Operating Systems

Course: OS, Second Semester 2017
Lecturer: Konstantin Markov
E-mail: markov@u-aizu.ac.jp

TA: Vazhenina Daria
E-mail: d8132102@u-aizu.ac.jp

Course URL: http://hi-mdle.u-aizu.ac.jp

Course Schedule

Lectures:

Wednesday, Period 1, M8

Exercises:

Wednesday, Periods 2 and 3, std6

Dates:

Lec.1–10/04	Cancel–11/8	Lec.9 – 12/13	Lec.12–01/17
Lec.2–10/11	Lec.6 –11/15	Lec.10 – 12/20	Lec.13–01/24
Lec.3–10/18	Lec.7 –11/22	W.Vac. – 12/27	Lec.14–01/31
Lec.4–10/25	**M.Ex.–12/04**	W.Vac. – 01/03	**F.Ex. –02/XX**
Lec.5–11/01	Lec.8 –12/06	Lec.11 – 01/10	

Grading Policy

Your final grade includes the following parts:

1. All Lab Exercises: **40 points**

2. Midterm Exam: **25 points**

3. Final Exam: **25 points**

4. Attendance: **10 points**

Course Management

1. Using **Moodle** system.
2. Need an account.
3. Exercises downloaded from **Moodle.**
4. Answers uploaded to **Moodle.**
5. Scores, comments – from/to **Moodle.**

Operating Systems

INTRODUCTION

Lecture Topics

- OS concepts
- Computer-system components
- Evolutional steps
- OS components
- OS services

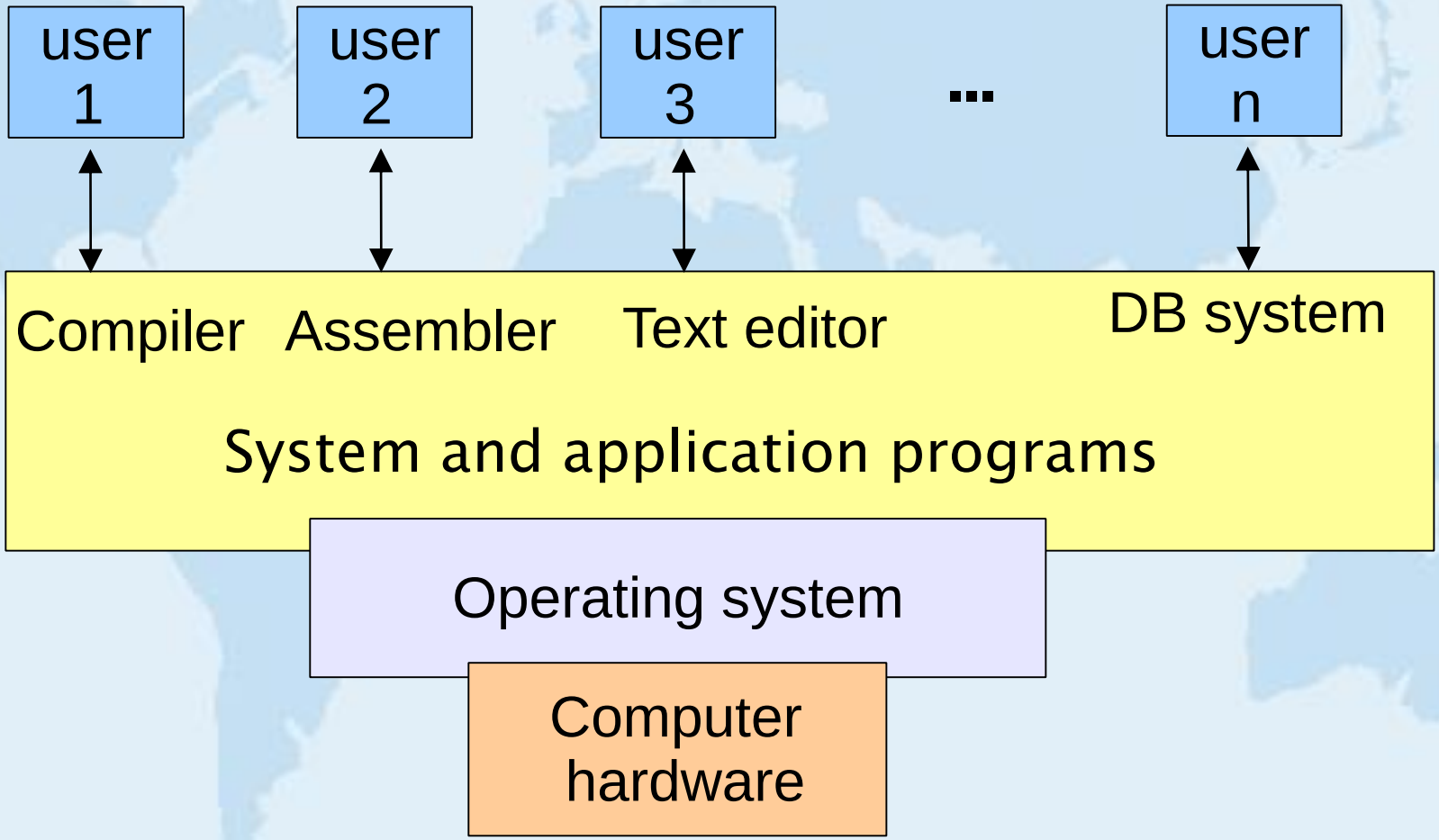
What is an OS?

- A program that **manages** the computer hardware.
- It acts as an **intermediary** between the computer user and the computer hardware.
- It provides an **environment** for the execution of programs.
- OS main goals:
  - to make the computer **convenient** to use.
  - to use the hardware in an **efficient** manner.

What is an OS?

- The OS is a **resource allocator**.
  - It acts as the manager of the computer resources and allocates them to specific programs and users. OS must decide to which requests resources should be allocated, so the computer system will work efficiently and fairly.
- The OS is a **control program.**
  - It controls the execution of user programs to prevent errors and improper use of the computer.
- The OS is the program running at **all times** on the computer.

Abstract View of the Computer System Components



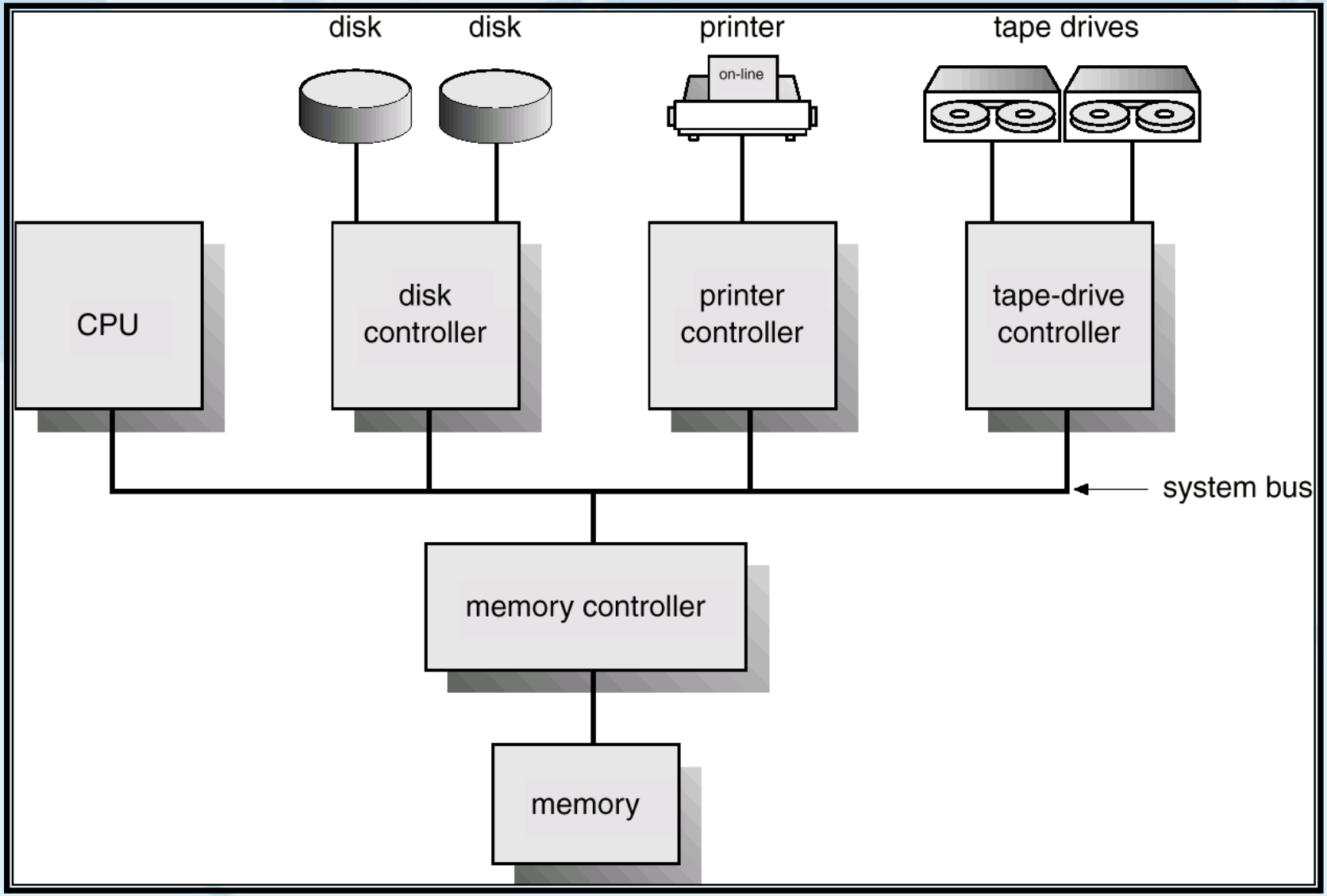
Early Systems

- They were **large and expensive** machines run from a console.
- The programmer, who was also the operator of the computer, would write a program, and then would operate the program directly from the console.
- Each job consisted of many separate steps involving the loading and unloading of **magnetic tapes, paper tapes,** and **punch cards**.
- While tapes were being mounted or the programmer was operating the console, the CPU sat **idle**.

Systems Evolution

- Replacing the very slow card readers, line printers, etc. with magnetic-type units. The readers and printers were operated **off-line**.
- Using **multiple** reader-to-tape and tape-to-printer systems for one CPU.
- Replacing tapes (**sequential-access devices**) with disks (random-access devices).
- Introducing the simultaneous peripheral operation on-line (**spooling**).

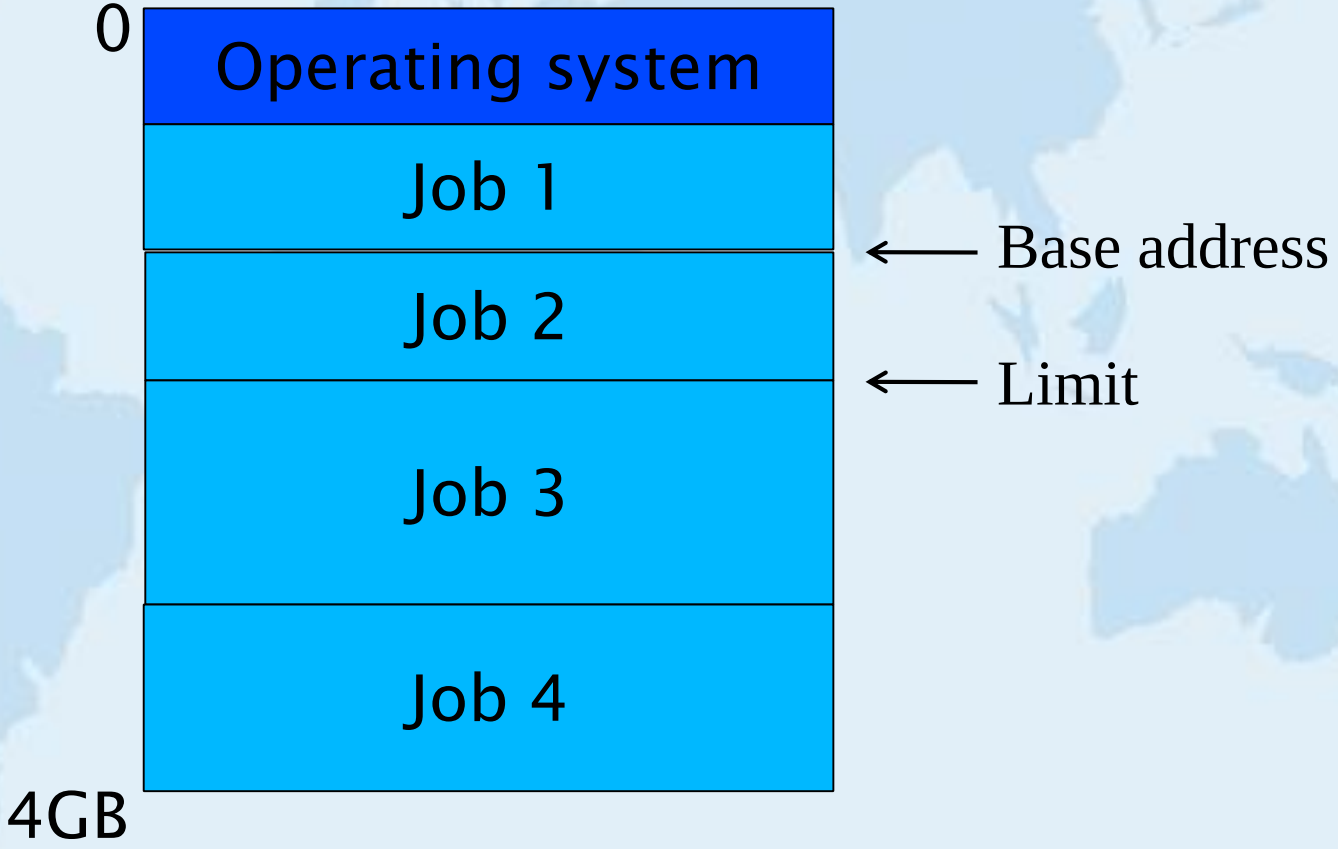
Computer System



Multiprogramming

- OS **increases** the CPU utilization by organizing jobs so that the CPU always has something to execute.
  - It keeps **several** jobs in memory.
  - It picks and begins to execute **one** of the jobs.
- OS switches to and executes **another job** when the current job needs to wait. So, when a program does I/O, the CPU works on another program.

Memory protection in a multiprogramming system



Other aspects of multiprogramming

- All the jobs are kept in the **job pool**.
- This pool consists of all processes residing on disk awaiting **allocation** of main memory.
- If several jobs are ready to be brought into memory, and there is not enough room for all of them, then OS must **choose** among them. This decision is ***job scheduling***.
- Having several programs in memory requires ***memory management*** and ***CPU scheduling***.

OS Components

- Process management
- Main-memory management
- File system management
- Mass-storage management
- I/O management
- Networking
- Protection system
- Command – interpreter system

Process Management

An operating system manages **processes, memory, files**, and **networking**. A process can be thought of as a program in execution. It needs certain resources. OS is responsible for the following activities in connection with process management:

- The *creation* and *deletion* of both user and system processes.
- The *suspension* and *resumption* of processes.
- The provision of mechanisms for process **synchronization, communication** and **deadlock handling.**

Main-memory management

Main memory is generally the only storage device that the CPU is able to address **directly**. OS is responsible for the following activities in connection with memory management:

- **Keep track** of which parts of memory are currently being used and by whom.
- **Decide** which processes are to be loaded into memory when memory space becomes available.
- **Allocate and free** memory space as needed.

File system management

Computer can store information on several different types of physical media. OS provides a uniform logical view of information storage and defines a logical storage unit, the **file**. OS is responsible for the following activities:

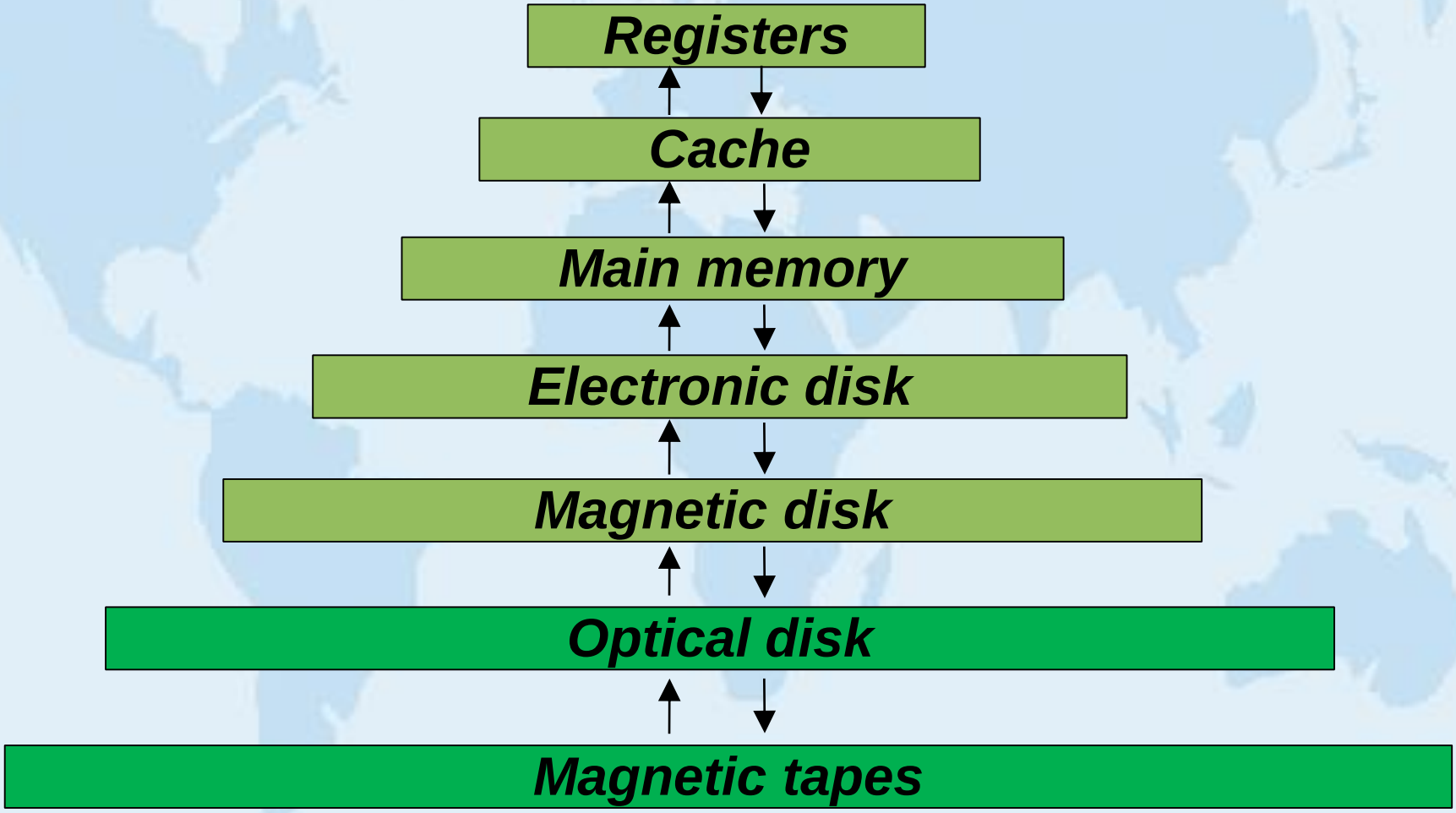
- The **creation** and **deletion** of files and directories.
- The support of primitives for **manipulating** files and directories.
- The **mapping** of files onto secondary storage.
- The **backup** of files on stable (non-volatile) storage media.

Mass-storage management

Because main memory is too small to accommodate all data and programs, and its data are lost when power is lost, the computer system must provide secondary storage (disk) to back up main memory. OS is responsible for the following activities in connection with disk management:

- Free-space management.
- Storage allocation.
- Disk scheduling.

Storage device hierarchy



I/O management

OS should **hide** the peculiarities of specific hardware devices from the user. For example, in UNIX, the peculiarities of I/O devices are hidden from the OS itself by the I/O subsystem, i.e. the set of device drivers. The I/O subsystem consists of:

- A memory-management component that includes **buffering, caching**, and **spooling.**
- A general **device driver** interface.
- Drivers for specific hardware **devices.**

Networking

A **distributed system** collects physically separate, possibly heterogeneous systems into a single coherent system, providing the user with access to the various resources that the system maintains.

- The processors in the system are connected through a **communication network**.
- There are two types of distributed systems: **LANs** (local-area networks) and **WANs** (wide-area networks).
- OS generalizes network access as a form of file access, with the details of networking being contained in the network interface's device driver.

OS services

OS provides **services** to programs and users. It supports:

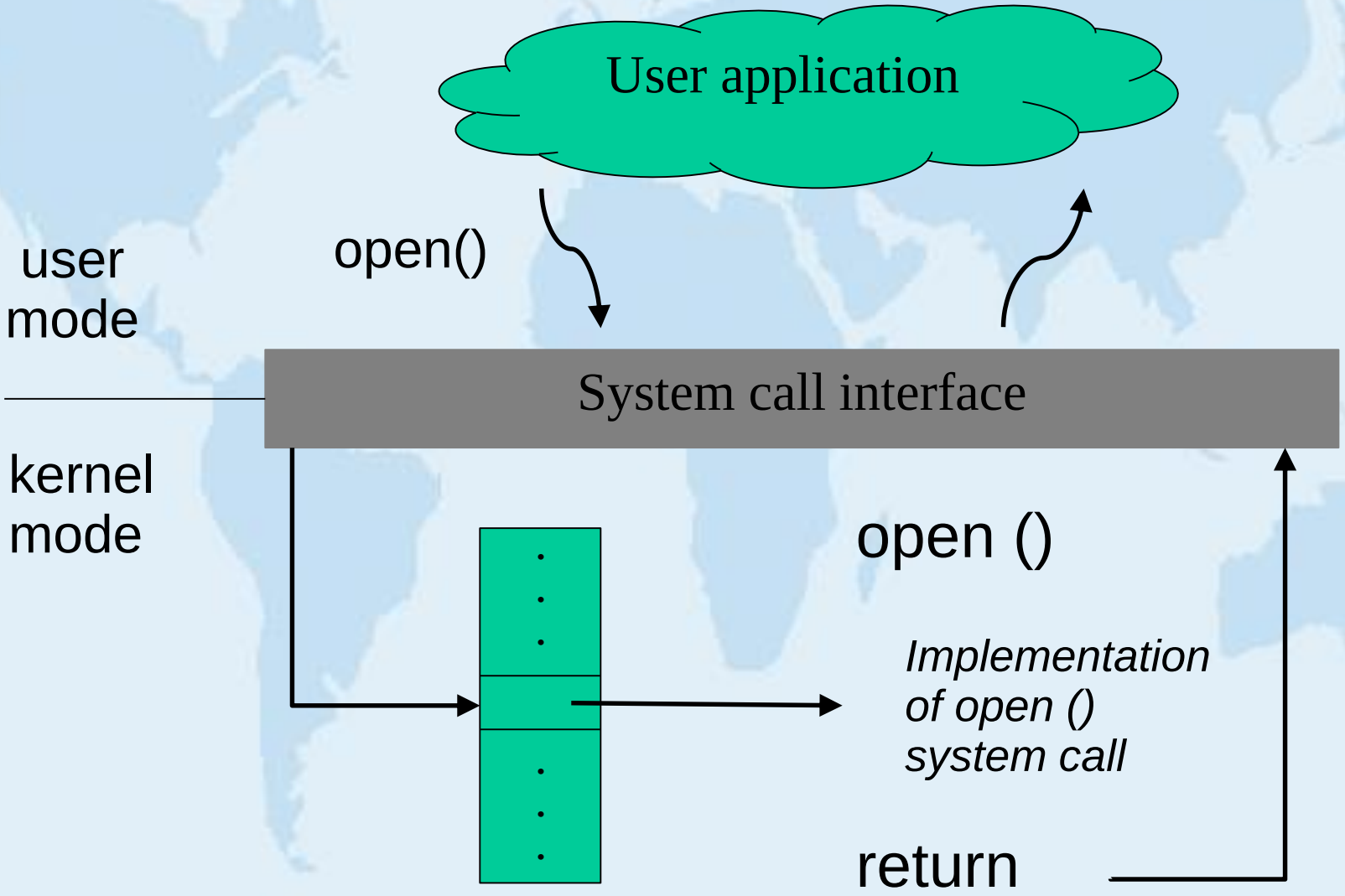
- Program execution and I/O operations.
- File-system manipulation.
- Communications.
- Error detection.
- Resource allocation.
- Accounting and Protection.

System calls

System calls provide the **interface** between the running program and the OS. They support:

- Process control.
- File manipulation.
- Device manipulation.
- Information maintenance.
- Communications.

Use of a system call to perform I/O



System programs

System programs provide a **more convenient environment** for program development and execution. They support:

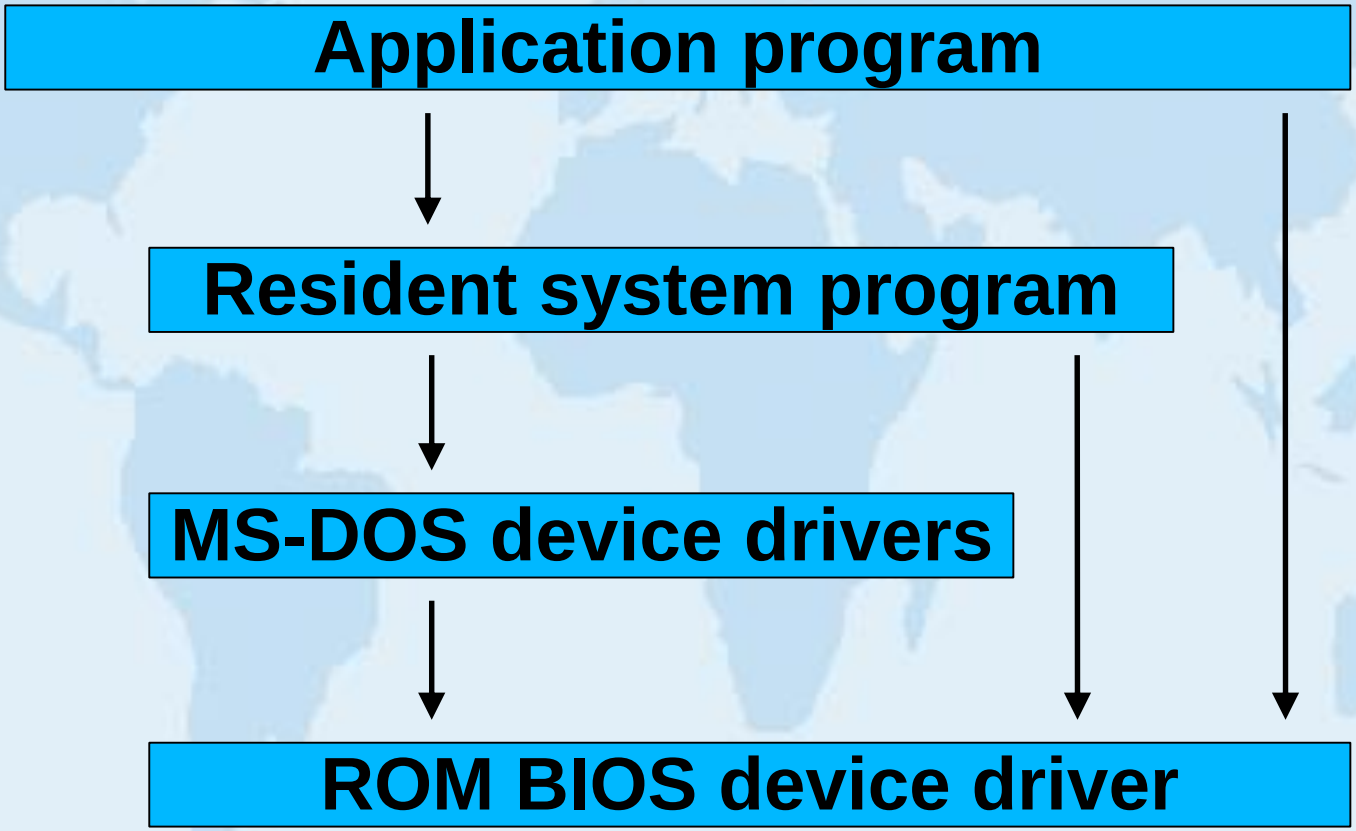
- File manipulation and modification (**cp, rm**).
- Status information (**stat**).
- Programming-language support (**gcc, g++**).
- Program loading and execution (**shell**).
- Communications (**ssh, ftp**).
- Application programs / utilities (**vi, emacs**).

System structure

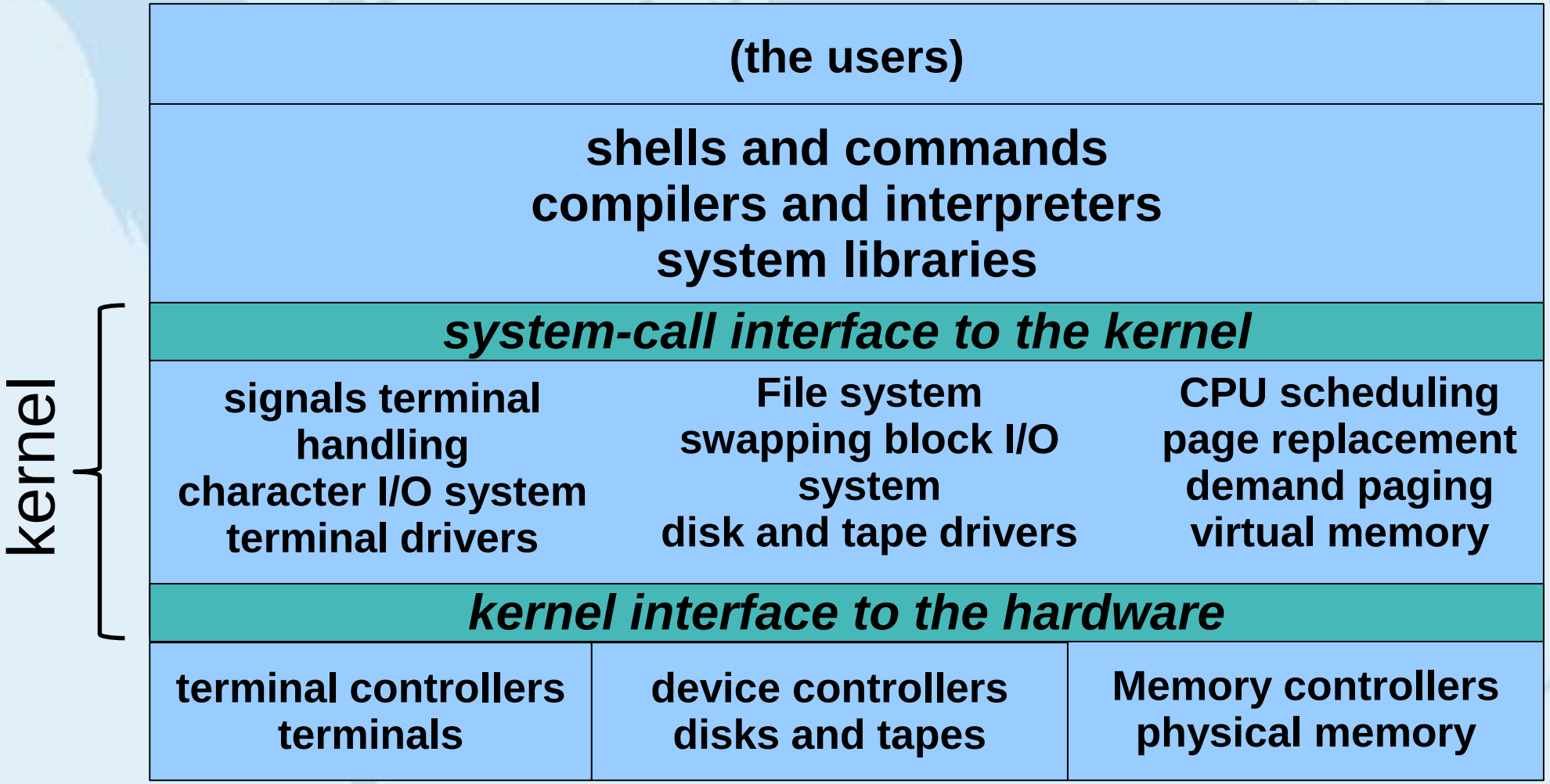
OS are complex and require layered solution. The main advantage of the layered approach is modularity. The layers are selected such that each uses functions and services of only lower-level layers.

- **MS-DOS** has simple structure designed to provide the most functionality in the least space.
- **UNIX** has limited structure with users, shells, kernel, controllers, hardware.
- **VENUS** has more structure with users, drivers, virtual memory, I/O channels, CPU scheduling, instruction interpreter, hardware.

MS-DOS layer structure



UNIX system structure



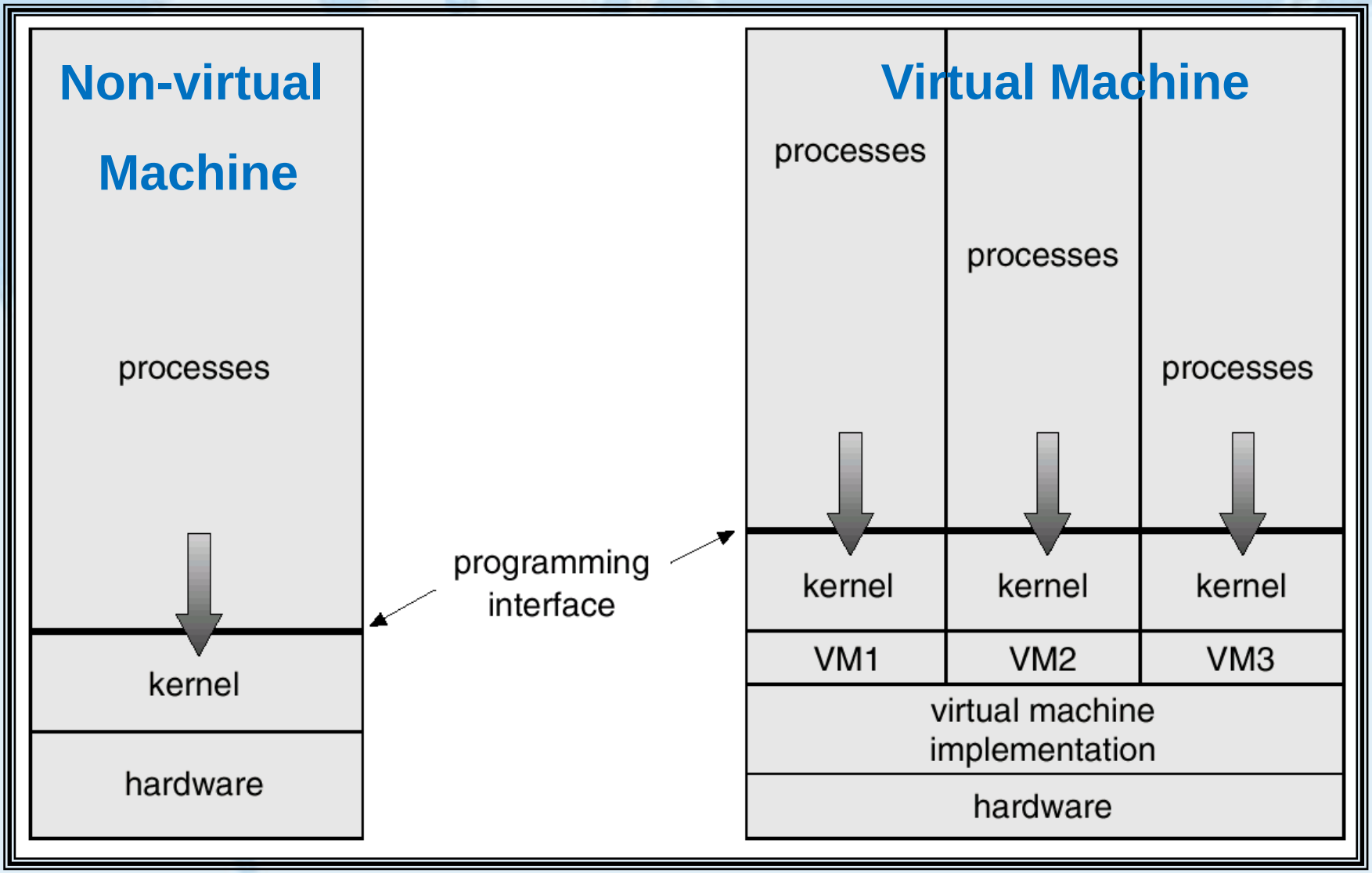
Virtual Machines

- A **virtual machine** takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware.
- A virtual machine provides an interface **identical** to the underlying bare hardware.
- The operating system creates the **illusion** of multiple processes, each executing on its own processor with its own (virtual) memory.

Virtual Machines

- The resources of the physical computer are **shared** to create the virtual machines.
  - CPU scheduling can create the appearance that users have their own processor.
  - Spooling and a file system can provide virtual card readers and virtual line printers.
  - A normal user time-sharing terminal serves as the virtual machine operator's console.

System Models



Virtual Machines

- The virtual-machine concept provides **complete protection** of system resources since each virtual machine is isolated from all other virtual machines. This isolation, however, permits no direct sharing of resources.
- A virtual-machine system is a perfect vehicle for operating-systems **research and development**. System development is done on the virtual machine, instead of on a physical machine and so does not disrupt normal system operation.
- The virtual machine concept is difficult to implement due to the effort required to provide an **exact duplicate** to the underlying machine.

Java Virtual Machine

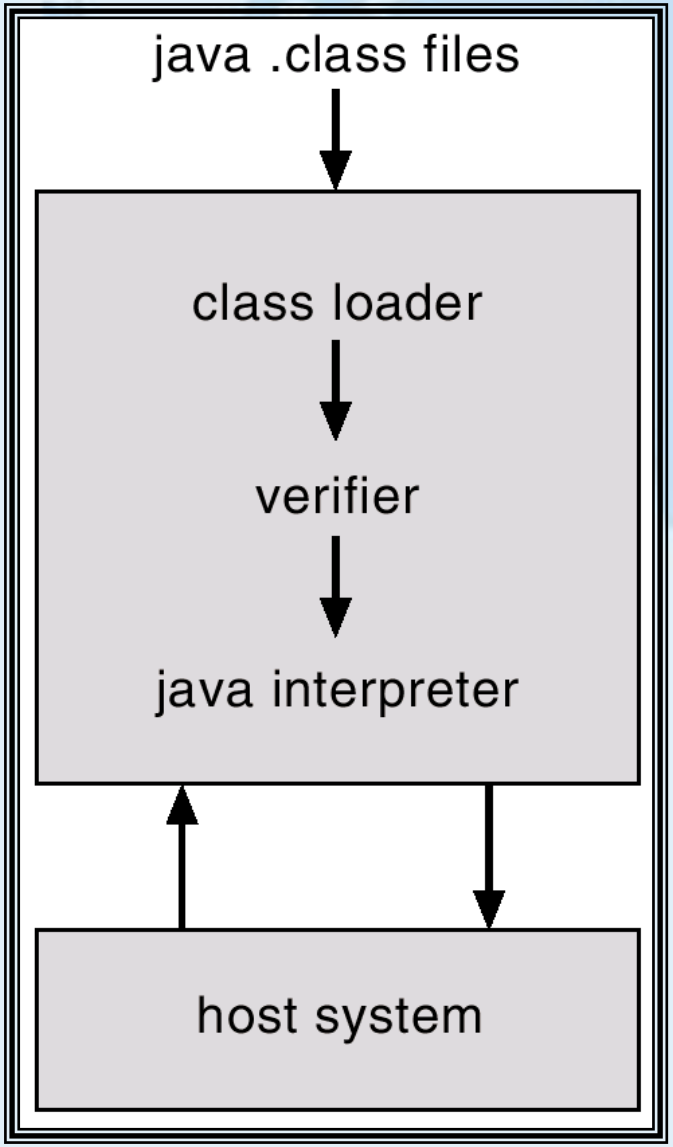
- Compiled Java programs are platform-neutral byte-codes executed by the Java Virtual Machine (**JVM**).
- JVM consists of

  - class loader.

  - class verifier.

  - runtime interpreter.
- Just-In-Time (**JIT**) compilers increase performance.

Java Virtual Machine



OS design goals

- User goals – operating system should be **convenient** to use, **easy** to learn, **reliable, safe**, and **fast**.
- System goals – operating system should be **easy** to design, implement, and maintain, as well as **flexible**, **reliable**, **error-free**, and **efficient**.

System Implementation

- Traditionally written in **assembly** language, operating systems can now be written in **higher-level** languages.
- Code written in a **high-level** language:
  - can be written faster.
  - is more compact.
  - is easier to understand and debug.
- An operating system is far easier to **port** (move to some other hardware) if it is written in a high-level language.


That is all for now!