Introduction to OS

OS Concepts and Structure

MOS 1.4 – 1.7

Mahmoud El-Gayyar

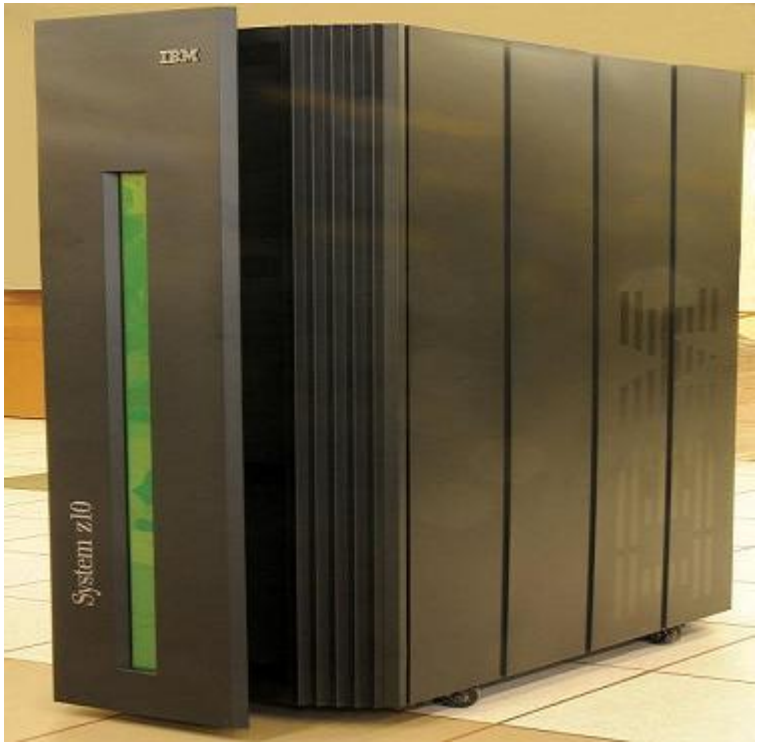
elgayyar@ci.suez.edu.eg

Outline

- ***The OS Zoo***
- The OS Concepts
- The OS Structure

Mainframe Operating Systems

- Big-thousands of disks….
- Lots of jobs with lots of I/O
- three kinds of services:
  - Batch
  - Transaction processing
  - Timesharing
- Elderly-Unix, Linux replacing them



Server Operating Systems

- personal computers, workstations, or even mainframes
- File, print, web servers
- FreeBSD, Linux, Windows

Personal Computer Operating Systems

- Linux
- Mac
- Windows

Sensor Node Operating Systems

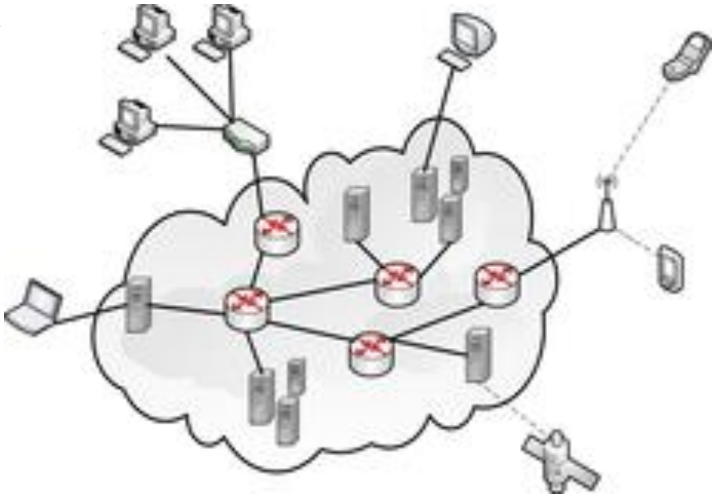
- Sensor nodes tiny computers that communicate with each other and with a base station using wireless communication...
- Each sensor node is a real computer, with a CPU, RAM, ROM, and one or more environmental sensors.
- ***TinyOS*** is a well-known operating system for a sensor node.

Real-Time Operating Systems

- Hard (e.g. factory) deadline
- Soft (e.g. multi-media) deadline
- Example: ***e-Cos***

Distributed Operating Systems

- distributed applications are running on multiple computers linked by communications.
- This system looks to its users like an ordinary centralized operating system
- The users of a true distributed system should not know, on which machine their programs are running and where their files are stored.
- ***LOCUS*** and ***MICROS*** are examples of distributed operating systems.



Outline

- The OS Zoo
- ***The OS Concepts***
- The OS Structure

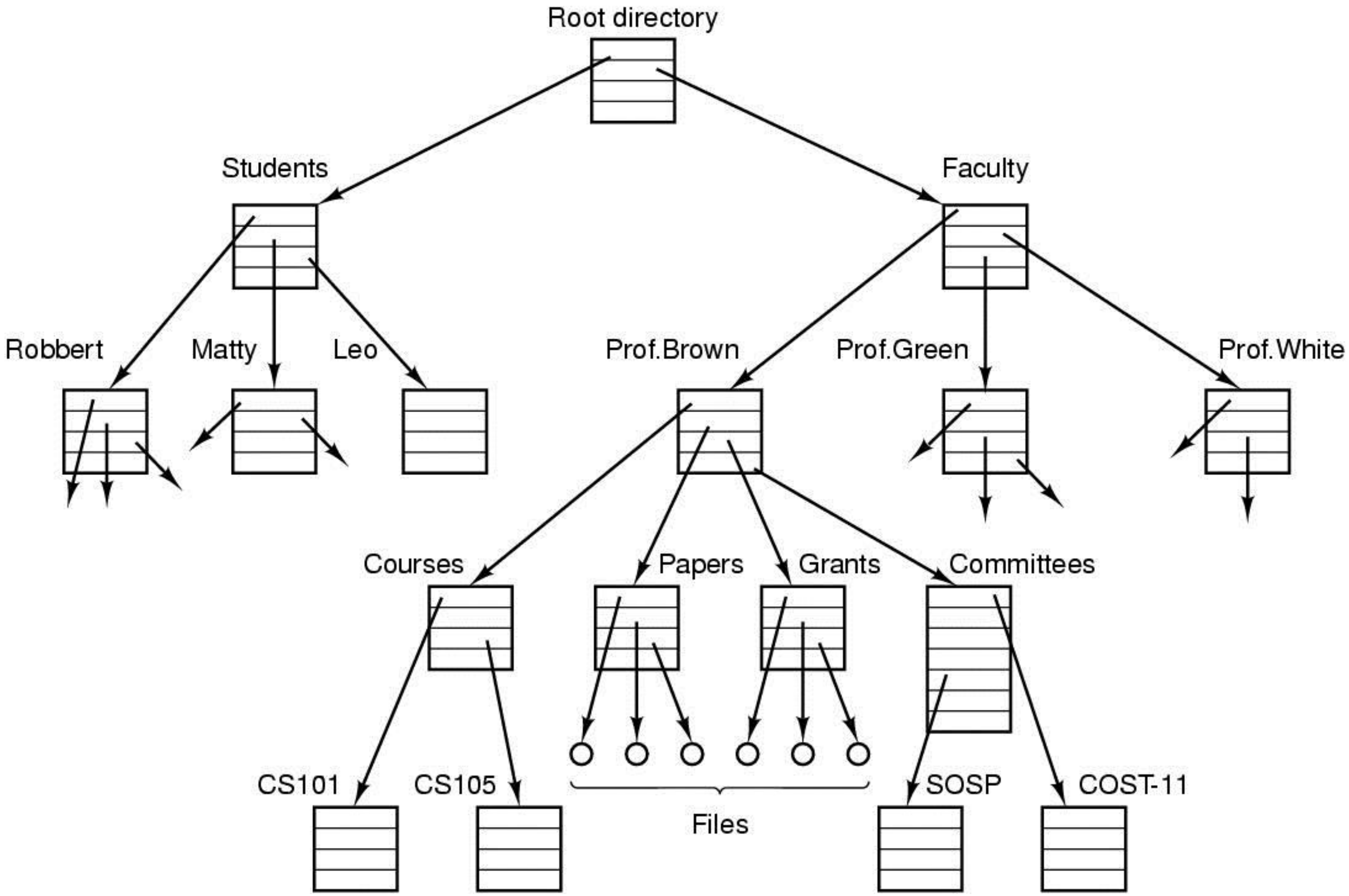
The OS Concepts

- Processes
- Address spaces
- Files
- The Shell
- System Calls

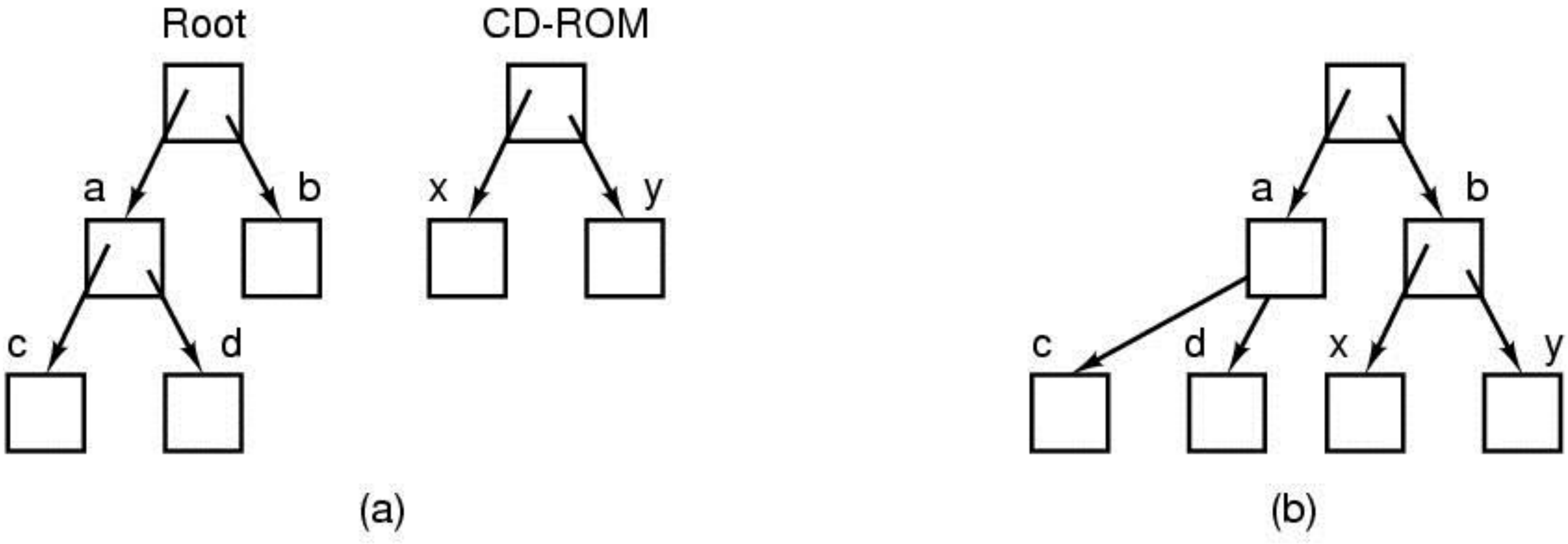
Processes

- Program in execution
- Lives in ***address space***
- ***Process table***
  - Keeps info about process
  - Used to re-start process (Why?)
- More details in CH. 2

File Directory



Mounting Files in UNIX



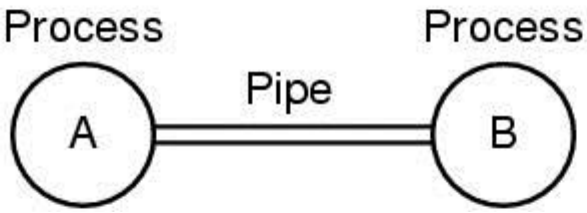
A CD-ROM is mounted on directory b.

Special Files

- Special files to represent I/O devices
  - OS treats them as files.
  - ***Block special files*** (disks)
  - ***Character special files*** (line printers, modems)
  - Kept in ***/dev*** directory, e.g. /dev/lp is line printer

Unix Pipe

- A pipe is a sort of pseudo file that can be used to connect two processes
- Processes communicate by writing into/ reading from a file in Unix



- A and B write into the pipe and read from the pipe.

The Shell

- UNIX command interpreter, called the ***shell***
- Has lots of flavors - sh, bash, csh, ssh…..

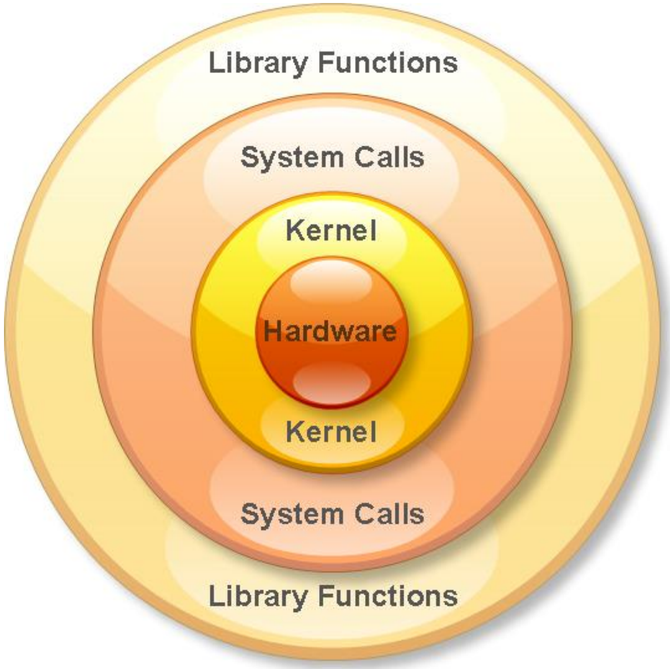
- Sort <file1 >file2

- cat file1 file2 file3 | sort > /dev/lp

- cat filel file2 file3 | sort > /dev/lp ***&***

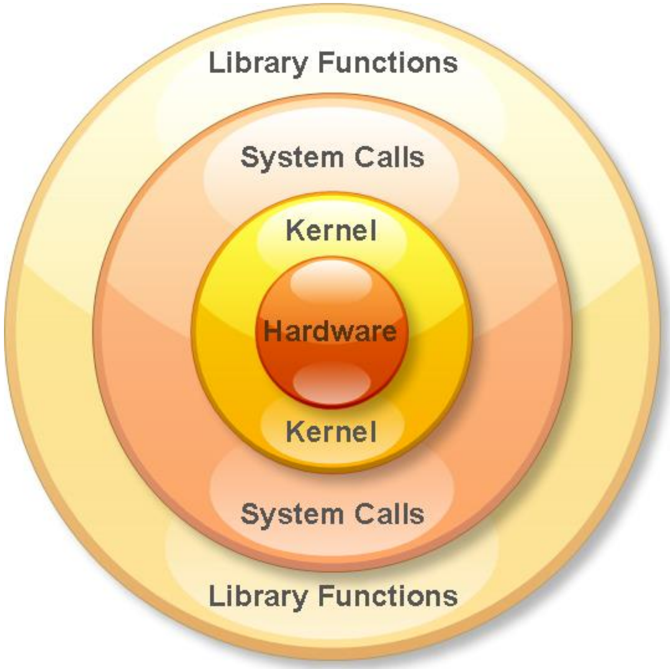
System Calls

- Interface between user programs and OS
- Varies from OS to OS
- e.g. Read data from a file..
  - it has to execute a ***trap instruction*** to context switch from user space to kernel space (also termed as ‘privileged mode’ or ‘superuser mode’)
  - Finds actual routine for system call in a table
  - Does the work involved in the call
  - Returns to user program
- As an example, consider the system call that is used to create a process.



Library Functions

- Library functions always execute in ***user space*** (also termed as ‘user mode‘). Hence, they cannot interact directly with the hardware.
- Library functions in-turn may utilize system calls for performing certain tasks which can only be carried out only in ‘kernel mode’.
- Library function with no system calls are faster (no context switch)

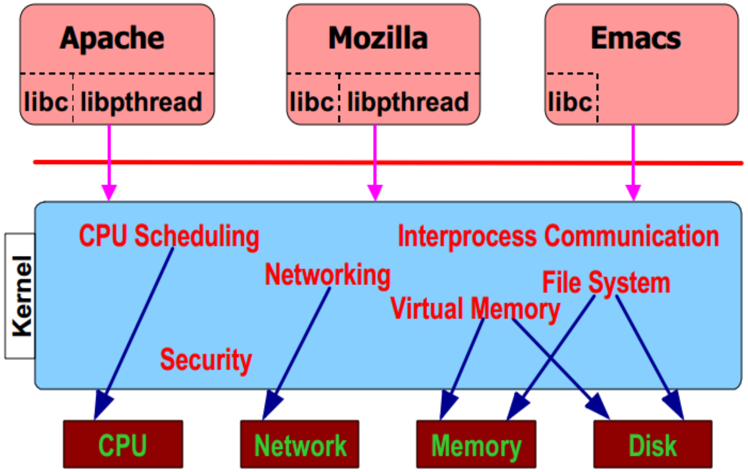


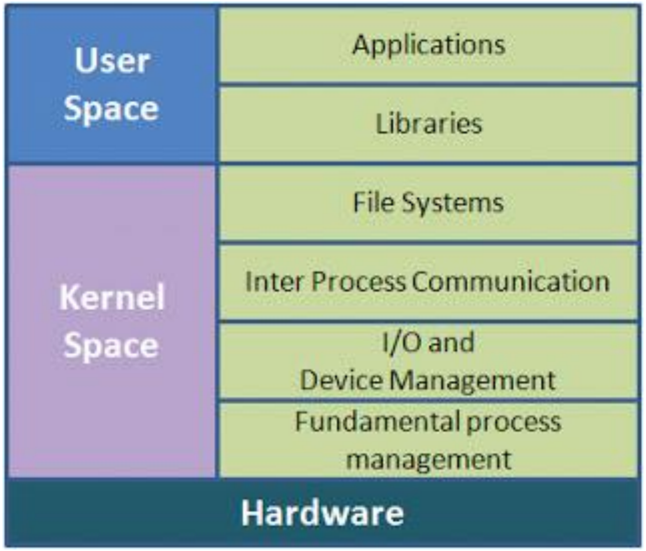
Outline

- The OS Zoo
- The OS Concepts
- ***The OS Structure***

Monolithic Systems

- All kernel routines are together
- A system call interface
- Examples:
  - Linux
  - Windows NT/XP
- Pros
  - Good performance
- Cons
  - Inflexible: Adding new features is not easy and need to recompile the whole source code
  - Instable: No protection between Kernel components





User Space

Applications

Libraries

Kernel Space

File Systems

Inter Process Communication

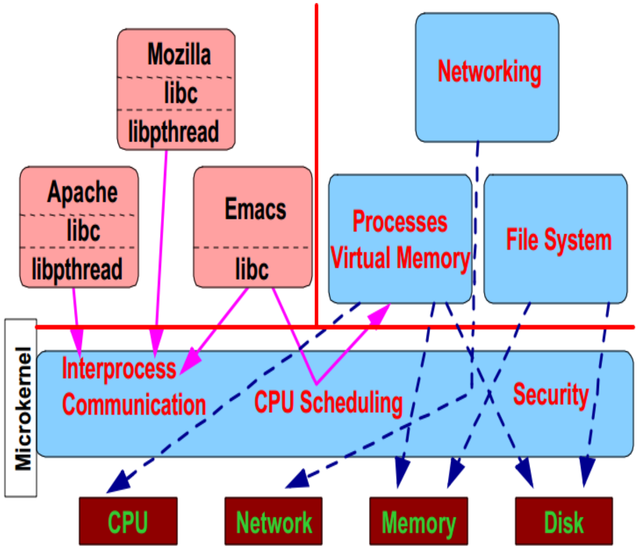
I/O and Device Management

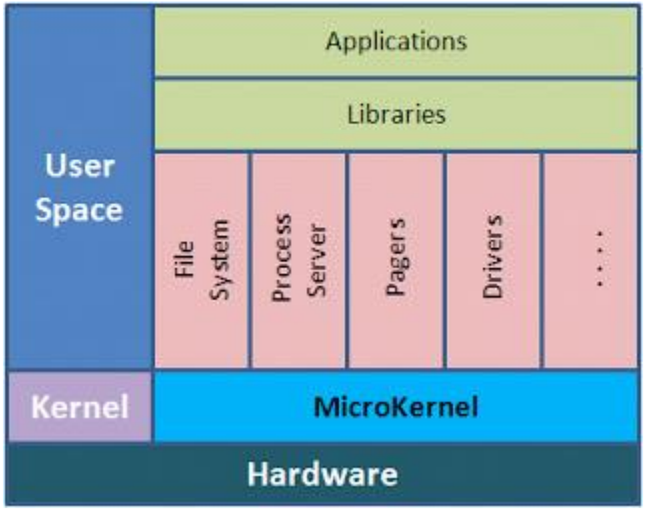
Fundamental process management

Hardware

Microkernels Systems

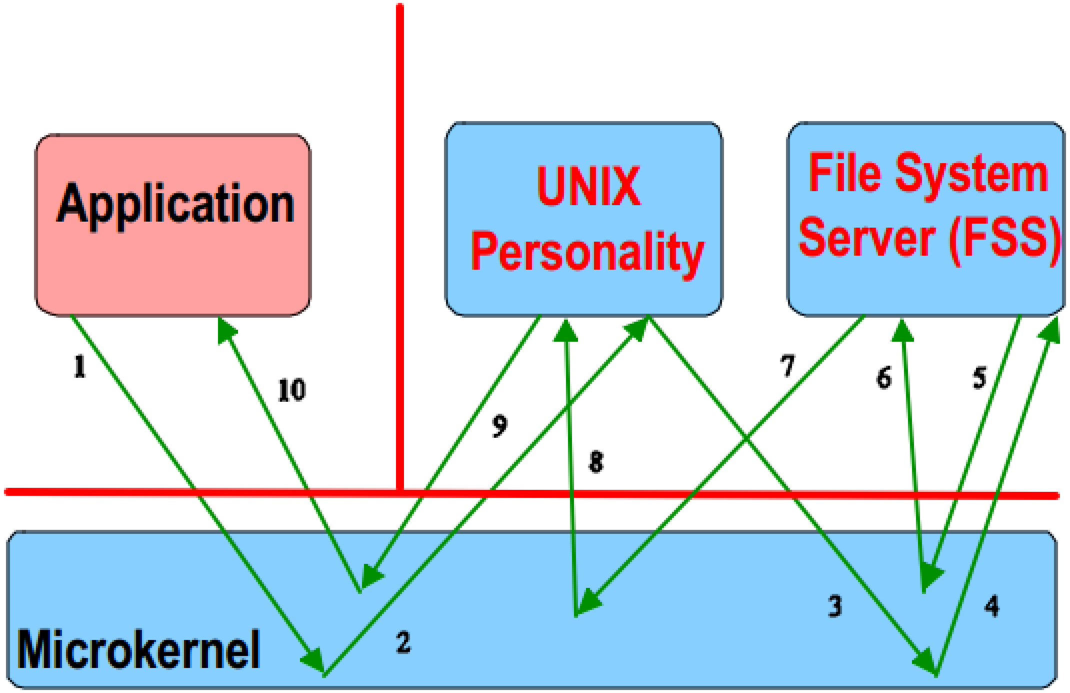
- reduce the kernel to basic process communication and I/O control
- other system services reside in user space in form of normal processes (as so called servers)
- message system is used for communication
- Examples:
  - Mach
  - QNX, a real-time OS for embedded systems.
- Pros
  - Flexibility: new functionality = add new server
  - Fault isolation (more reliable)
- Cons
  - Inefficient (Lots of boundary crossings)





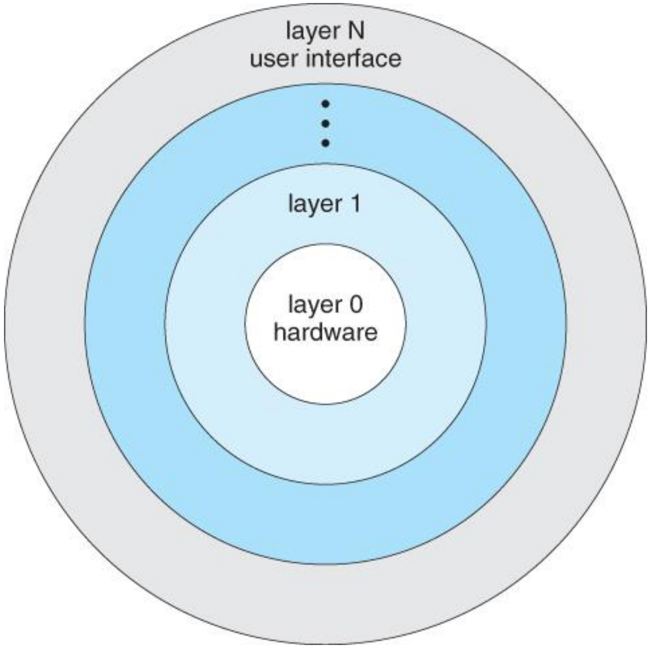
Microkernels Sytem Call Example

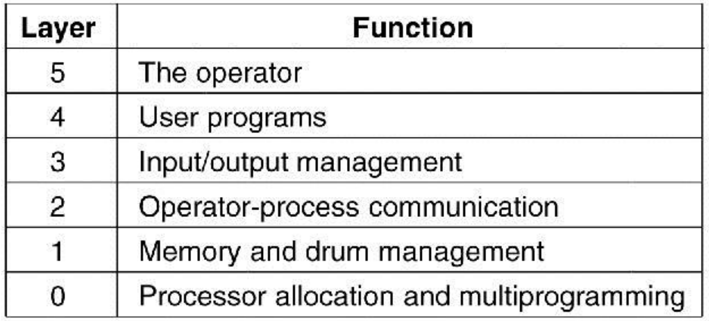
1. Application calls read(), traps to microkernel
2. microkernel sends message to Unix Personality requesting read
3. Unix personality sends message to File System Server (FSS) asking for data
4. FSS receives message and begins processing
5. FSS sends message to microkernel asking for disk blocks
6. Microkernel sends data back to FSS
7. FSS sends message to UNIX Personality with results
8. Unix Personality receives message with data
9. Unix Personality sends data to Application
10. Application receives data



Layered Systems

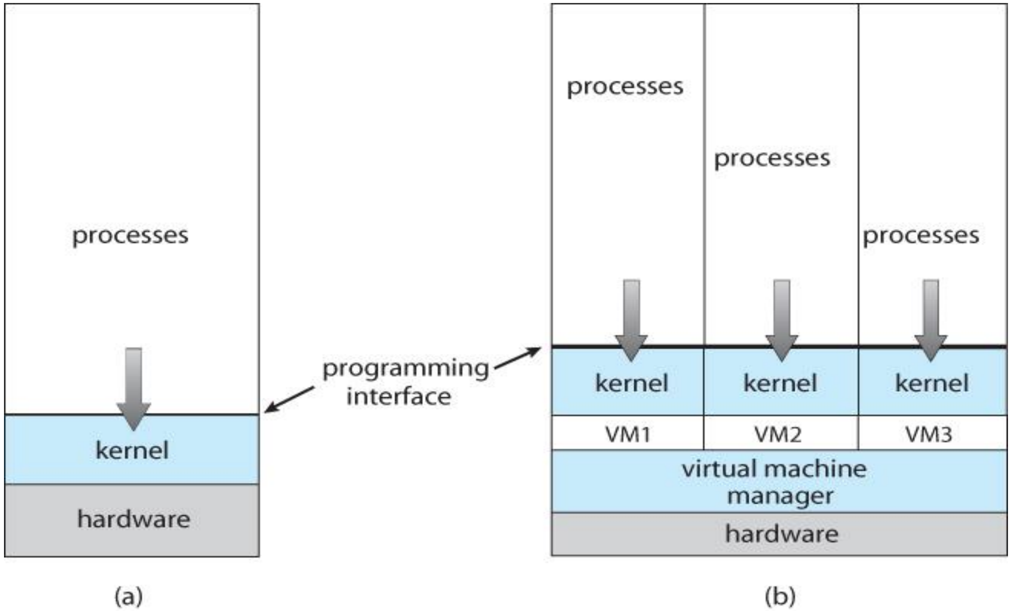
- each layer rests on the layer below it, and relies solely on the services provided by the next lower layer.
- Examples:
  - THE (6 layers)
  - DOS (4 layers)
- Pros
  - Layer abstraction (Easy dev.)
- Cons
  - Inflexible: Which order of layers?
  - Inefficient: service from higher layer has to filter through all lower layers





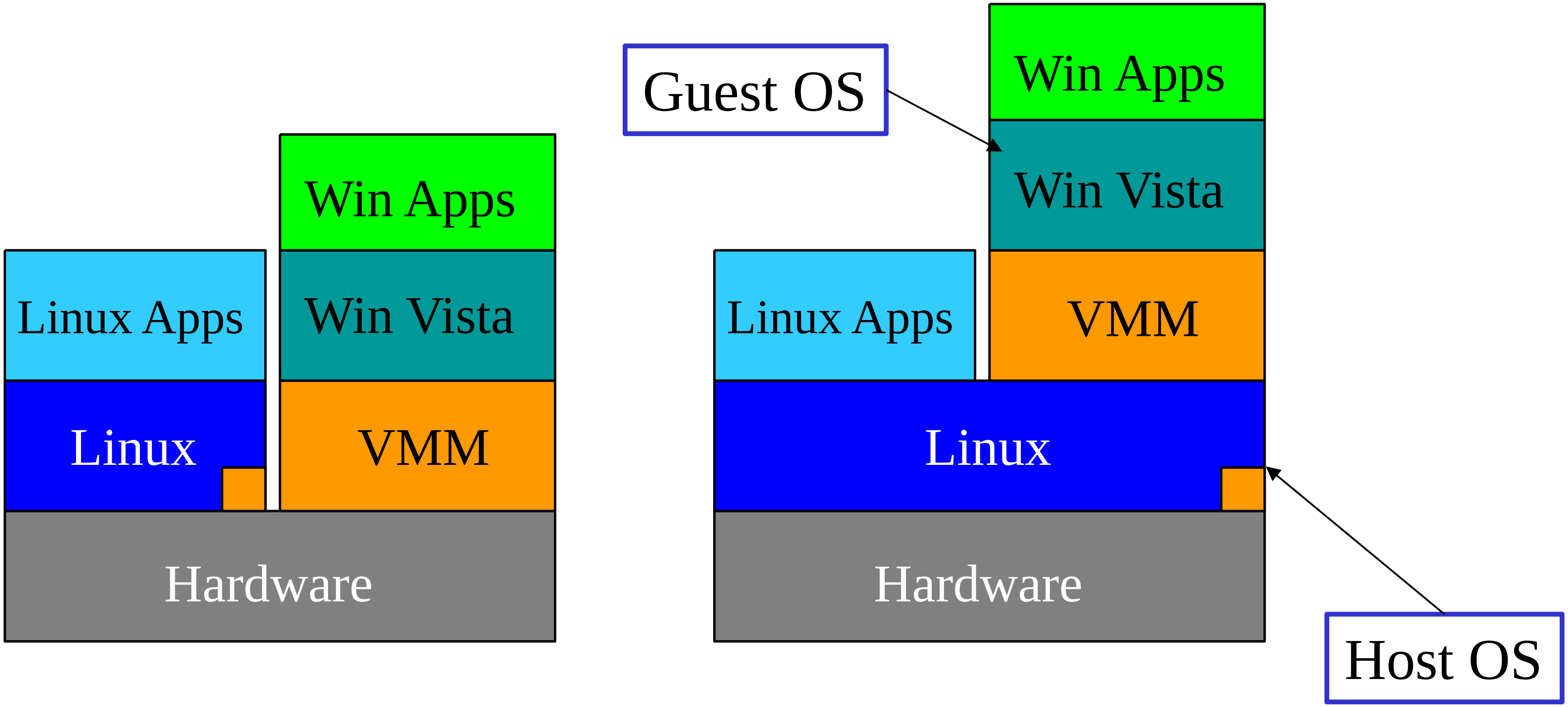
Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

The Virtual Machines



- The concept is to provide an interface that looks like independent hardware, to multiple different OS(es) running simultaneously on the same physical hardware.
- Each OS believes that it has access to and control over its own CPU, RAM, I/O devices, hard drives, etc.
- Examples: Virtual Box, Java VM, VMWare, Xen

Virtual Machine Hypervisors



VMM runs on hardware

Type 1 Hypervisor

VMM as an application

Type 2 Hypervisor

Review

- What is the difference between a trap and an interrupt?
- Why is the process table needed in a timesharing system? Is it also needed in personal computer systems in which only one process exists, that process taking over the entire machine until it is finished?
- Is there any reason why you might want to mount a file system on a nonempty directory?

  If so, what is it?