Sistemi Operativi

Corso di Laurea in Informatica

a.a. 2019-2020

SAPIENZA
UNIVERSITÀ DI ROMA

Gabriele Tolomei

Dipartimento di Informatica

Sapienza Università di Roma

tolomei@di.uniroma1.it

Useful Information

Class schedule

- **Tuesday:** 5 p.m. - 7 p.m.
- **Friday:** 4 p.m. - 7 p.m.

Useful Information

Class schedule

- **Tuesday:** 5 p.m. - 7 p.m.
- **Friday:** 4 p.m. - 7 p.m.

Useful Information

Class schedule

- **Tuesday:** 5 p.m. - 7 p.m.
- **Friday:** 4 p.m. - 7 p.m.

Office hours

- **Thursday:** 10 a.m. - 12 p.m.

Useful Information

Class schedule

- **Tuesday:** 5 p.m. - 7 p.m.
- **Friday:** 4 p.m. - 7 p.m.

Office hours

- **Thursday:** 10 a.m. - 12 p.m.

Exam

written + oral (*optional*)

Useful Information

Class schedule

- **Tuesday:** 5 p.m. - 7 p.m.
- **Friday:** 4 p.m. - 7 p.m.

Office hours

- **Thursday:** 10 a.m. - 12 p.m.

Exam

written + oral (*optional*)

Contacts

- **email:** tolomei@di.uniroma1.it
- **website:** http://www.di.uniroma1.it/~tolomei

Resources

- Class material released on the website of the course [https://github.com/gtolomei/operating-systems]
- Suggested books (though not mandatory!):
  - "*Operating System Concepts*" Ninth Edition – Silberschatz, Galvin, Gagne
  - "*Modern Operating Systems*" Fourth Edition – Tanenbaum, Bos
  - "*Operating Systems: Three Easy Pieces*" – Remzi and Andrea Arpaci-Dusseau [available online]
- Any additional resource available on the Web!

Outline of the Course

- Introduction

Outline of the Course

- Introduction
- Processes and Threads

Outline of the Course

- Introduction
- Processes and Threads
- Memory Management

Outline of the Course

- Introduction
- Processes and Threads
- Memory Management
- File System and I/O

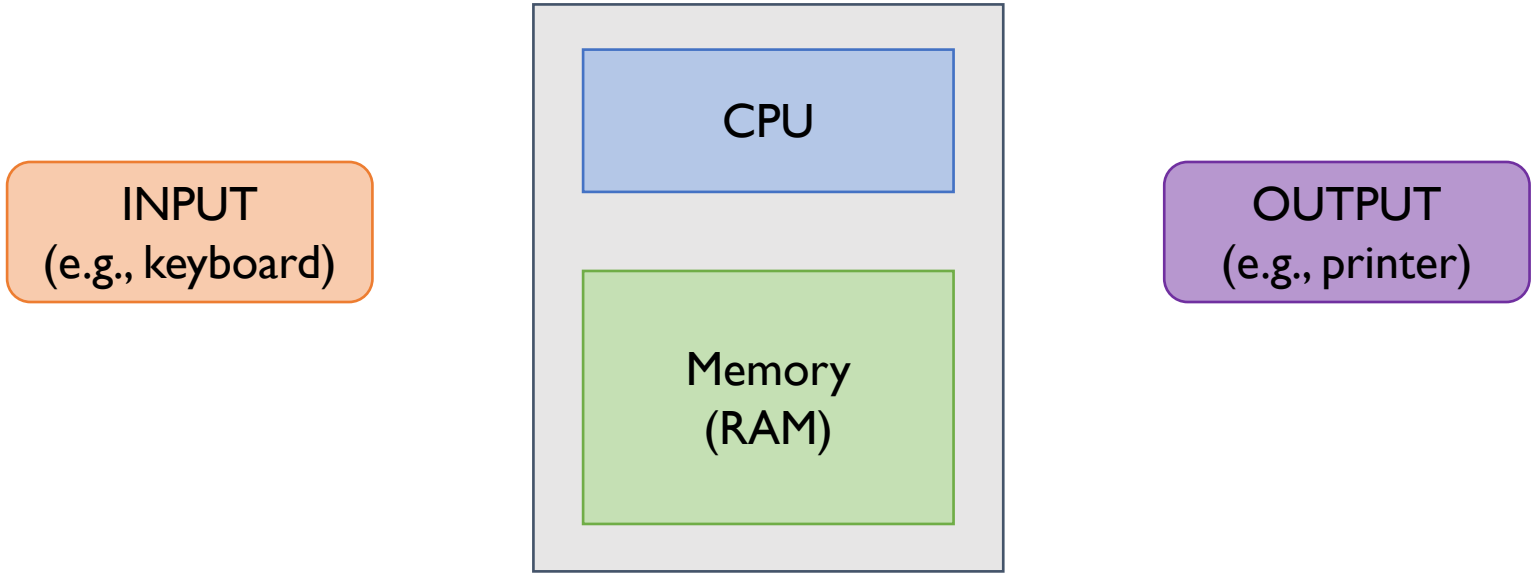
Outline of the Course

- Introduction
- Processes and Threads
- Memory Management
- File System and I/O
- Advanced Topics
  - Protection/Security?
  - Distributed/Mobile Systems?

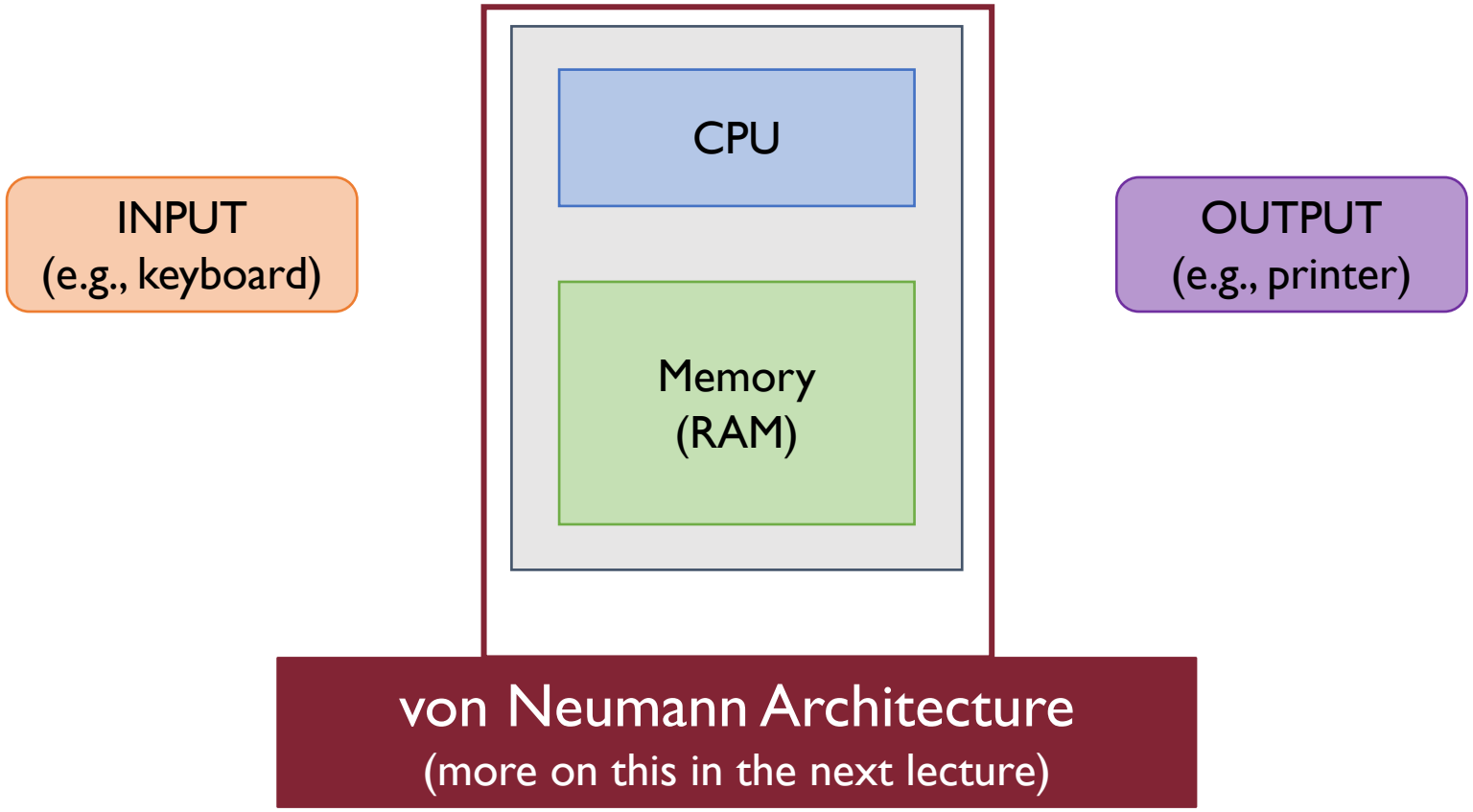
Language and Naming Conventions

- As you may have noticed, slides are all written in english!
- OS → Operating System
- HW → Hardware
- SW → Software
- VM → Virtual Machine
- …
- Other shortcuts/acronyms may appear here and there without notice! Please, ask if anything is not clear!

High-Level View of a Computer System



High-Level View of a Computer System



What is an Operating System?

What is an Operating System?

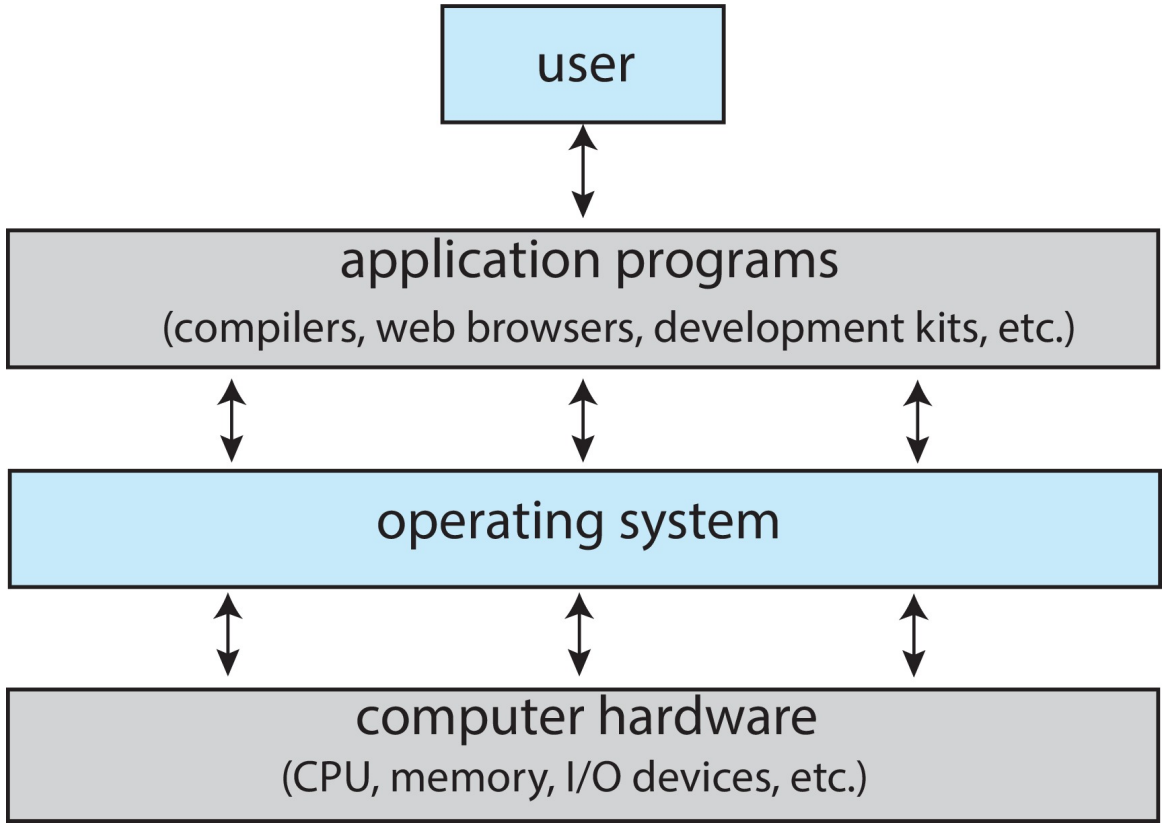
- There exists no universally accepted definition!

What is an Operating System?

- There exists no universally accepted definition!
- However, the following definition is quite appropriate:

> Implementation of a **virtual machine** that is (hopefully) easier to program than bare hardware

Computer System Overview



What is Inside an Operating System?

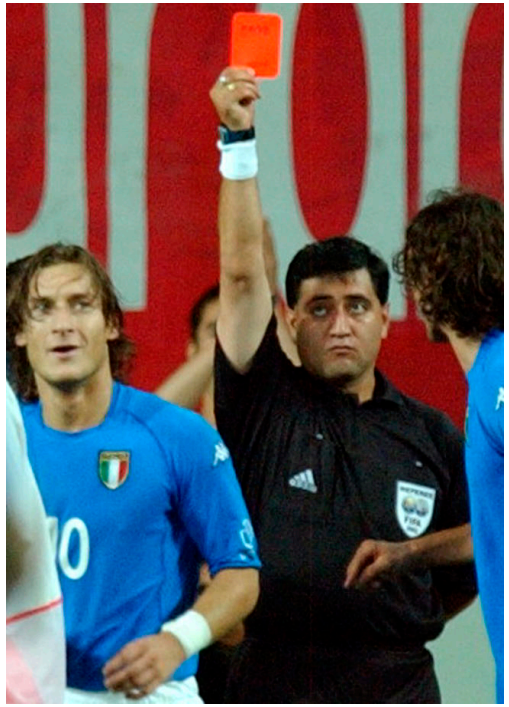
- Again, no single answer to this question!
- It is a **system design** choice to decide what to include in the OS
- Different systems may have different requirements:
  - general-purpose, real-time, mobile, etc.
- Typically, we distinguish between:
  - **kernel** → the "core" of the OS (always up and running)
  - **system programs** → everything else which is still part of the OS

OS Wears Many Hats

- **Referee (Resource Manager)**
  - Manages shared physical resources: CPUs, memory, I/O, etc.
  - To achieve fairness and efficiency

OS Wears Many Hats

- **Referee (Resource Manager)**
  - Manages shared physical resources: CPUs, memory, I/O, etc.
  - To achieve fairness and efficiency

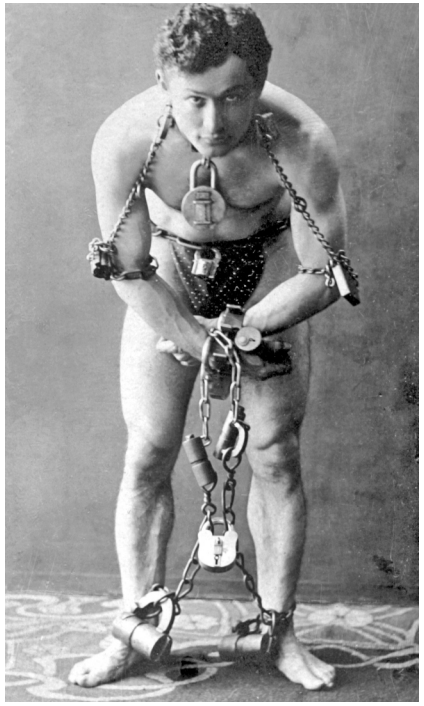


OS Wears Many Hats

- **Referee (Resource Manager)**
  - Manages shared physical resources: CPUs, memory, I/O, etc.
  - To achieve fairness and efficiency

OS Wears Many Hats

- **Illusionist (Virtual Machine)**
  - Virtualize any physical resource
  - To give applications/users the illusion of infinite resources available



OS Wears Many Hats

- **Glue (HW/SW Interface)**
  - Provides a set of common services (APIs) to separate HW from SW
  - To allow applications/users to interact with the system without talking directly to the HW

History of Operating Systems

Phase 1: Expensive HW, Cheap Humans

- **1** machine : **M** users
- Hand-programmed systems
- Single-user console systems (mainframes)
- Batch systems
- Multi-programming systems

1945-55: Vacuum Tubes and Plugboards

- Used by a restricted and skilled group of people
- All programming was done in machine language directly
- Basically, no OS whatsoever!
- **Problem: ?**

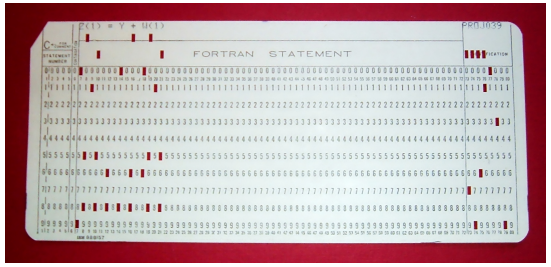
1945-55: Vacuum Tubes and Plugboards

- Used by a restricted and skilled group of people
- All programming was done in machine language directly
- Basically, no OS whatsoever!
- **Problem:** low utilization of expensive HW

Phase 1: Expensive HW, Cheap Humans

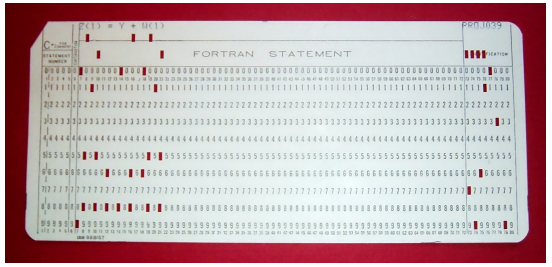
- **1** machine : **M** users
- Hand-programmed systems
- Single-user console systems (mainframes)
- Batch systems
- Multi-programming systems

1955-65: Mainframes



- One user at a time interacting with the machine as program runs
- Programs are written on cards
- Executes one thing at a time: no overlap between computation and I/O
- Primitive OS: program loader
- **Problem: ?**

1955-65: Mainframes



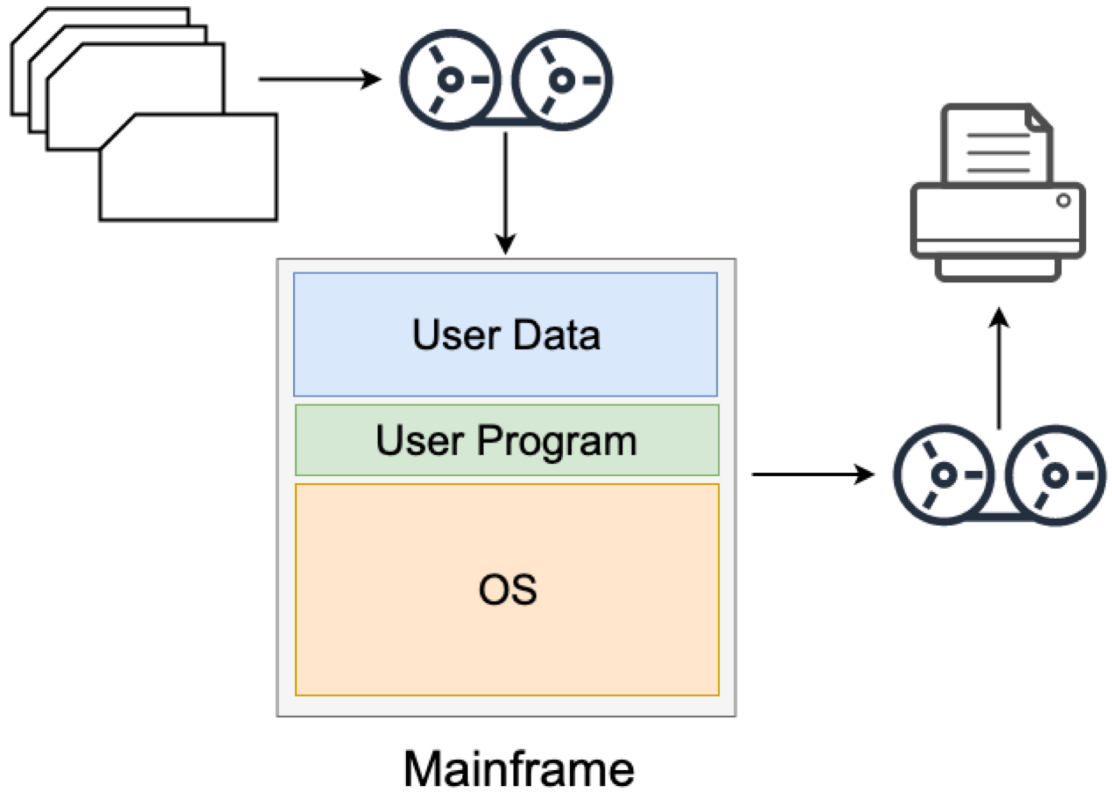
- One user at a time interacting with the machine as program runs
- Programs are written on cards
- Executes one thing at a time: no overlap between computation and I/O
- Primitive OS: program loader
- **Problem:** inefficient for multiple users

Phase 1: Expensive HW, Cheap Humans

- **1** machine : **M** users
- Hand-programmed systems
- Single-user console systems (mainframes)
- Batch systems
- Multi-programming systems

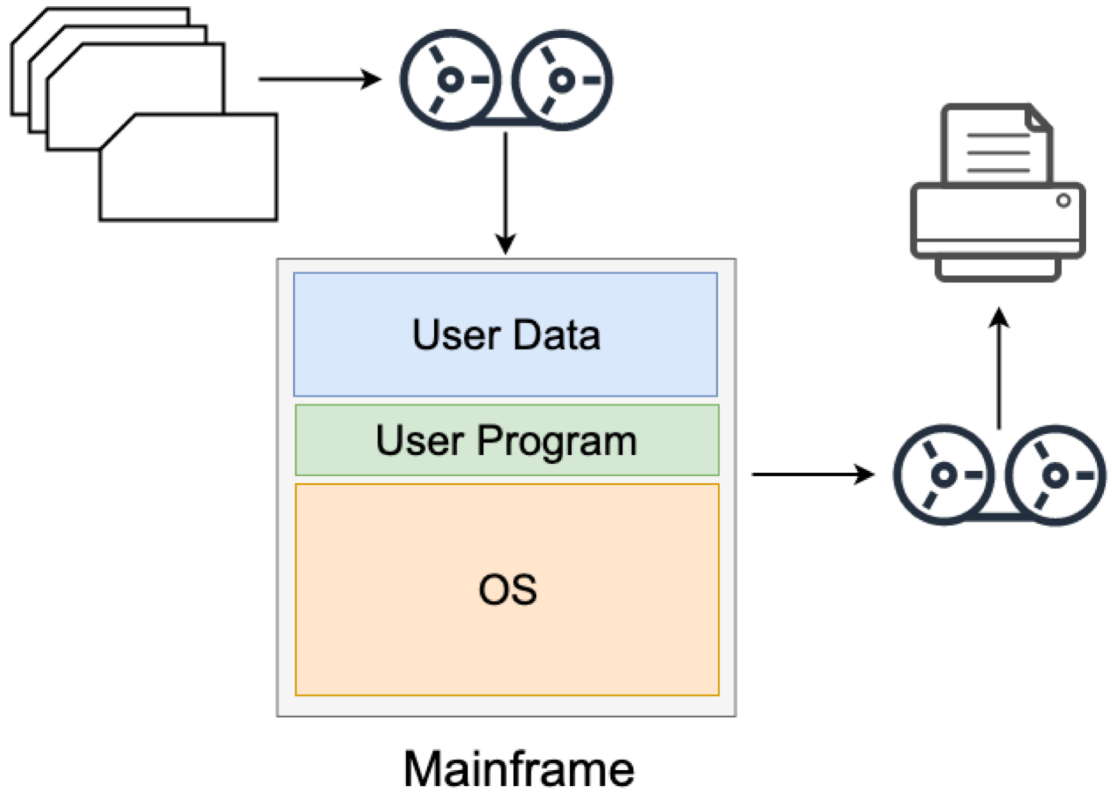
1955-65: Batch Systems

- Execute multiple **jobs** in batch
- Users submit jobs (on cards or tapes)
- Technician still schedules jobs
- OS loads and run jobs
- More efficient use of the machine
- **Problem: ?**



1955-65: Batch Systems

- Execute multiple **jobs** in batch
- Users submit jobs (on cards or tapes)
- Technician still schedules jobs
- OS loads and run jobs
- More efficient use of the machine
- **Problem:** still one job at a time!

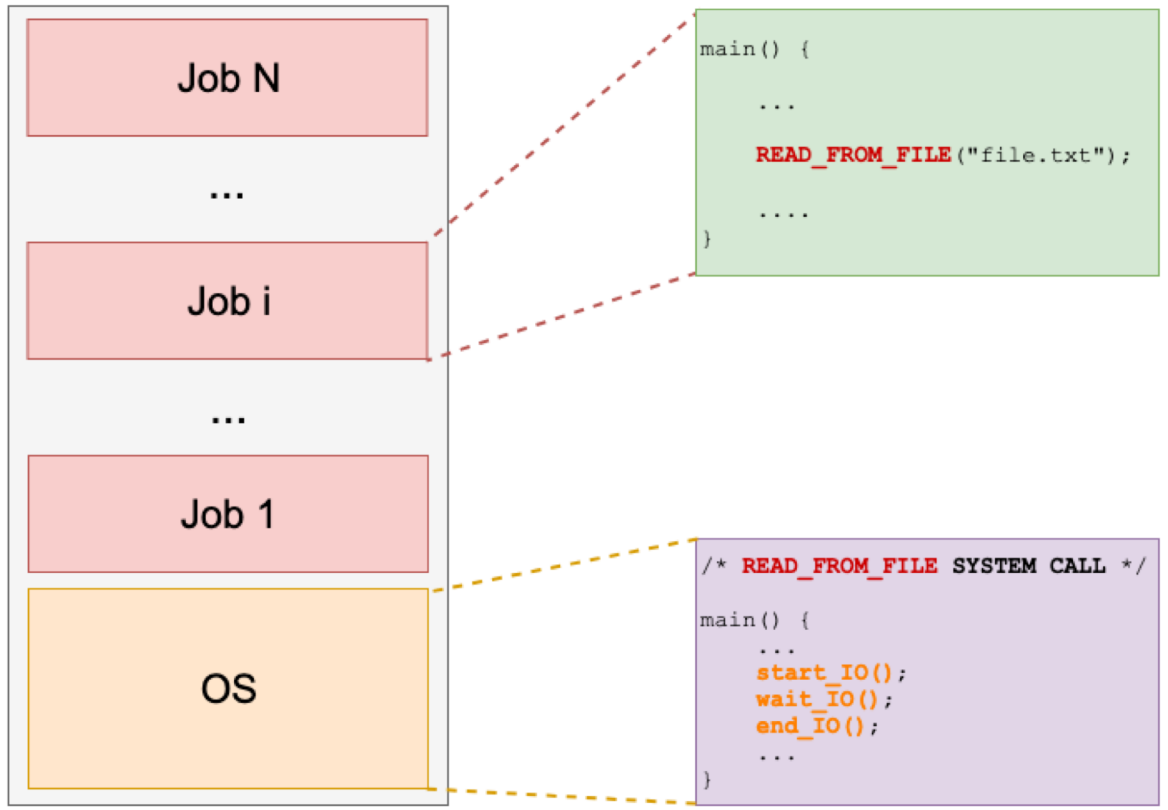


Phase 1: Expensive HW, Cheap Humans

- **1** machine : **M** users
- Hand-programmed systems
- Single-user console systems (mainframes)
- Batch systems
- Multi-programming systems

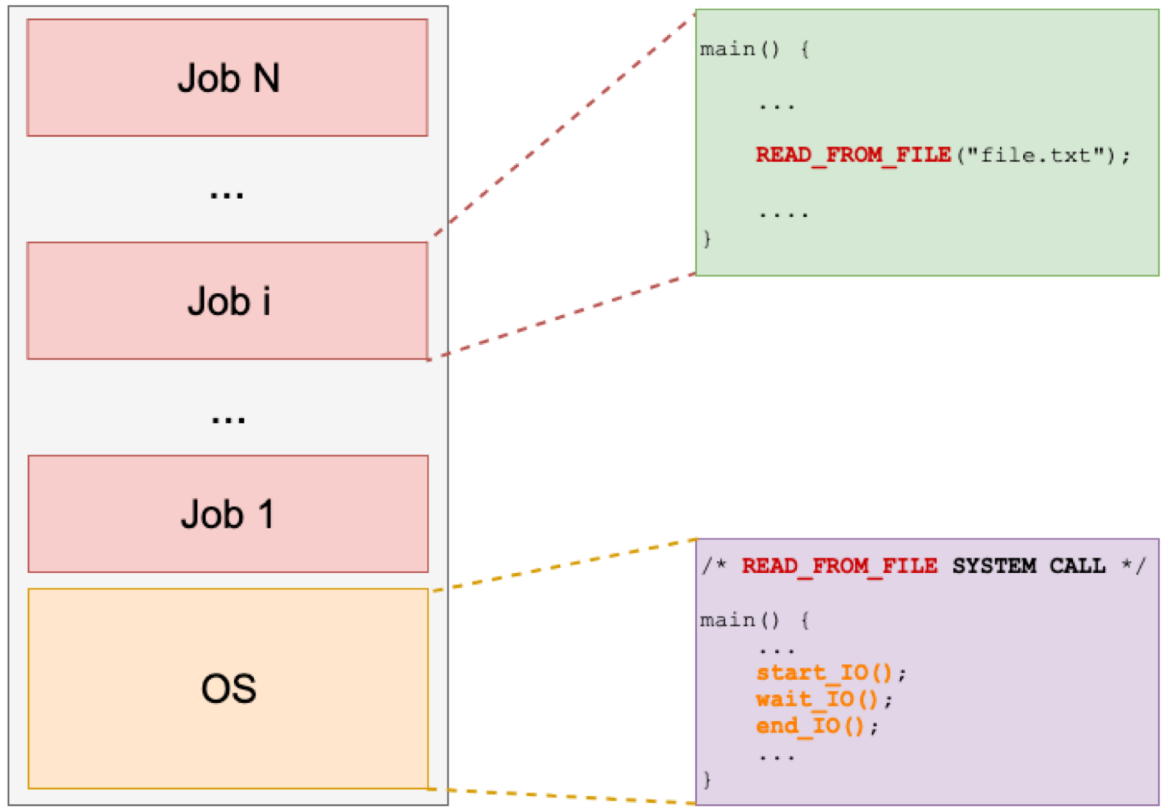
1955-65: Multiprogramming Systems

- Keep several jobs loaded in memory
- Multiplex CPU between jobs
- OS responsibilities:
  - job scheduling
  - memory protection
  - I/O operations
- **Problem: ?**

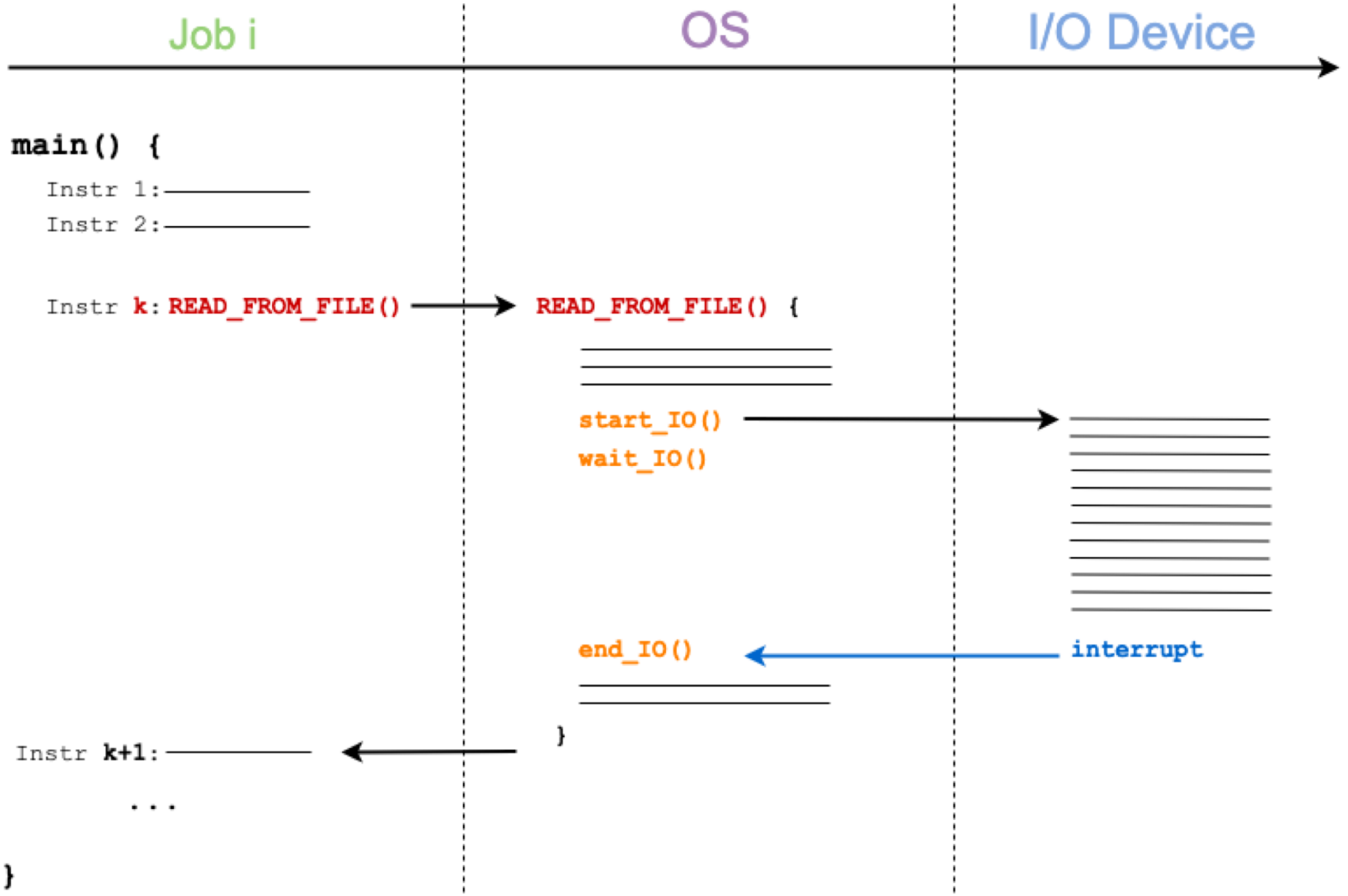


1955-65: Multiprogramming Systems

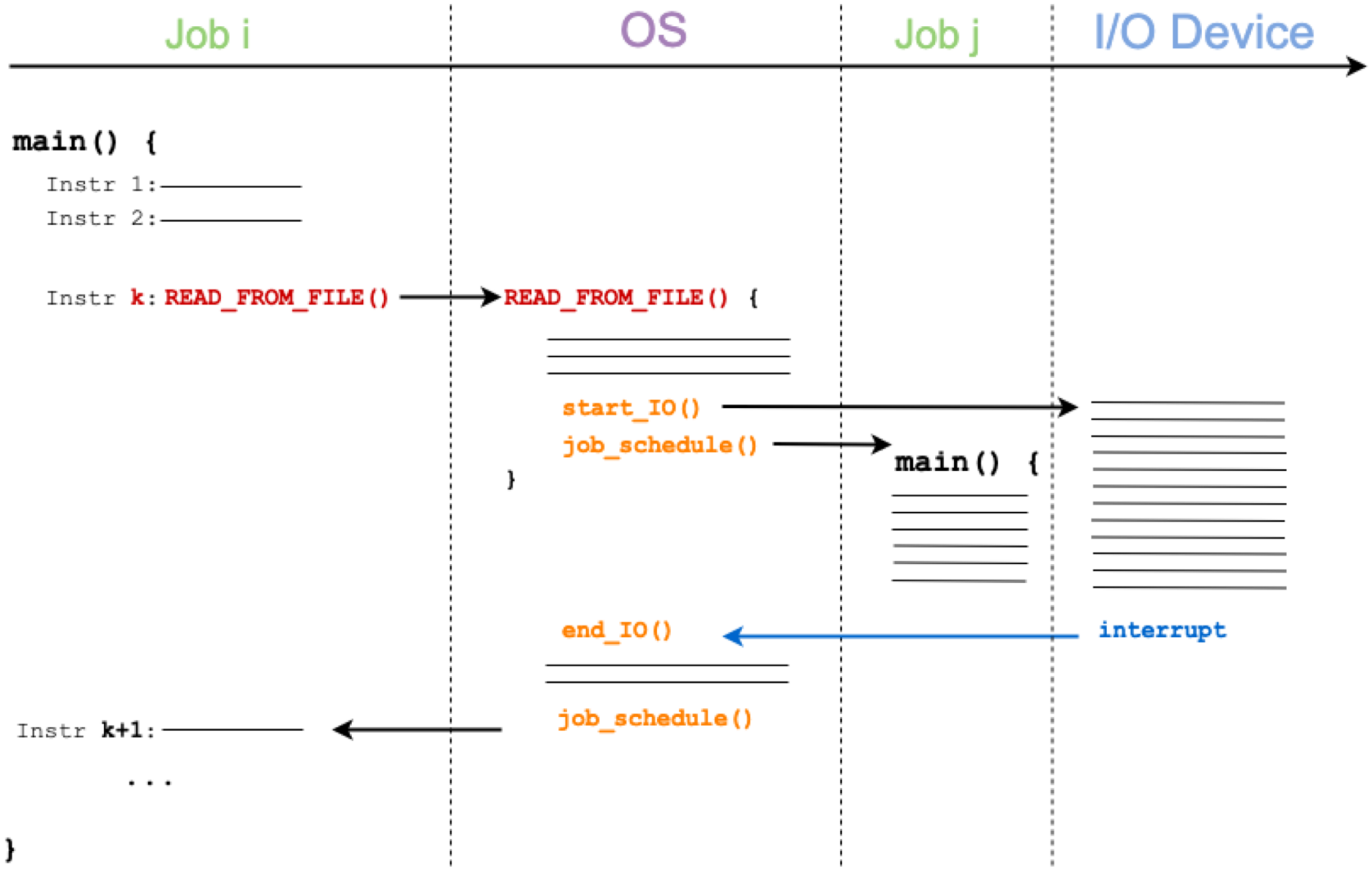
- Keep several jobs loaded in memory
- Multiplex CPU between jobs
- OS responsibilities:
  - job scheduling
  - memory protection
  - I/O operations
- **Problem:** CPU is left idle while blocking I/O operations take place



Blocking I/O


Job i
OS
I/O Device
main() {
Instr 1:
Instr 2:
Instr k: READ_FROM_FILE()
READ_FROM_FILE() {
start_IO()
wait_IO()
end_IO()
interrupt
}
Instr k+1:
...
}

Non-Blocking I/O

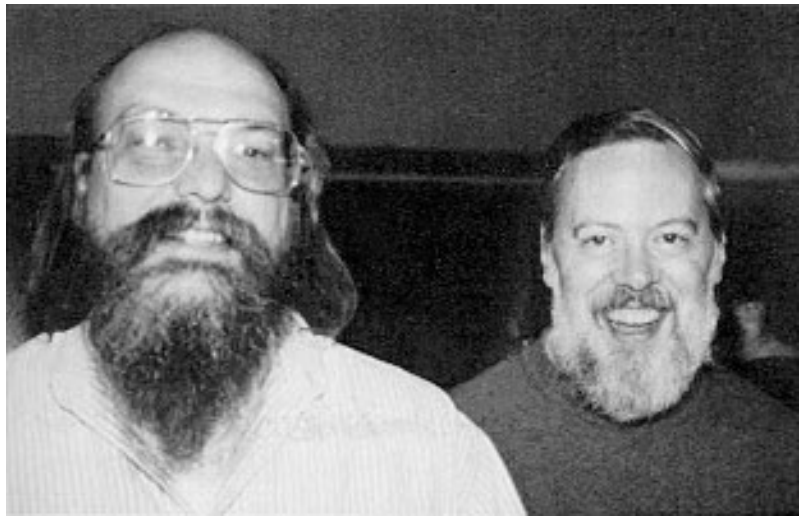


Phase 2: Cheap HW, Expensive Humans

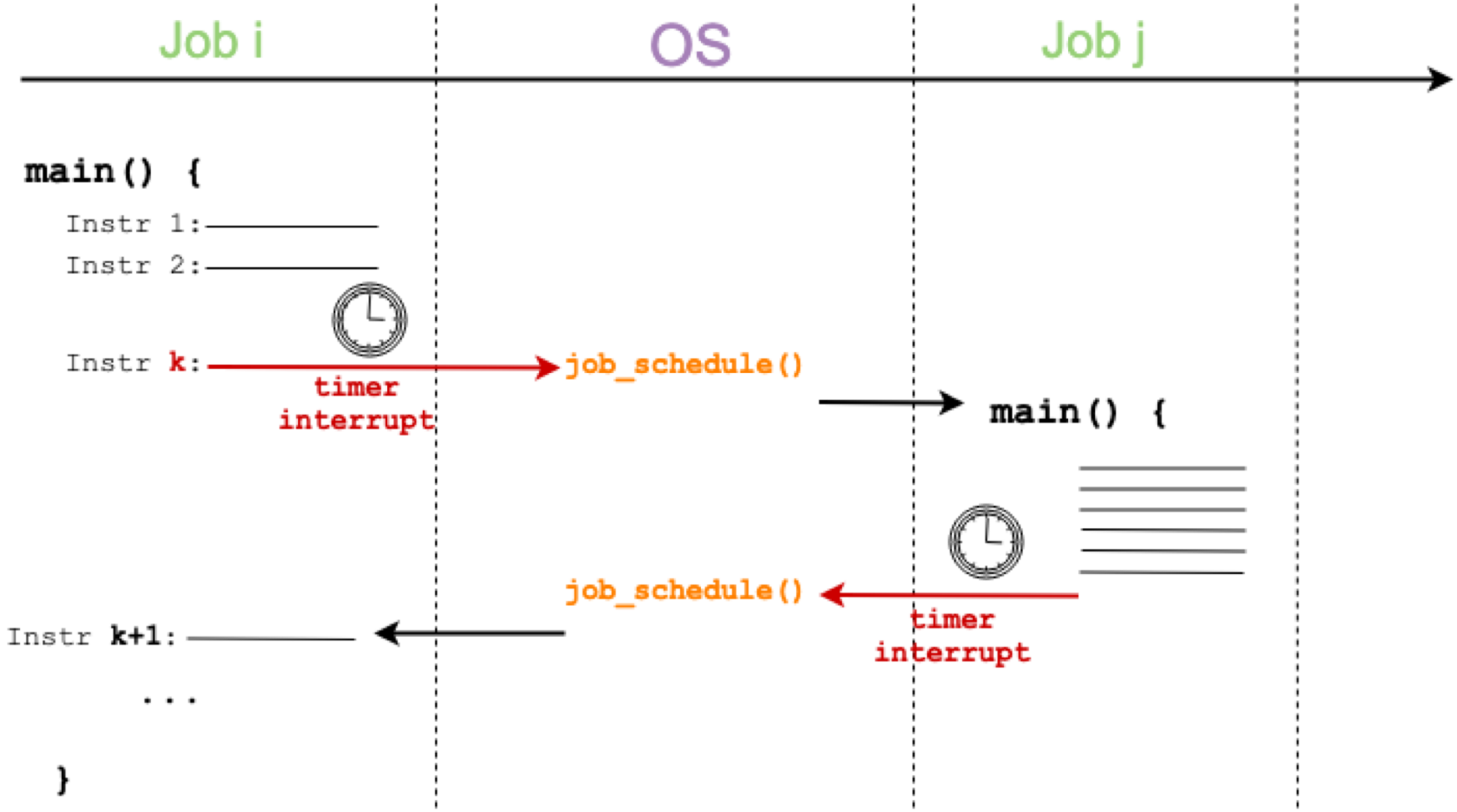
- **1** machine : **M** users (still)
- Time-sharing systems

1970-: Time-sharing

- Many users connected to the same CPU via cheap consoles
- Timer interrupt used to multiplex CPU between jobs
- Illusion of parallelism (pseudo-parallelism)
- Ken Thompson and Dennis Ritchie → **UNIX OS**



Pseudo-parallelism



Phase 3: Very Cheap HW, Very Expensive Humans

- Personal Computing → **1** machine : **1** user

Phase 3: Very Cheap HW, Very Expensive Humans

- Personal Computing → **1** machine : **1** user
- Distributed/Ubiquitous Computing → **M** machines : **1** user

1980's: Personal Computers

- Initially, simple OSs:
  - No multiprogramming, concurrency, memory protection, etc.
- Later on:
  - Networking, file sharing, Graphical User Interfaces (GUIs)
- IBM PCs (1981) and Apple Macintosh (1984)

1990's-: Personal Computers (and more)

- PCs are now equipped with a fully fledged OS:
  - Windows NT (1991)
  - Mac OS X and related (2001)
  - Linux (1991)

1990's-: Personal Computers (and more)

- PCs are now equipped with a fully fledged OS:
  - Windows NT (1991)
  - Mac OS X and related (2001)
  - Linux (1991)
- Several new (computing) environments where an OS is needed:
  - Transportations: airplanes, cars, etc.
  - Telecommunications: smartphones
  - Home appliances: smart TVs
  - …

1990's-: Personal Computers (and more)

- PCs are now equipped with a fully fledged OS:
  - Windows NT (1991)
  - Mac OS X and related (2001)
  - Linux (1991)
- Several new (computing) environments where an OS is needed:
  - Transportations: airplanes, cars, etc.
  - Telecommunications: smartphones
  - Home appliances: smart TVs
  - …
- Plus, the Web has made everything distributed!

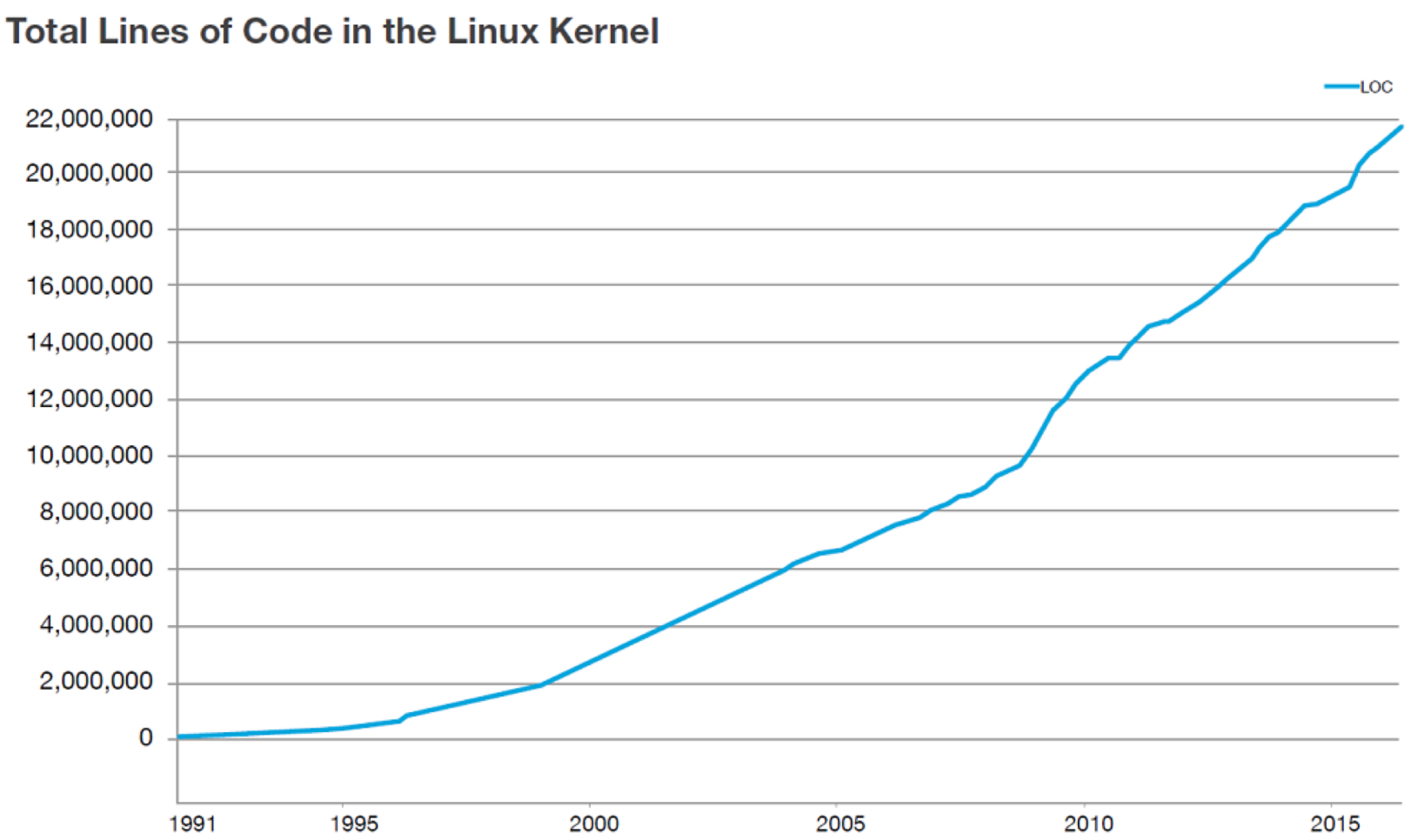
New Trends in OS Design

- Active field of research
  - OS demand is growing (many computing devices are available)
  - New application settings (Web, Cloud, mobile, cars, etc.)
  - Hardware is rapidly changing (new CPUs coming out)

New Trends in OS Design

- Active field of research
  - OS demand is growing (many computing devices are available)
  - New application settings (Web, Cloud, mobile, cars, etc.)
  - Hardware is rapidly changing (new CPUs coming out)
- Open-source OS (Linux)
  - Allows developers to contribute to OS development
  - Excellent research platform to experiment with

Linux Kernel Size



Why Study OSs?

- To learn important concepts of computer science
  - **Abstraction**
    - Virtualize any physical resource (CPUs, memory, I/O, etc.)

Why Study OSs?

- To learn important concepts of computer science
  - Abstraction
    - Virtualize any physical resource (CPUs, memory, I/O, etc.)
  - **Systems Design Tradeoffs**
    - Performance vs. Cost of OS abstractions
    - Performance vs. Complexity of OS design
    - HW vs. SW implementation of key features

Why Study OSs?

- To learn important concepts of computer science
  - Abstraction
    - Virtualize any physical resource (CPUs, memory, I/O, etc.)
  - Systems Design Tradeoffs
    - Performance vs. Cost of OS abstractions
    - Performance vs. Complexity of OS design
    - HW vs. SW implementation of key features
  - **How computers work**

Large Computer Systems

- The world is increasingly dependent on computer systems
  - Large, complex, interconnected, distributed, etc.
- Huge demand for experts who deeply understand and can build such systems, which need to be:
  - Reliable, effective, efficient, secure, etc.

Large Computer Systems

- The world is increasingly dependent on computer systems
  - Large, complex, interconnected, distributed, etc.
- Huge demand for experts who deeply understand and can build such systems, which need to be:
  - Reliable, effective, efficient, secure, etc.

OS is a great example of a large computer system

OS as Large Computer System

- Designing large computer systems requires you to know
  - **Each computer:**
    - Architectural details
    - High-level programming language (mostly, C/C++)
    - Memory management
    - Concurrency and scheduling
    - File system and I/O

OS as Large Computer System

- Designing large computer systems requires you to know
  - Each computer:
    - Architectural details
    - High-level programming language (mostly, C/C++)
    - Memory management
    - Concurrency and scheduling
    - File system and I/O
  - **Across clusters of computers:**
    - Server architectures
    - Distributed file systems and computing frameworks

OS Design Issues (1)

- **Structure** → How the whole system is organized
- **Concurrency** → How parallel tasks are managed
- **Sharing** → How resources are shared
- **Naming** → How resources are identified by users
- **Protection** → How critical tasks are protected from each other
- **Security** → How to authenticate, authorize, and ensure privacy
- **Performance** → How to make it more efficient (quick, compact)

OS Design Issues (2)

- **Reliability** → How to deal with failures
- **Portability** → How to write once and run anywhere
- **Extensibility** → How to add new features/capabilities
- **Communication** → How to exchange information
- **Scalability** → How to scale up as demand increases
- **Persistency** → How to save task's status
- **Accounting** → How to claim on control resource usage

Architectural Trends: CPU

*Million Instructions Per Second

**1 MHz = 1,000,000 clock cycles per second

	1971 (Intel 4004)	Today (Intel Core i9)	Δ (orders of magnitude)
MIPS*	~0.09	~300,000+	+7
Instructions (fetch, decode, execute) per clock cycle	~0.12	~100+	+3
Clock frequency (MHz)**	0.74	~5,000	+4
Cheap size (μm)	10	0.014	-3

Architectural Trends: CPU

*Million Instructions Per Second

**1 MHz = 1,000,000 clock cycles per second

	1971 (Intel 4004)	Today (Intel Core i9)	Δ (orders of magnitude)
MIPS*	~0.09	~300,000+	+7
Instructions (fetch, decode, execute) per clock cycle	~0.12	~100+	+3
Clock frequency (MHz)**	0.74	~5,000	+4
Cheap size (μm)	10	0.014	-3

Moore's law: the number of transistors in a dense integrated circuit doubles about every two years

Architectural Trends: Main Memory

	1973 (DEC PDP-8)	Today (Samsung DDR4)	Δ (orders of magnitude)
Capacity (kB)	12	128,000,000	+7
Cost ($/MB)	~400,000	~0.005	-8

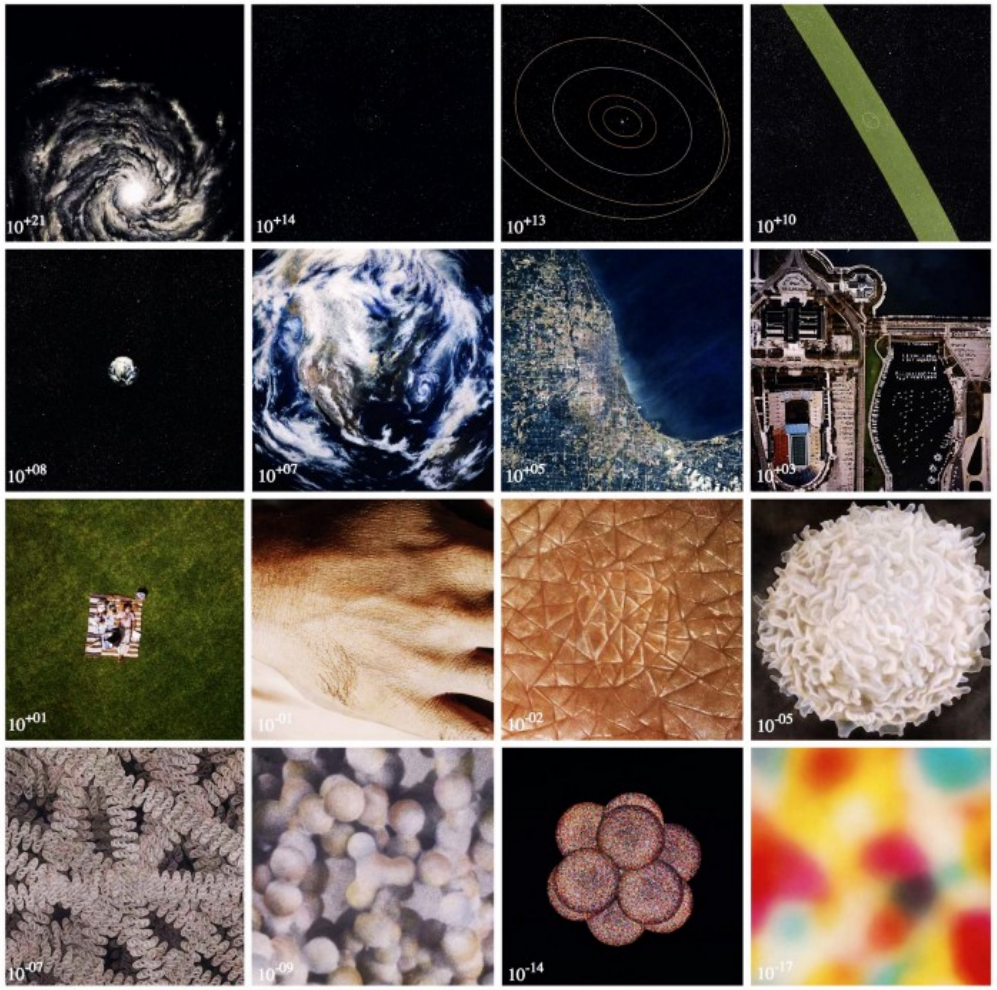
Architectural Trends: Disk

	1956 (IBM RAMAC 305)	Today (Western Digital)	Δ (orders of magnitude)
Capacity (MB)	5	15,000,000	+7
Size (inch)	24 (x50)	3.5	-3
Cost ($/MB)	640 (per month)	~0.000018	-9

Architectural Trends: Orders of Magnitude

$10^0 = 1$

Architectural Trends: Orders of Magnitude



What's Next?

- Moore's law has hit its limit(?)
  - chip size has physical constraints
  - power vs. heat tradeoff
  - alternatives have already pushed forward the end of it:
    - multicore-manycore processors
  - other approaches are subject of research:
    - molecular/DNA transistors
    - quantum computing

Summary

- Basic roles of an Operating System

Summary

- Basic roles of an Operating System
- A brief history of Operating Systems

Summary

- Basic roles of an Operating System
- A brief history of Operating Systems
- Operating Systems as large and complex computer systems

Summary

- Basic roles of an Operating System
- A brief history of Operating Systems
- Operating Systems as large and complex computer systems
- New architectural trends open up novel opportunities and challenges in Operating System design