


1. Send Z
2. Wait for X
3. Do nothing
4. Send something else
5. Do this



1. Wait for query
2. Connect to database
3. Send db-query
4. Wait for results
5. Create web page
6. Reply to request

# Interrupts


Instruction Fetch
Instruction Decode
Instruction Execute
Check for Interrupt
Yes
No
Update Program Counter
Save processor state
Execute interrupt service routine (ISR)
Restore processor state

# Software Interrupts

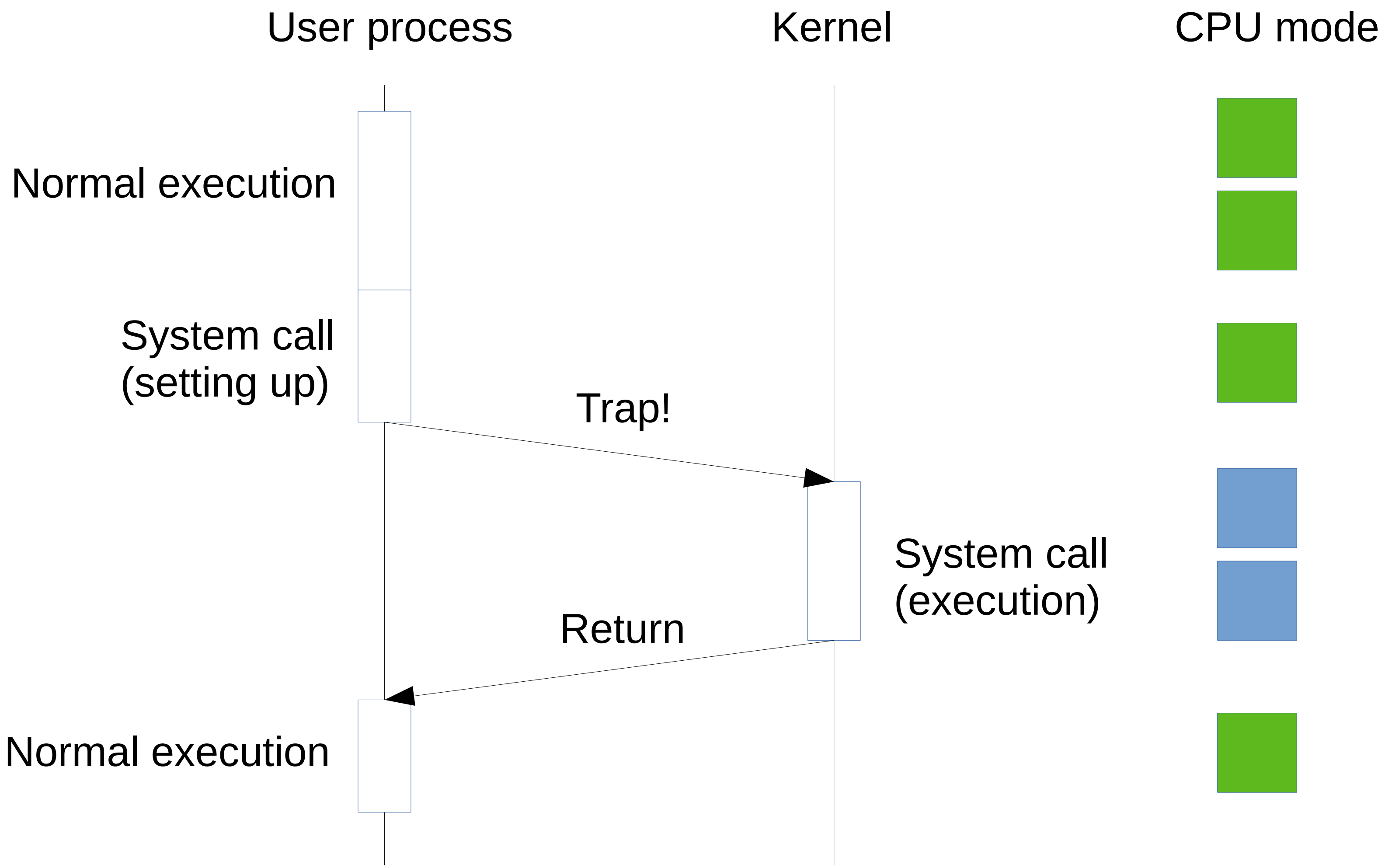
- A ***trap*** is a *software*-generated interrupt caused either by an error or a user request.
  - Examples: Division by zero;
    Request for OS service

An operating system is ***interrupt* driven**.

Why not just have a function API?

# Dual Mode

# Dual mode



User process

Kernel

CPU mode

Normal execution

System call
(setting up)

Trap!

System call
(execution)

Return

Normal execution

🟩 User mode

🟦 Kernel mode

# Types of System Calls

- **Process control**
  load, execute, end, abort, create, terminate, wait ...
  memory allocation and deallocation
- **File management**
  open, close, create, delete, read, write, get/set attributes...
- **Device management**
  request / release device, read, write, ...
- **Information maintenance**
  get / set time, date, system data, process / file attributes
- **Communications**
  create / delete connection, send, receive, ...

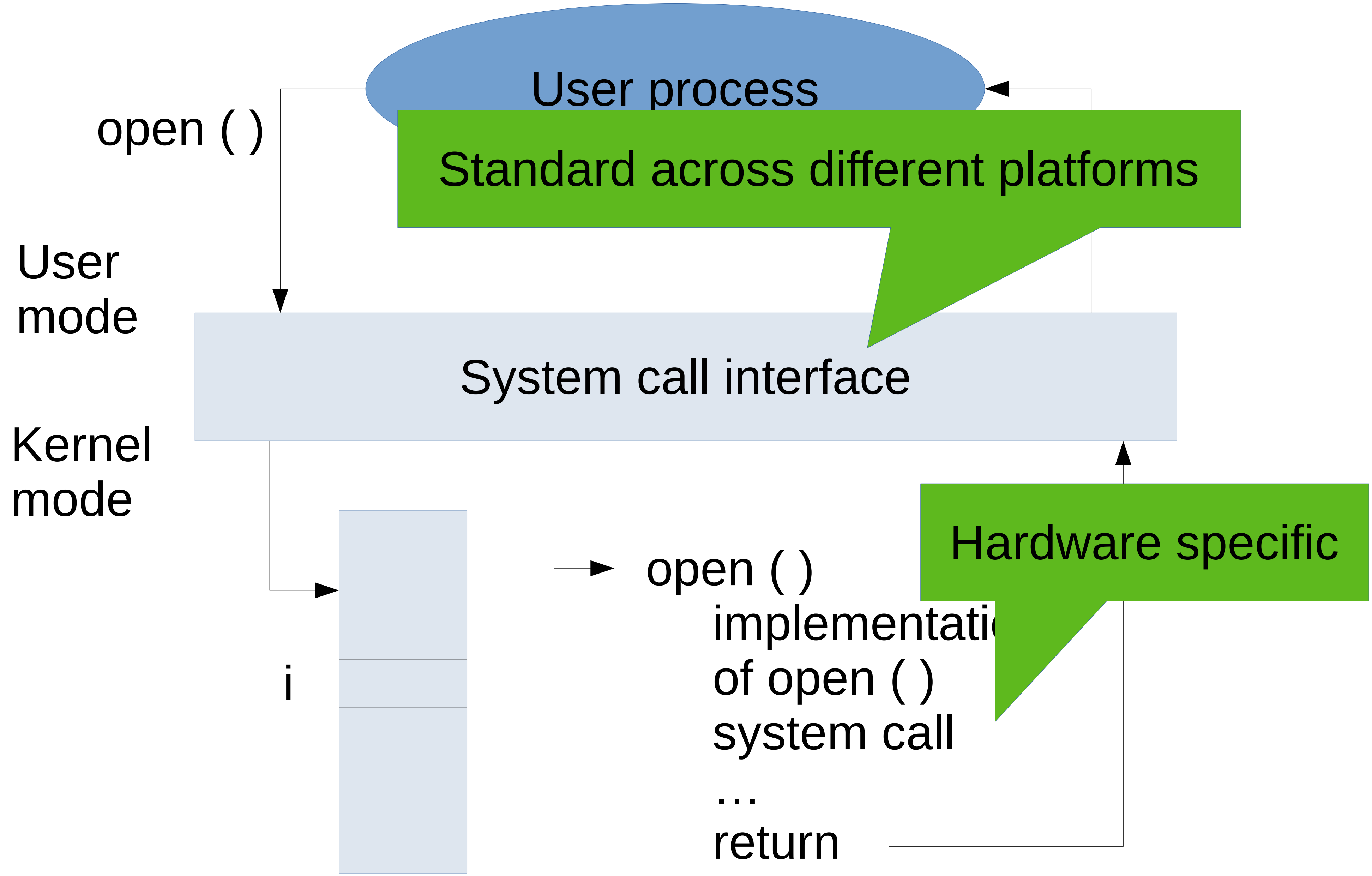
# Syscalls In Linux

(man syscalls)

Tracing system calls

Linux – ltrace
Mac OSX  - dtrace

# System Call API – OS Relationship



User process

open ( )

User mode

System call interface

Kernel mode

i

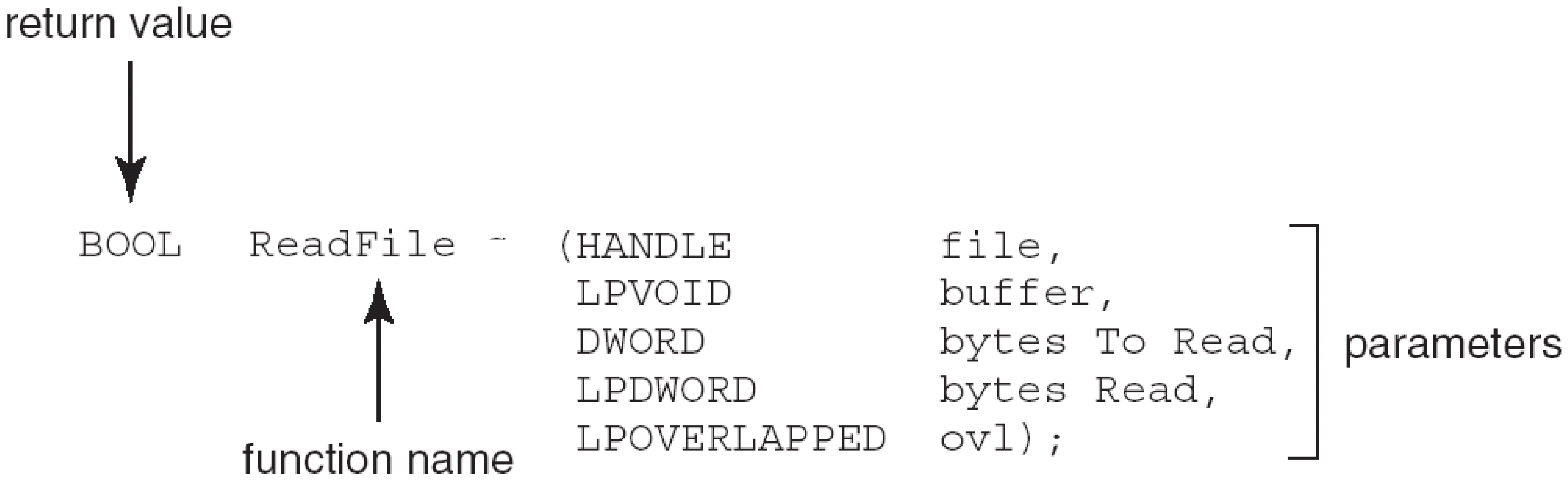
open ( )
implementati
of open ( )
system call
…
return

Standard across different platforms

Hardware specific

# Example of a System Call API

- ReadFile() function in Win32 API   (function for reading from a file)



```
BOOL   ReadFile   (HANDLE         file,
                   LPVOID         buffer,
                   DWORD          bytes To Read,
                   LPDWORD        bytes Read,
                   LPOVERLAPPED   ovl);
```

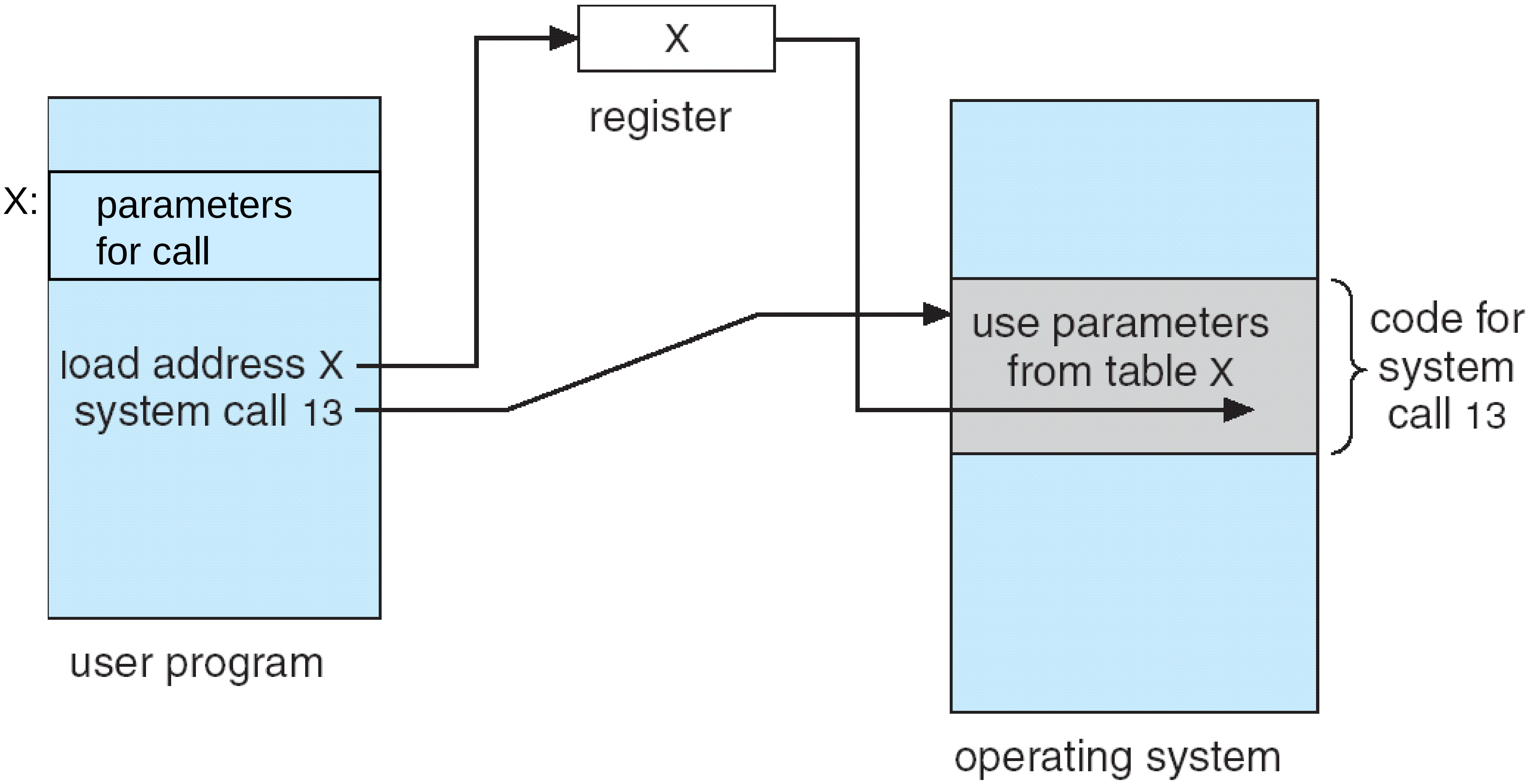
- Parameters passed to ReadFile():
  - file — the file to be read
  - buffer — a buffer where the data will be read into and written from
  - bytesToRead — the number of bytes to be read into the buffer
  - bytesRead — the number of bytes read during the last read
  - ovl — indicates if overlapped I/O is being used

# System Call Parameter Passing

- Three general methods used to pass parameters to syscalls:
  - Parameters **in *registers***
  - Parameters **in a *block in memory***, and address to block in a register
    - This approach taken by Linux
  - Parameters ***pushed onto the stack***

- What are the advantages/disadvantages?

# System Call Parameter Passing via Block



# Summary

- **Operating System** = OS Kernel + System Programs
  - Mediates all accesses to system resources
  - **Interrupt**-driven
    - Error handling
    - Controlled access to system resources, e.g.
      - I/O devices, DMA
      - CPU time sharing
    - …
- **Dual-Mode** (user mode, kernel mode)
  - **System Call** API for portability

# Reading guidelines

- 9th edition
  - Chapter 1: Sections 1.1-1.7
  - Chapter 2: 2.3-2.5

- 10th edition
  - Chapter1: 1.1-1.5
  - Chapter2: 2.3-2.4

# What's next?

- **Today**
  - Sign up in Webreg!
  - Get the book
- **Tomorrow**
  - 13-17 Introductory C lecture
  - Try some C programming
- **Thursday**
  - 8-10 lesson to introduce the labs
- **Friday**
  - First lab (lab0)