# OS
Structures

Operating System Structure

(Multiprogramming & Multitasking)

- Operating Systems vary greatly in their makeup internally
- COMMONALITIES:
  - (i) Multiprogramming
  - (ii) Time Sharing (Multitasking)

(i) Multiprogramming

➤ A single user cannot, in general, keep either the CPU or the I/O devices busy at all times

➤ Multiprogramming increases CPU utilization by organizing jobs (code and data) so that the CPU always has one to execute.

Job Pool
Job 1 Job 2
Job 3 Job 4 Job 5
Job 6 Job 7

0
Operating System
Job 1
Job 2
Job 3
Job 4

512 M
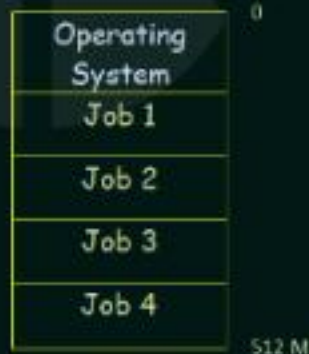
Memory layout for a multiprogramming system

Multiprogrammed systems provide an environment in which the various system resources (for example, CPU, memory, and peripheral devices) are utilized effectively, but they do not provide for user interaction with the Computer system.

(ii) Time Sharing (Multitasking)

- CPU executes multiple jobs by switching among them
- Switches occur so frequently that the users can interact with each program while it is running
- Time sharing requires an interactive (or hands-on) computer system, which provides direct communication between the user and the system.
- A time-shared operating system allows many users to share the computer simultaneously.

- Uses CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared computer.
- Each user has at least one separate program in memory
- A program loaded into memory and executing is called a "PROCESS"

Operating System Services

- An OS provides an environment for the execution of programs
- It provides certain services to programs and to users of those programs
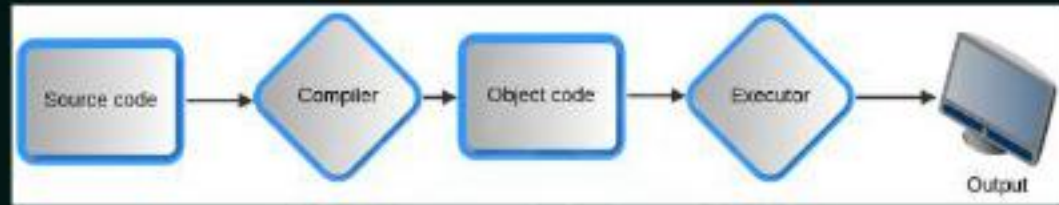
1) User Interface
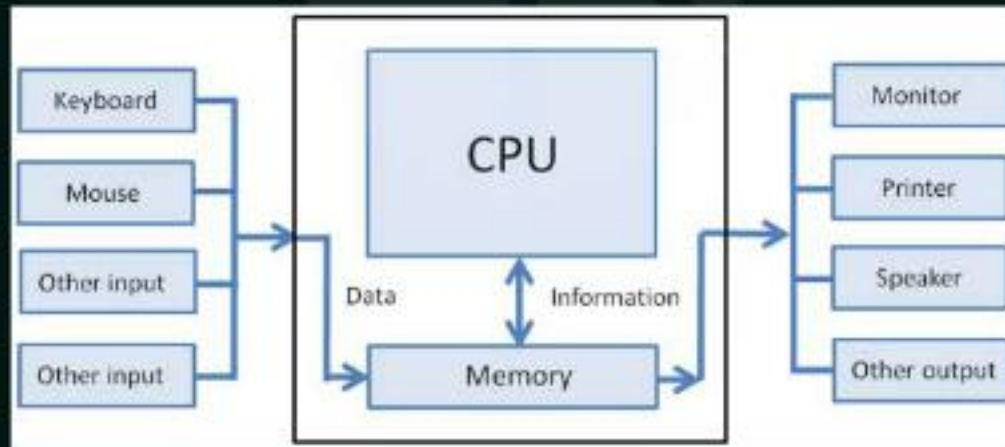

Command Line Interface (CLI)

Graphical User Interface (GUI)

2) Program Execution

Source code → Compiler → Object code → Executor → Output

3) I/O Operations

Keyboard, Mouse, Other input, Other input → Data → CPU ↔ Information ↔ Memory → Monitor, Printer, Speaker, Other output

4) File System Manipulation

6) Error detection

Error

5) Communications

7) Resource Allocation

R_1 R_2 R_3

P_1 P_2 P_3

R_4

8) Accounting

9) Protection and Security

Operating System Services

1. User Interface
2. Program Execution
3. I/O Operations
4. File System manipulation
5. Communications
6. Error Detection
7. Resource Allocation
8. Accounting
9. Protection and Security

User Operating System Interface

There are two fundamental approaches for users to interface with the operating system:

1) Provide a **Command-Line Interface (CLI) or Command Interpreter** that allows users to directly enter commands that are to be performed by the operating system.

2) Allows the user to interface with the operating system via a **Graphical User Interface or GUI.**

Command Interpreter

- Some operating systems include the command interpreter in the kernel.
- Others, such as Windows XP and UNIX, treat the command interpreter as a special program.
- On systems with multiple command interpreters to choose from, the interpreters are known as shells.
- E.g.
  - Bourne shell,
  - C shell
  - Bourne-Again shell (BASH)
  - Korn shell, etc.

System Calls

System calls provide an interface to the services made available by an Operating System.
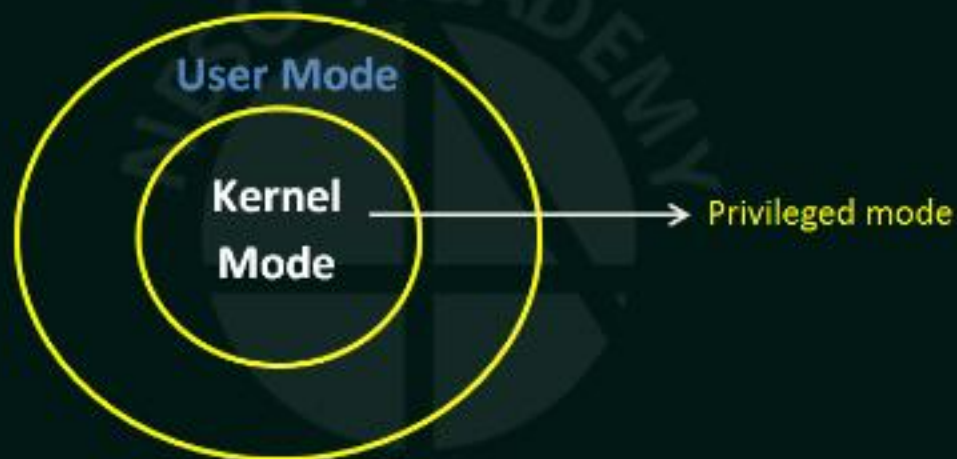
- System call is the programmatic way in which a computer program requests a service from the kernel of the operating system.
- These calls are generally available as routines written in C and C++.

Example of a System Call sequence for writing a simple program to read data from one file and copy them to another file.

Source File → Destination File

Acquire Input Filename

Write prompt to screen

Accept Input

Acquire Output Filename

Write prompt to screen

Accept Input

Open Input File

If File doesn't exist, ABORT

Create Output File

If File exists, ABORT

Loop

Read from Input File

Write to Output File

Until Read fails

Close Output File

Write completion message to screen

Terminate normally

Types of System Calls

System calls can be grouped roughly into five major categories:

1. Process Control
2. File Manipulation
3. Device Management
4. Information Maintenance
5. Communications

1. Process Control

- end, abort
- load, execute
- create process, terminate process
- get process attributes, set process attributes
- wait for time
- wait event, signal event
- allocate and free memory

Process Control

2. File Manipulation

- create file, delete file
- open, close
- read, write, reposition
- get file attributes, set file attributes

File Management

3. Device Manipulation

- request device, release device
- read, write, reposition
- get device attributes, set device attributes
- logically attach or detach devices

Device Management

4. Information Maintenance

- get time or date, set time or date
- get system data, set system data
- get process, file, or device attributes
- set process, file, or device attributes

20.3.2018
11:30 am

5. Communications

- create, delete communication connection
- send, receive messages
- transfer status information
- attach or detach remote devices

System Programs

An important aspect of a modern system is the collection of system programs.

- System programs provide a convenient environment for program development and execution.
- Some of them are simply user interfaces to system calls.
- Others are considerably more complex.

System Programs can be divided into the following categories:

File Management

- Create
- Delete
- Copy
- Rename
- Print
- Dump
- List, and generally manipulate files and directories.

Status Information

Ask the system for:
- Date, Time
- Amount of available memory or disk space
- Number of users
- Detailed performance
- Logging, and debugging information etc.

File modification

- Several text editors may be available to create and modify the content of files stored on disk or other storage devices.
- There may also be special commands to search contents of files or perform transformations of the text.

Programming-language support

- Compilers
- Assemblers
- Debuggers and
- Interpreters

for common programming languages
(such as C, C++, Java, Visual Basic, and PERL)
are often provided to the user with the operating system.

Program loading and execution

Once a program is assembled or compiled, it must be
loaded into memory to be executed.
The system may provide:
- Absolute loaders
- Relocatable loaders
- Linkage editors and
- Overlay loaders

Debugging systems for either higher-level languages or machine language are
needed as well.

Communications

These programs provide the mechanism for:
- Creating virtual connections among processes,
  users, and computer systems.
- Allowing users to send messages to one another's screens
- To browse webpages
- To send electronic-mail messages
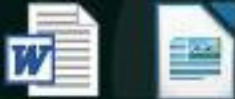- To log in remotely or to transfer files from one machine to another.

In addition to systems programs, most operating systems are supplied with programs that are useful in solving common problems or performing common operations.

Examples:

Web Browsers

Word Processors

Spreadsheets

Database Systems

Games

etc.

Application Programs

Categories of System Programs:

- ☑ File Management
- ☑ Status Information
- ☑ File Modification
- ☑ Programming-language support
- ☑ Program Loading and Execution
- ☑ Communications

Operating System Design & Implementation

Design Goals:

1st Problem: -   Defining Goals and specification

- Choice of Hardware
- Type of System

Beyond this highest design level, the requirements may be much harder to specify.

Requirements:

➢ User Goals

➢ System Goals

User Requirements:
(User Goals)

The system should be:
Convenient to use,
Easy to learn & use,
Reliable, safe & fast

The system should be:
Easy to design, implement,
maintain, operate.
It should be flexible, reliable
error free & efficient

Designer,
Engineer Requirements:
(System goals)

Mechanisms and Policies:
Mechanisms determine how to do something.

Policies determine what will be done.

One important principle is the separation of policy from mechanism.

Good and Flexible   Not Good

Implementation:

> Once an operating system is designed, it must be implemented.
> Traditionally, operating systems have been written in assembly language.
> Now, however, they are most commonly written in higher-level languages such as C or C++

Advantages of writing in high level languages:

- ➢ The code can be written faster
- ➢ It is more compact
- ➢ It is easier to understand and debug
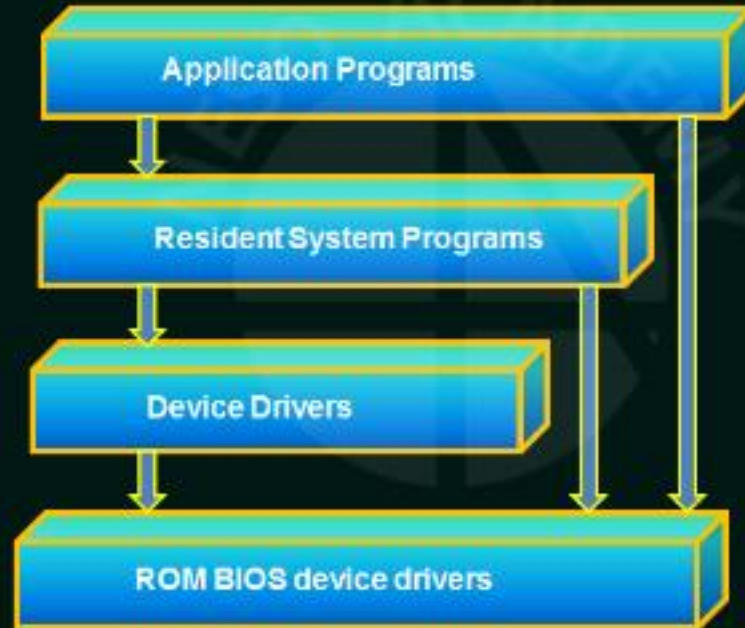- ➢ It is easier to port

E.g.
 MS-DOS was written in Intel 8088 assembly language. Consequently, it is available on only the Intel family of CPUs.
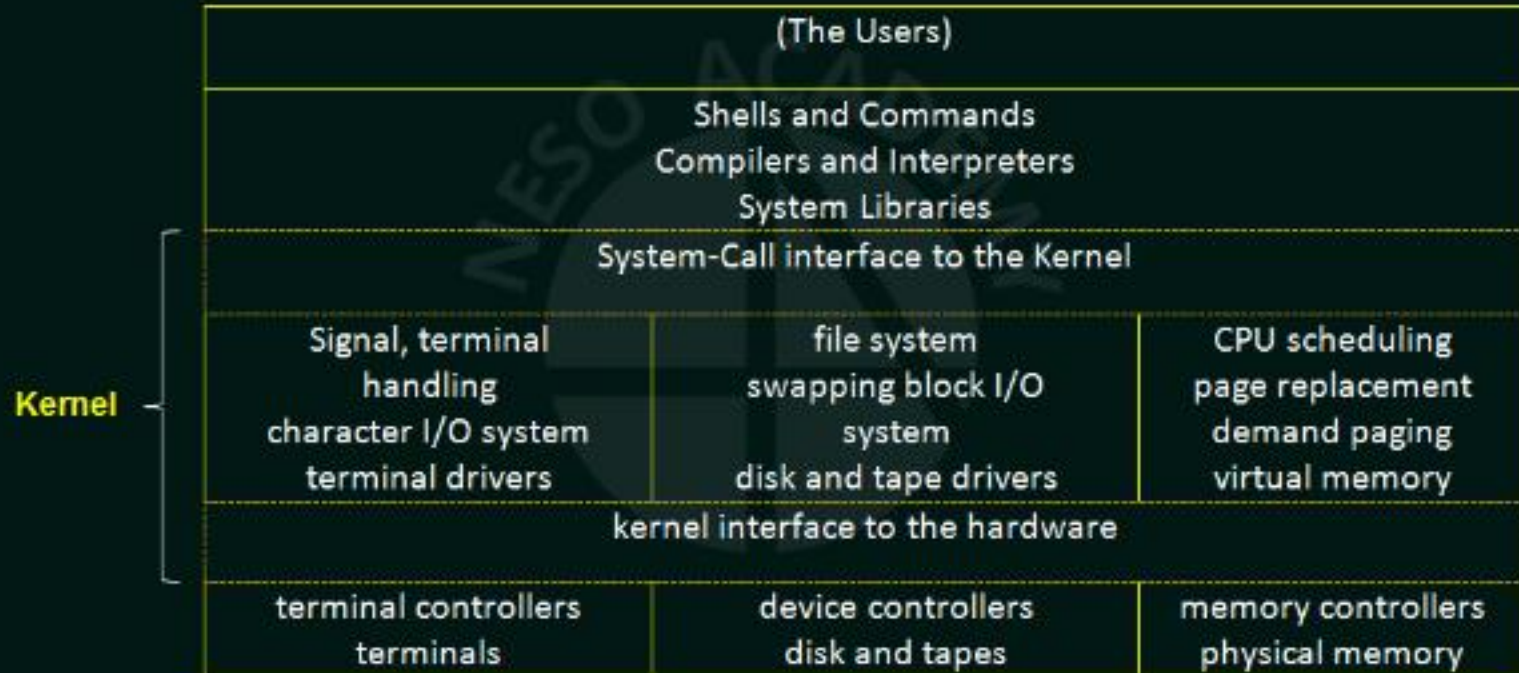
The Linux operating system, in contrast, is written mostly in C and is available on a number of different CPUs, including Intel 80X86, Motorola 680X0, SPARC, and MIPS RX000.
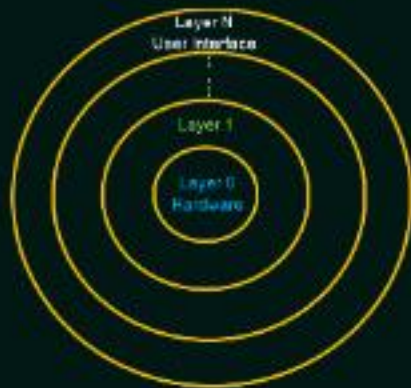
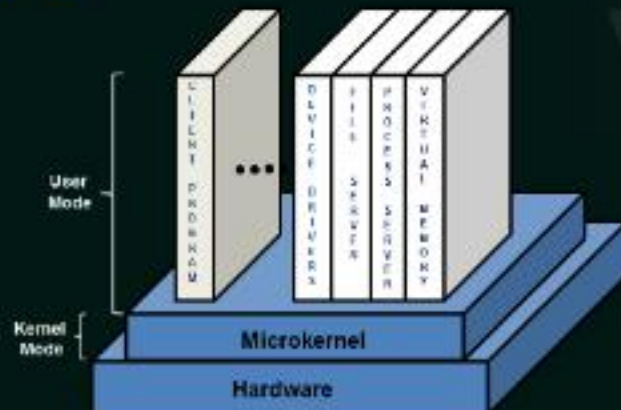# Structures of Operating System

Simple Structure

Monolithic Structure

(The Users)		
Shells and Commands Compilers and Interpreters System Libraries		
System-Call interface to the Kernel		
Signal, terminal handling character I/O system terminal drivers	file system swapping block I/O system disk and tape drivers	CPU scheduling page replacement demand paging virtual memory
kernel interface to the hardware		
terminal controllers terminals	device controllers disk and tapes	memory controllers physical memory

Kernel — spans from "System-Call interface to the Kernel" through "kernel interface to the hardware"

Layered Structure

Microkernels

Modules
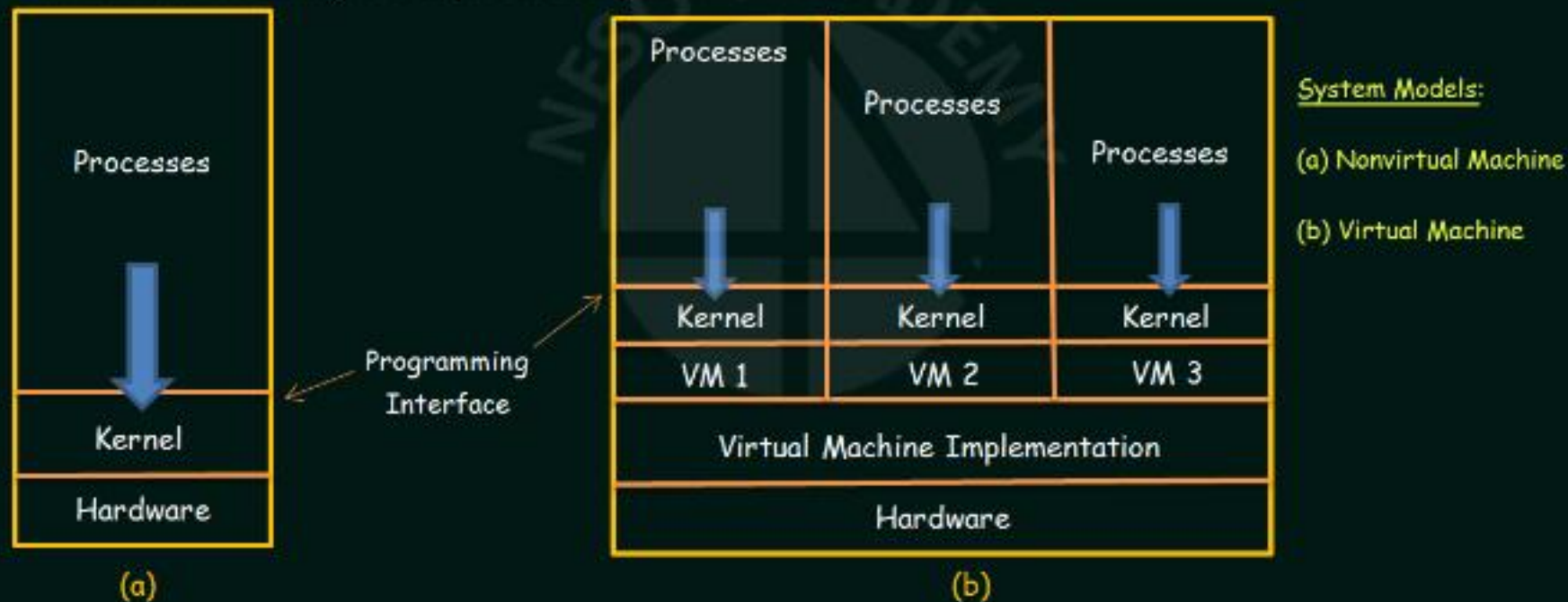
Virtual Machines

The fundamental idea behind a virtual machine is to abstract the hardware of a single computer (the CPU, memory, disk drives, network interface cards, and so forth) into several different execution environments, thereby creating the illusion that each separate execution environment is running its own private computer.

System Models:

(a) Nonvirtual Machine

(b) Virtual Machine

IMPLEMENTATION

Virtual Machine Software –   Runs in Kernel mode
Virtual Machine itself –   Runs in User Mode

Just as the physical machine has two modes, however,
so must the virtual machine.

Consequently, we must have:
- A virtual user mode and
- A virtual kernel mode

BOTH OF WHICH RUN IN A PHYSICAL USER MODE

Processes	Processes	Processes
Kernel	Kernel	Kernel
VM 1	VM 2	VM 3
Virtual Machine Implementation		
Hardware		
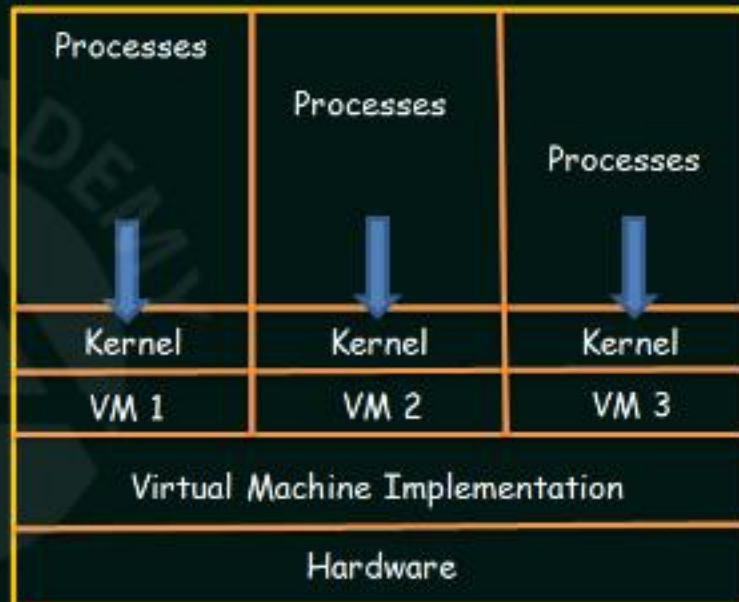
Operating System Generation & System Boot

- Design, code, and implement an operating system specifically for one machine at one site

- Operating systems are designed to run on any of a class of machines at a variety of sites with a variety of peripheral configurations

- The system must then be configured or generated for each specific computer site, a process sometimes known as system generation (SYSGEN) is used for this

  The following kinds of information must be determined by the SYSGEN Program:

  - What CPU is to be used?
  - How much memory is available?
  - What devices are available?
  - What operating-system options are desired?

System Boot

- The procedure of starting a computer by loading the kernel is known as booting the system.

- On most computer systems, a small piece of code known as the bootstrap program or bootstrap loader locates the kernel

- This program is in the form of read-only memory (ROM), because the RAM is in an unknown state at system startup. ROM is convenient because it needs no initialization and cannot be infected by a computer virus.

When the full bootstrap program has been loaded, it can traverse the file system to find the operating system kernel, load it into memory, and start its execution.
It is only at this point that the system is said to be
RUNNING

www.nesoacademy.org

Neso Academy