COMPUTING PLATFORMS

Practicalities

TKT20016 COMPUTING PLATFORMS (5 ETCS)

- **Contents:** Computing platforms' structure and functionalities, processes and threads, virtualization of CPU and memory and entire machines, concurrency and inter process communication, file systems, security in computing platforms
- **Prerequisites:**
  - TKT10005 Computer Organization I or TKT10006 Computer and Internet
  - TKT10002 Introduction to Programming and TKT10003 Advanced course in Programming
  - Strongly recommended: TKT200011 Data Structures and Algorithms 1
- Replaces the old TKT20003 Operating Systems course
  - Lots of overlapping content
  - You can include only one of Computing Platforms or Operating Systems (**not both!**) in your degree

LANGUAGE OF THE COURSE

- Computing platforms is a compulsory course both in the Bachelor programme of Computer Science and Computer and Data Science study track in the Bachelor Programme in Science (English language programme)
  -> Therefore the main language of the course is English
- Exercise groups also in Finnish
- You can submit exercise solutions / review question solutions / exam answers in English, Finnish or Swedish
- An opportunity for Finnish language students to learn working in an English language environment!

PRACTICAL ARRANGEMENTS

- **Lectures:** Mondays 14:15-16:00 and Tuesdays 12:15-14:00 (in English) in A111 by Leena Salmela, Jussi Kangasharju and Tiina Niklander (best effort will be made to provide a recording)
- **Exercise sessions** (choose one, first sessions Jan 22nd) by Kirke Ruusalu / Reima Kuosmanen / Heljä Räisänen / Leena:
  - Group 1, Wed 10:15.12:00 (Kirke, in English)
  - Group 2, Wed 12:15-14:00 (Kirke, in English)
  - Group 3, Thu 14:15-16:00 (Heljä, in Finnish)
  - Group 4, Thu 16:15-18:00 (Reima, in Finnish)
  - Group 5, Fri 10:15-12:00 (Reima, in Finnish)
  - Group 6, Fri 14:15-16:00 (Leena, in English, online)
- **Workshop/Paja:** Tue 14:15-16:00 by Heljä (get help for solving the exercises)
  - First workshop / paja on Jan 21st
- **Exam:** March 4th 9:00-11:30 (electronic hall exam)

FEEDBACK

- Changes based on feedback from last year:
  - Some practical and more hands-on exercises have been added
  - For each topic indicate a primary material (+ potentially additional materials) or indicate that there are alternative materials you can choose from
- We hope to get feedback from the students both during and after the course.

GENERAL POLICIES

- All **deadlines** are **strict**
- **Exam is mandatory**, all other activities are voluntary (as long as you obtain enough points to pass)
- Highly recommended to make use of teaching provided (e.g. lectures, exercise sessions, workshop/paja) but you can also use other means (read the books, work on coding problems (check the OSTEP book), watch online courses,…)

PLAGIARISM AND CHEATING

- **Strict policy: No plagiarism allowed, no cheating allowed!**
- Read the policy at https://studies.helsinki.fi/instructions/article/what-cheating-and-plagiarism
- Notice that working together with someone (or in a group) to solve exercises during the course is definitively OK, as long as everyone prepares their own solutions
- Exam is open book but no collaboration is allowed

USE OF LARGE LANGUAGE MODELS

- Permitted for information retrieval or explaining and summarizing topics.
- Prohibited for text generation to create exercise or review question solutions.
- Prohibited for generating program code.
- Completely prohibited in the exam.
- If/when you use language models, you must report which model you have used and what you have used it for.

MATERIALS (I.E. WHAT YOU NEED TO STUDY)

- Remzi H. Arpasi-Dusseau & Andrea C Arpasi-Dusseau, *Operating Systems: Three Easy Pieces,* v. 1.10
  - Free pdf: ostep.org
- William Stallings, *Operating Systems – Internals and Design principles*, 9th ed, Pearson 2018
- See the lecture schedule in Moodle for the book chapters to read!
- Lectures and lecture slides
- Exercises and their solutions
- Review questions and their solutions

GRADING SCHEME

- **Exam:** max 42 points
- **Exercises and review questions:** max 18 points divided as follows
  - **Exercises:** max 12 points
  - **Review questions:** max 6 points
- **Bonus points:**
  - **Active participation in exercise sessions:** max 6 points
  - **Feedback form (at the end of the course):** max 1 point
- **To pass (with grade 1):**
  - Exam points ≥ 21
  - Total points ≥ 30
- To obtain grade 5, at most 54 points are needed.

EXERCISES

- Sign up to one session (if necessary you can visit another group)
  - One remote group, rest are on site on campus
- Procedure:
  - Solve the exercises and submit your solutions (pdf) to Moodle (DL: Tue 23:00)
  - Participate actively in the exercise session (**active participation = 1 course point per week**)
  - Self-evaluate your solutions in Moodle (each exercise on scale 0-2, where 1=partial solution / real attempt / good try, 2=satisfactory / good solution (i.e. does not need to be perfect))
- **You only get points for exercises if you self-evaluate them!**
- Exercise points scaled (linearly) to course points, so that ~80% of exercise points = 12 course points, ~6.7% of exercise points = 1 course point)

REVIEW QUESTIONS

- Review questions where you need to **explain central concepts** related to computing platforms in your own words
- Good study technique: make notes when you study; These review questions encourage you to adapt this study technique.
- **Make hand-written answers** (**required!,** helps to process your learning), take a photo / scan them, and submit to Moodle
- Weekly deadline Tue 23:00; self-evaluated
- The number of questions / points available per week may vary!
- **You only get points for review questions if you self-evaluate them!**
- Review question points scaled (linearly) to course points, so that ~80% of review question points = 6 course points, ~13.3% of review question points = 1 course point)

PAJA / WORKSHOP

- Here you can get help with the exercises and review questions
- Tuesdays 14:15-16:00 in BK106

SUMMARY: WEEKLY SCHEDULE

- Lectures on Mon and Tue
- Paja / Workshop Tue 14-16
- Tue 23:00 – DL for review question submission to Moodle
- Tue 23:00 – DL for self-evaluation of previous week's review questions
- Tue 23:00 – DL for exercise submission to Moodle
- Tue 23:00 – DL for self-evaluation of previous week's exercises
- Wed (~10:00, at latest) - model solutions for review questions available in Moodle
- Exercise groups between Wed 10:15 and Fri 14:15
- Wed (~before noon, at latest) - exercises and review questions for next week available in Moodle
- Fri (~after 16:00) - model solutions for exercises available in Moodle

EXAM

- Electronic hall exam
- Open book (i.e., you can use sources) but you need to write the answers yourself and only by yourself (no direct copying, no plagiarism, no working together with others)
- Use of large language models is prohibited
- Course exam March 4th at 9:00
- Renewal exam April 10th (last exam where your exercise points etc. are valid)
- **You need to separately register for the exams!**
  (Register now to the course exam so you don't forget)

COMMUNICATIONS

- Moodle (News-section) is used for official announcements
- Discord is used for discussion and peer support (course personnel will visit time to time)
- Use email / Moodle messages to contact lecturer
  - Contact Leena Salmela for practical issues

COMPUTING PLATFORMS

Introduction to computing platforms and operating systems

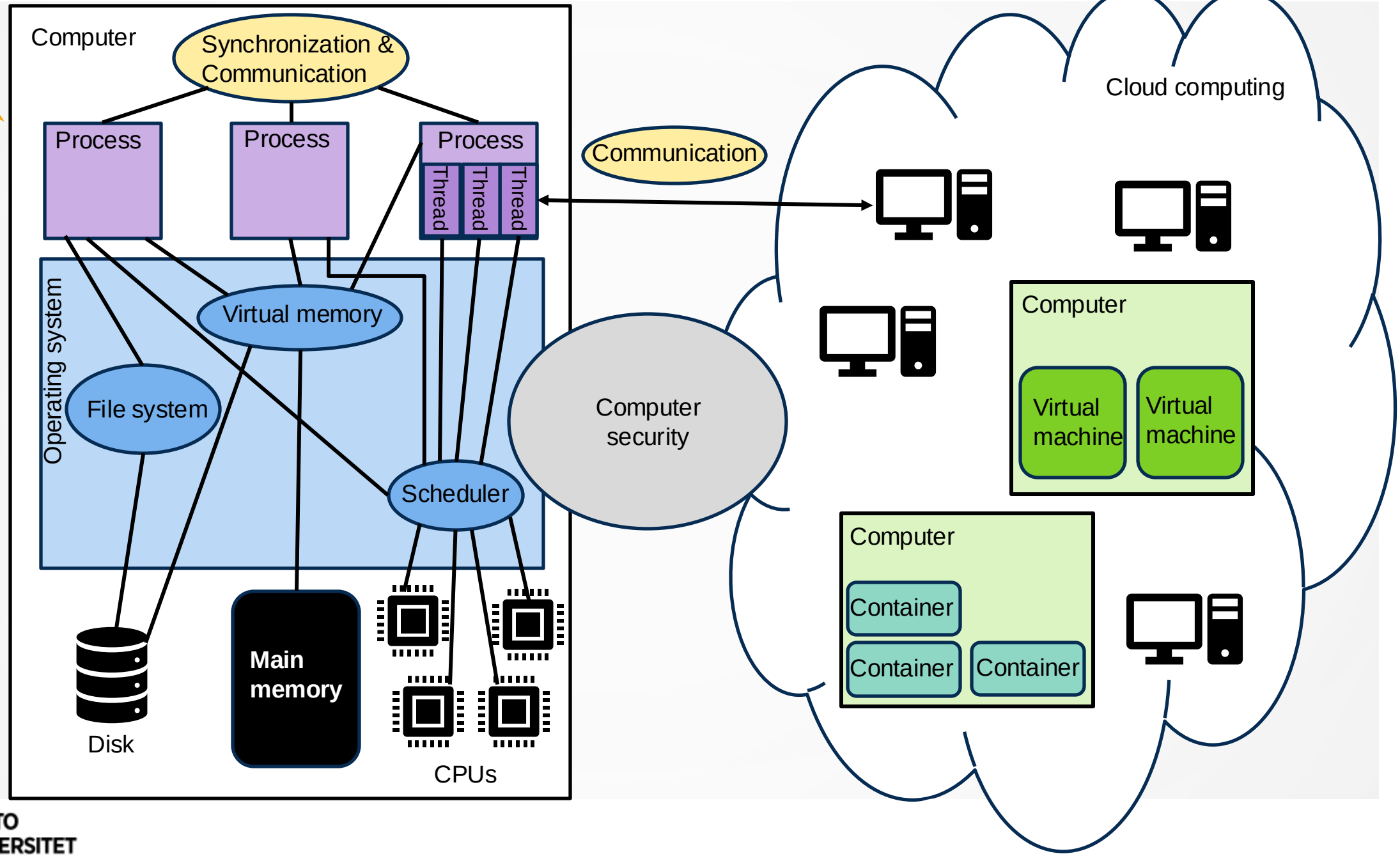
WHAT IS A COMPUTING PLATFORM?

- **Combination of hardware and software** that provides an environment where applications can be executed
- Computing platform can vary a lot depending on their intended use:
  - Personal computer
  - Servers
  - Mobile devices
  - Embedded systems
  - Cloud computing
  - ...

COMPONENTS OF A COMPUTING PLATFORM

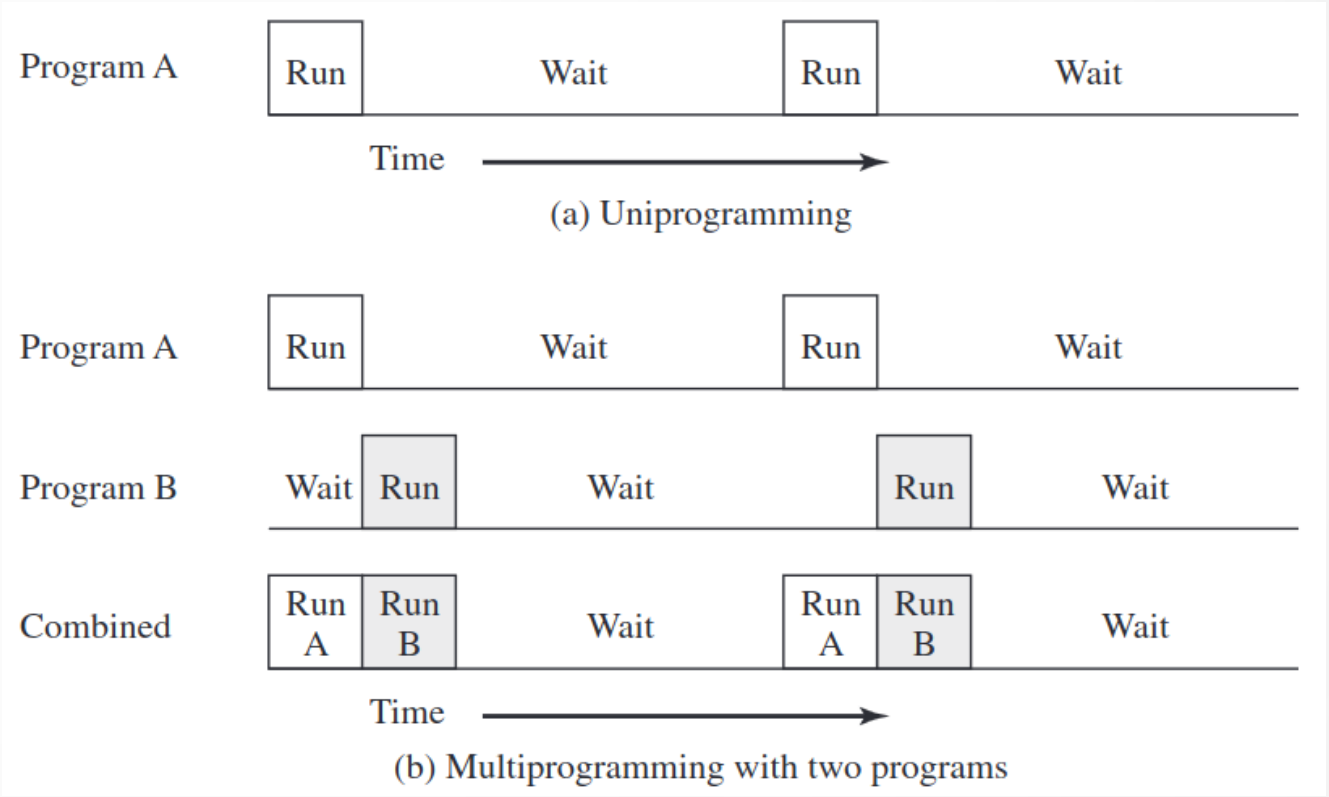
- **Hardware:**
  - PC, mobile devices, servers, cloud, …
  - CPU, memory, disks, SSDs, network, I/O devices, ...
- **Operating system:**
  - Management and allocation of resources
  - Facilitation of communication
  - Manage system security
  - Handle error situations
- **Virtualization and containerization:**
  - Multiple operating systems or applications running on the same hardware
- **Supporting software:**
  - Software frameworks and libraries, development tools, …

CONTENTS OF THE COURSE



SINGLE/MULTIPLE PROCESSES AND SINGLE/MULTIPLE PROCESSORS

- **Uniprogramming:** Single process running on a single processor
- **Multiprogramming:** Multiple processes running on a single processor



Figures from [Stallings, Operating systems: Internals and design principles, 9th ed]

SINGLE/MULTIPLE PROCESSES AND SINGLE/MULTIPLE PROCESSORS

- **Uniprogramming:** Single process running on a single processor
- **Multiprogramming:** Multiple processes running on a single processor
- **Multiprocessing or multitasking:** Multiple processes running on multiple processors
- **Multithreading:** Process divided into threads that can run concurrently
- **Distributed processing:** Multiple processes at the same time on connected systems, many complete systems that are interconnected

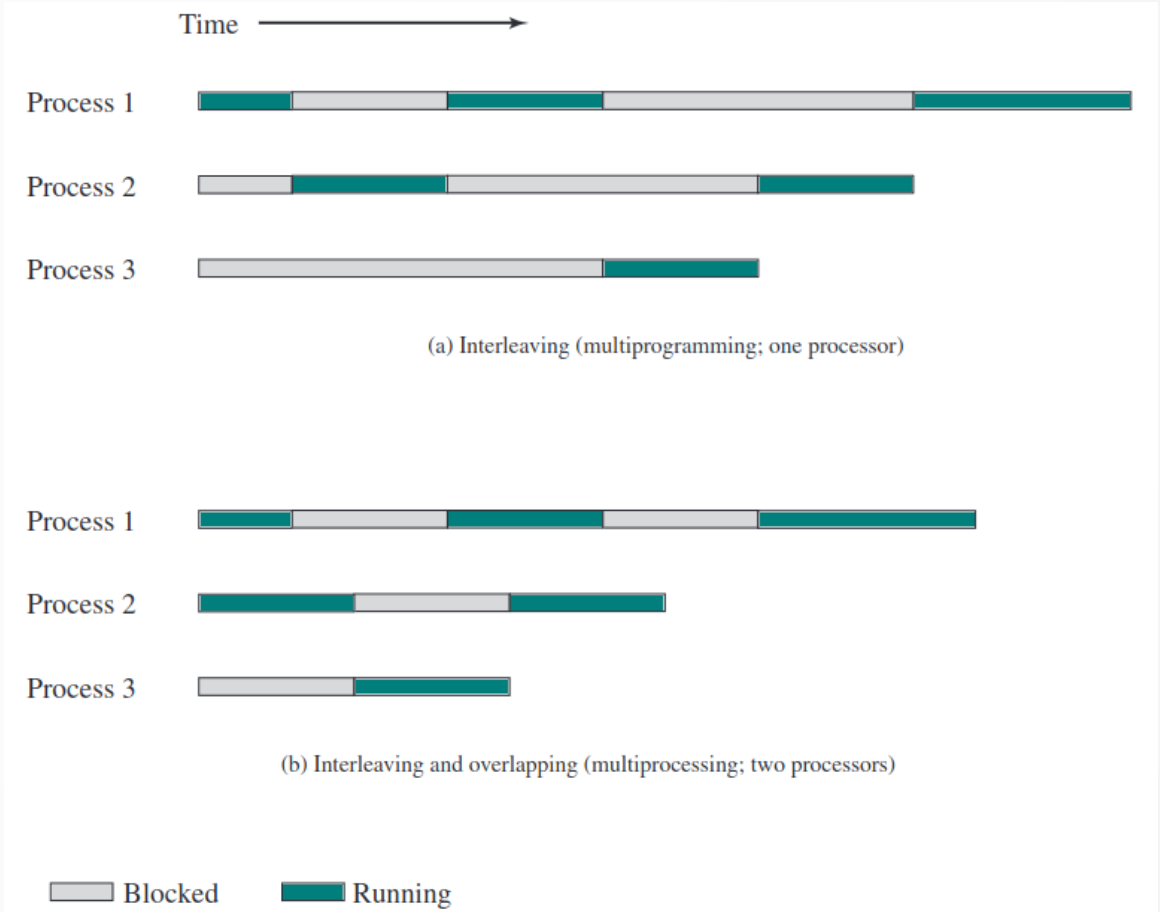


Figure 2.12 Multiprogramming and Multiprocessing

Figures from [Stallings, Operating systems: Internals and design principles, 9th ed]

BATCH MULTIPROGRAMMING AND TIME SHARING

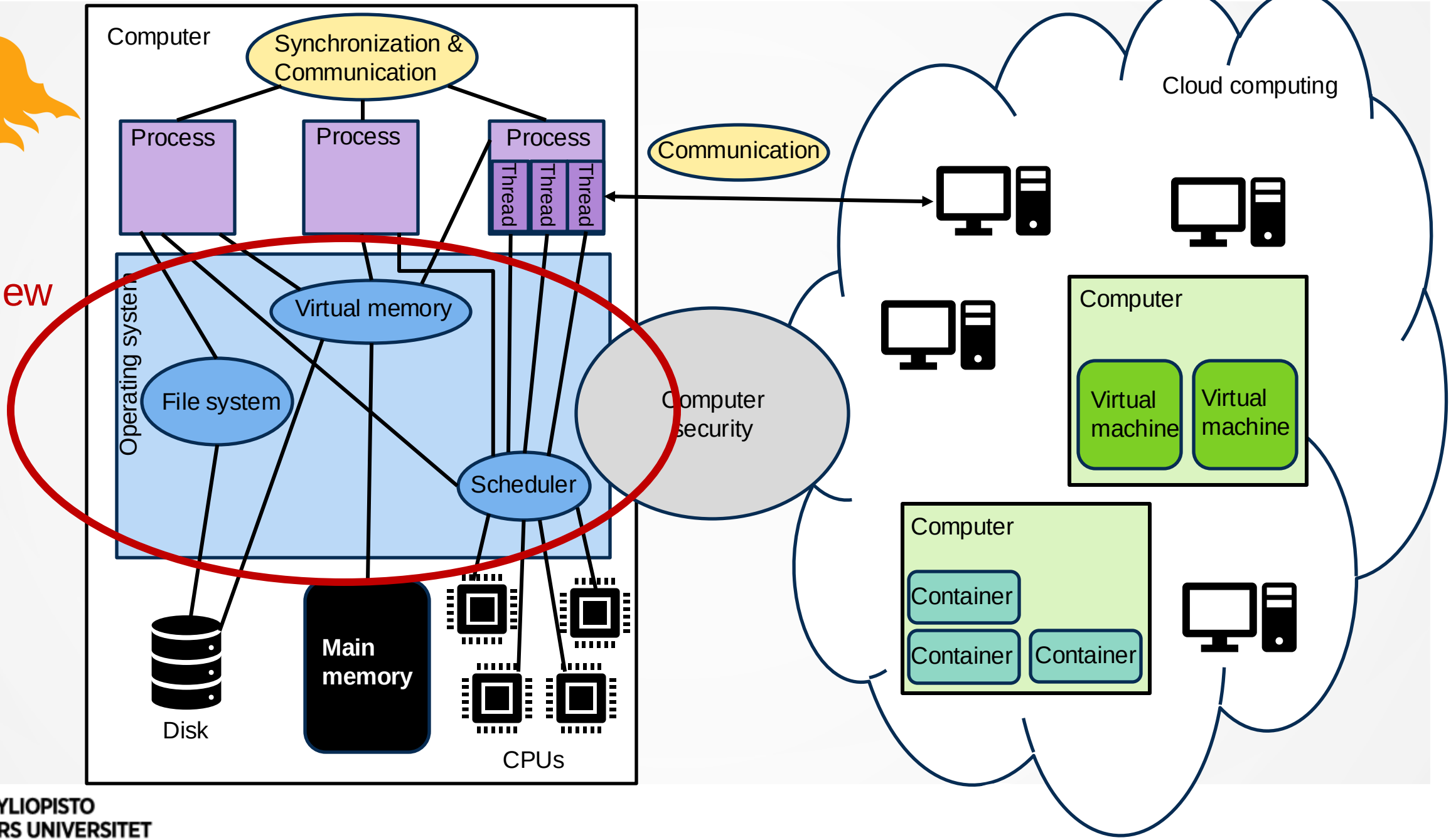
- **Batch multiprogramming:** Multiple processes executed concurrently without the need for user interaction
- **Time sharing:**
  - Multiple users simultaneously accessing a system
  - OS interleaves the execution of each user program in a short burst or **quantum** of computing
  - If there are *n* users, each user will see on average *1/n* of the effective computer capacity (not counting OS overhead)

Table 2.3 Batch Multiprogramming versus Time Sharing

	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

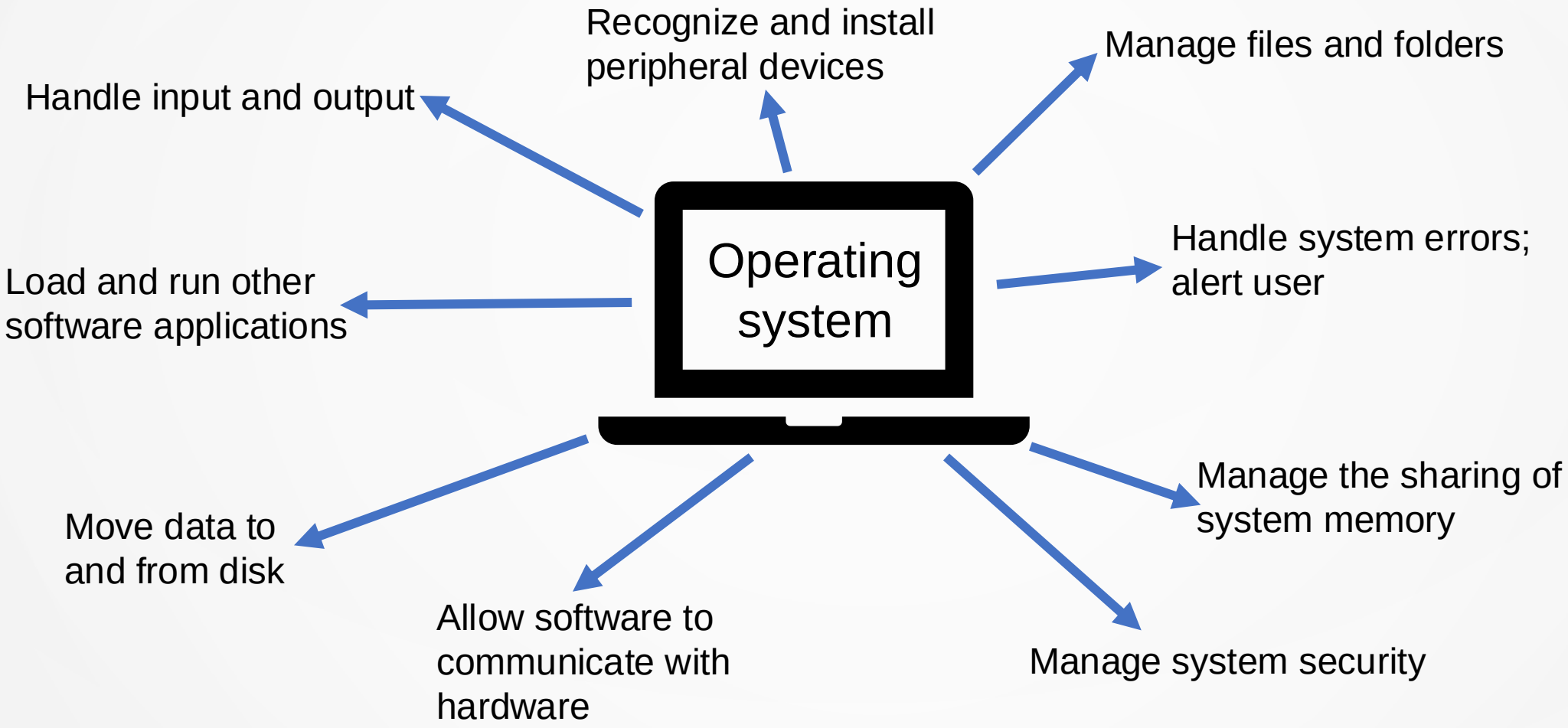
Table from [Stallings, Operating systems: Internals and design principles, 9th ed]

Today:
Overview
of OS



TASKS OF AN OPERATING SYSTEM?

TASKS OF AN OPERATING SYSTEM



OPERATING SYSTEM (OS)

- Software responsible for running programs **efficiently** and **easy-to-use** manner; acts as an interface between hardware and user programs; acts as **resource manager**
- **Three main concepts:**
  - **Virtualization:** take a **physical resource**, e.g. processor, memory or disk, and **transform it into a virtual form**
  - **Concurrency:** handle problems that arise when **many things are done at once** in the same program
  - **Persistence:** hardware and software need to **store data** persistently

VIRTUALIZATION

- **Virtualizing CPU:** from the point of view of a process, it has access to (seemingly its own) virtual CPU, while in reality there is a single (or a small number of) physical CPU(s) which are shared between processes
- **Virtualizing memory:** each process accesses its own virtual address space, which the operating system maps onto the physical memory of the machine
  - Memory reference within a process does not affect the address space of other processes and OS itself
- OS provides user processes an illusion that each of them is running on their own computer

CONCURRENCY

- How to **synchronize** so that **shared resources** are not messed up?
- How to avoid **race conditions, deadlock, starvation**? How to ensure **safety** and **liveness**?
- What about (partial/total) **correctness**? Concurrency bugs are tricky to find, consequences can be serious, e.g.
  - Therac-25 (race condition) - three deaths
  - North East blackout 2003 (race condition) - affected 55 million people
- How to partition code so that everything runs **efficiently**?
- Problems of concurrency not limited to OS – **multi-threaded** programs exhibit the same problems!

PERSISTENCE

- System memory is **volatile –** when power goes down or the system crashes, any data in memory is lost
  - How to ensure that data is not lost?
- Both hardware and software need to be able to store data **persistently**
- E.g. hard drives and SSDs are the hardware for long-term storage
- **File system** is the software in OS that manages the disk
  - Responsible for storing user's files in a **reliable** and **efficient** way on the disks of the system

GOALS IN DESIGNING AN OPERATING SYSTEM

- Depends on, e.g., the choice of hardware, type of the system (batch, time-shared, single vs. multi-user, distributed, embedded, …)
- ...

GOALS IN DESIGNING AN OPERATING SYSTEM

- Depends on, e.g., the choice of hardware, type of the system (batch, time-shared, single vs. multi-user, distributed, embedded, …)
- **User goals**: convenience of use, easy to learn, reliable, safe, fast, …
- **System goals**: easy to design, implement and maintain, reliable, error-free, efficient, …
  - High degree of reliability necessary (OS must run non-stop, all applications fail if OS fails)
- **No unique solution** for all requirements!

HOW TO ACHIEVE SOME OF THESE GOALS?

- **Abstractions** make the system easy to use
- **Minimizing overheads** maintains high performance
- **Isolation** provides protection (between applications, between OS and applications)

- Concepts:
  - **Policy**: What will be done?
  - **Mechanism**: How to do something?
  - Example: Two programs want to run at the same time. A policy answers the question Which program should run?'. A mechanism is a method or protocol that implements this functionality

OPERATING SYSTEM DESIGN APPROACHES

- **Kernel** vs **operating system**:
  - Kernel is a core component of an OS responsible for managing system **memory**, **scheduling** processes, and **accessing** the system's **hardware** on behalf of the users
  - In addition to kernel, OS can contain e.g. graphical user interface, file system utilities, text editors, compilers, …
- Monolithic vs microkernel:
  - In a **monolithic** kernel all **services** as well as core functionality share the same address space
  - **Microkernel** minimizes the kernel by moving as much functionality as possible from the kernel into user space (external subsystems, user processes)
  - In real life, hard to categorize exactly – many operating systems contain features of both approaches

SUMMARY

- **Computing platform** is a **combination of hardware and software** that provides an environment where applications can be executed
- An **operating system** is **software** responsible for running processes **efficiently** and **easy-to-use** manner; an **interface** between hardware and user programs
- Key concepts:
  - **Multiprogramming, time sharing, multiprocessing, multithreading, distributed processing**
  - **Virtualization, concurrency, persistence**
  - **Policy** and **mechanism**
  - **Kernel**