KG REDDY
College of Engineering
& Technology

Operating Systems

Introduction to Operating System (OS)

Mr.M.Saidi Reddy

Slides Credits for all PPTs of OS course

- The slides/diagrams in this course are an adaptation, combination, and enhancement of material from the following resources and persons:

1. Slides of A. Silberschatz, P. B. Galvin and G. Gagne (see book references ahead) with some enhancements by Y. Wiseman.
2. Slides of Mario Marchand of the University of Ottawa and Patricia Roy of Manatee Community College based on W. Stallings (see book reference ahead) with some enhancements by O. Kremien and E. Gruengard.
3. Some diagrams from R. A. Finkel and Andrew S. Tanenbaum (see book references ahead).

Initial Objectives

- To describe the basic organization of computer systems and operating systems.
- To give an overview of the many types of computing environments.
- To explore varied types of operating systems.
- To provide a grand tour of the major components of operating systems.
- To describe the services an operating system provides to users, processes, and other systems.
- To discuss the various ways of structuring an operating system.

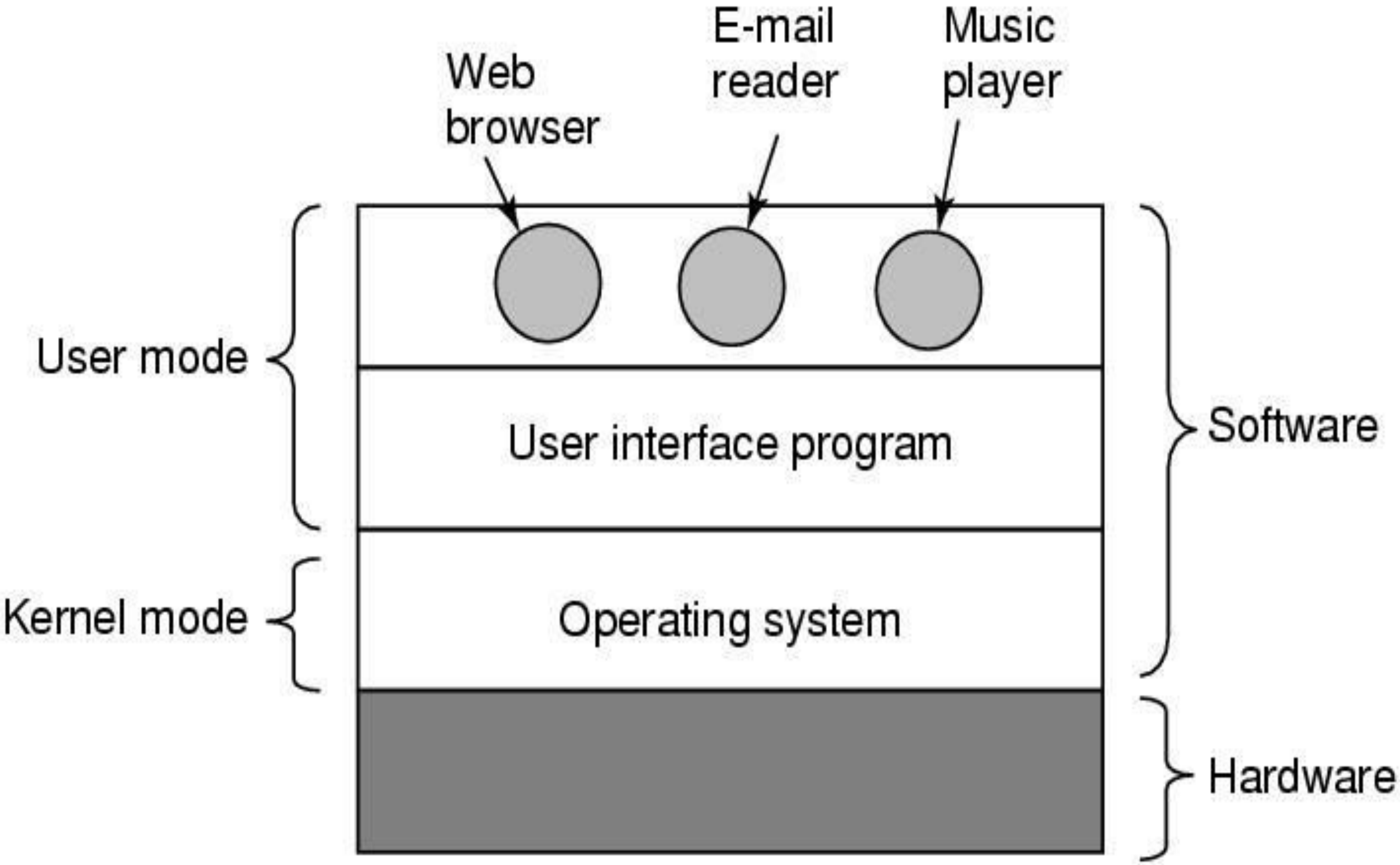
What is an Operating System (1)?

- A modern computer consists of:
  - One or more processors
  - Main memory
  - Disks
  - Printers
  - Various input/output devices.
- Managing all these varied components requires a layer of software – the **Operating System (OS).**

What is an Operating System (2)?

- An Operating System is a program that acts as an intermediary/interface between a user of a computer and the computer hardware.
- OS goals:
  - Control/execute user/application programs.
  - Make the computer system convenient to use.
  - Ease the solving of user problems.
  - Use the computer hardware in an efficient manner.

Where does the OS fit in?



Services provided by an OS

- Facilities for program creation
  - editors, compilers, linkers, debuggers, etc.
- Program execution
  - loading in memory, I/O and file initialization.
- Access to I/O and files
  - deals with the specifics of I/O and file formats.
- System access
  - resolves conflicts for resource contention.
  - protection in access to resources and data.

Why are Operating Systems Important?

- Important to understand and know how to correctly use when writing user applications.
- Large and complex systems that have a high economic impact and result in interesting problems of management.
- Few actually involved in OS design and implementation but nevertheless many general techniques to be learned and applied.
- Combines concepts from many other areas of Computer Science: Architecture, Languages, Data Structures, Algorithms, etc.

Course Syllabus (1)

- **Motivation for Operating Systems (OS)**
- **Introduction**
  - What's an Operating System?
  - Computer/Operating System Overview
  - Evolution of Operating Systems
  - Functional/Protection Aspects
  - Operating System Structures

Course Syllabus (2)

- **Concurrent Processes**
  - Process Models and Management
  - Process Description and Control
  - Task/Thread Description and Control
  - Concurrency: Mutual Exclusion and Synchronization
  - Concurrency: Deadlock and Starvation

Course Syllabus (3)

- **Memory Management**
  - Real Memory Management
  - Motivation for Virtual Memory (VM)
  - Paging and Segmentation
  - Page Fetch, Placement and Replacement

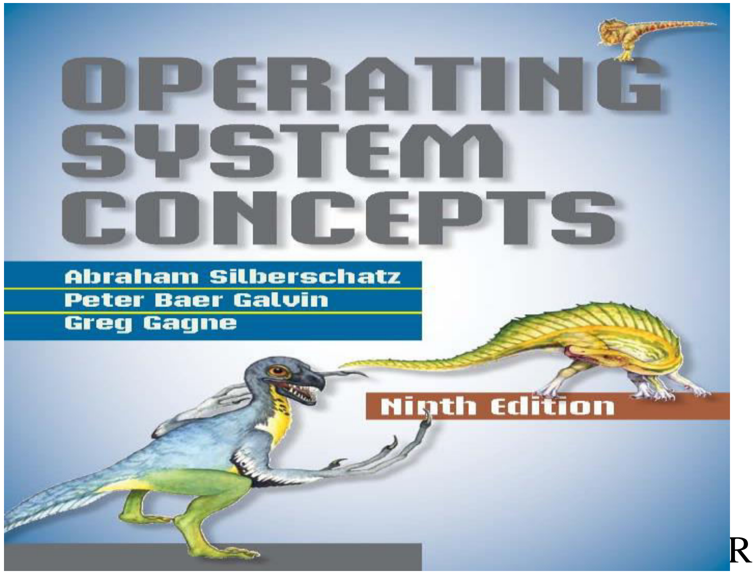
Course Syllabus (4)

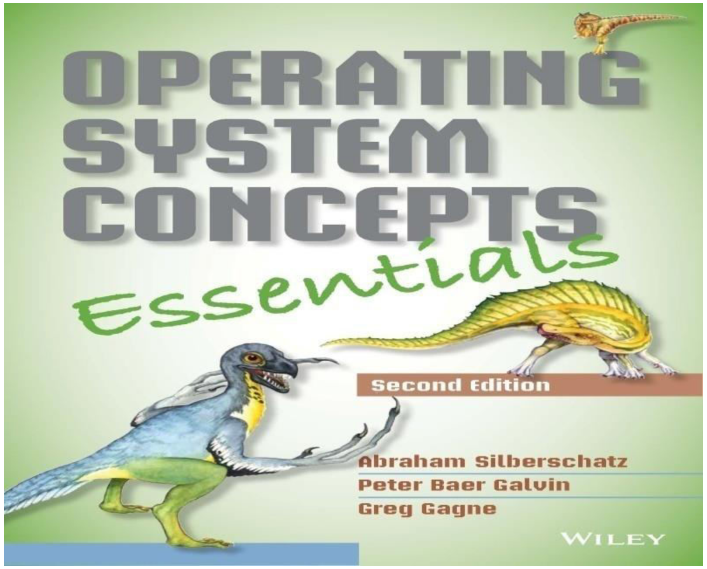
- **Uniprocessor Scheduling**
  - Levels of CPU Scheduling
  - Process Scheduling
- **External Storage Management**
  - File Systems/Management
  - Directories
  - File Allocation
  - Disk Scheduling

Main Bibliography

A. Silberschatz, P. B. Galvin, and G. Gagne, "Operating Systems Concepts (Essentials)", 9th Edition, John Wiley & Sons, 2012.

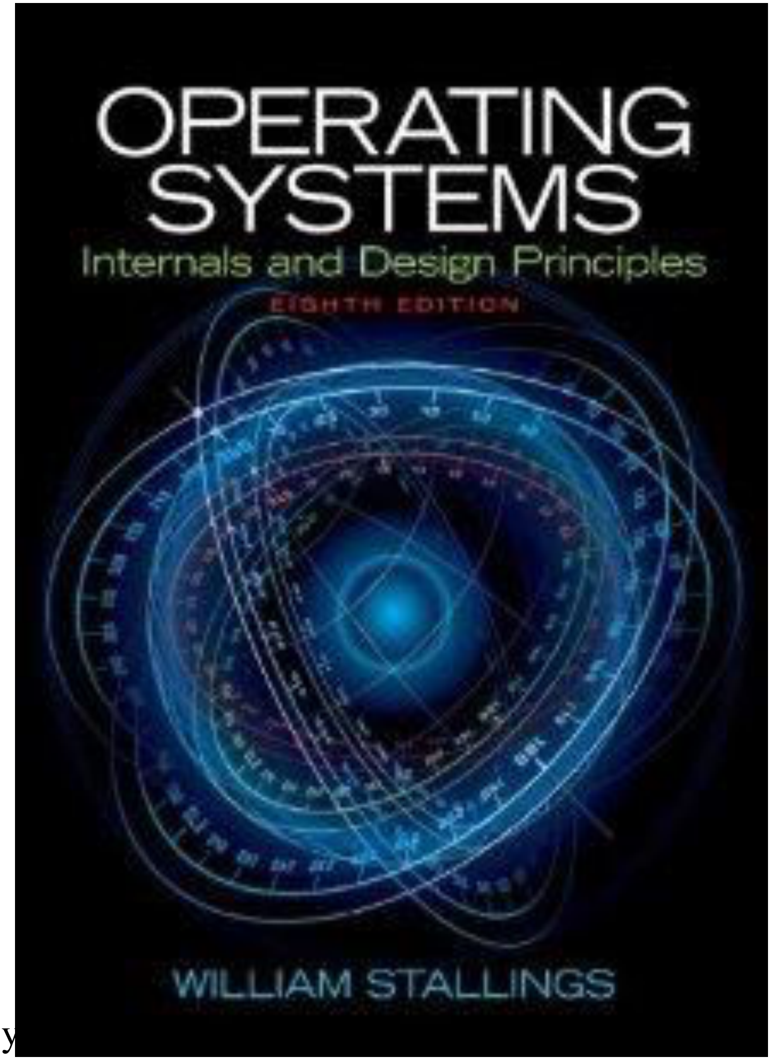
http://codex.cs.yale.edu/avi/os-book/





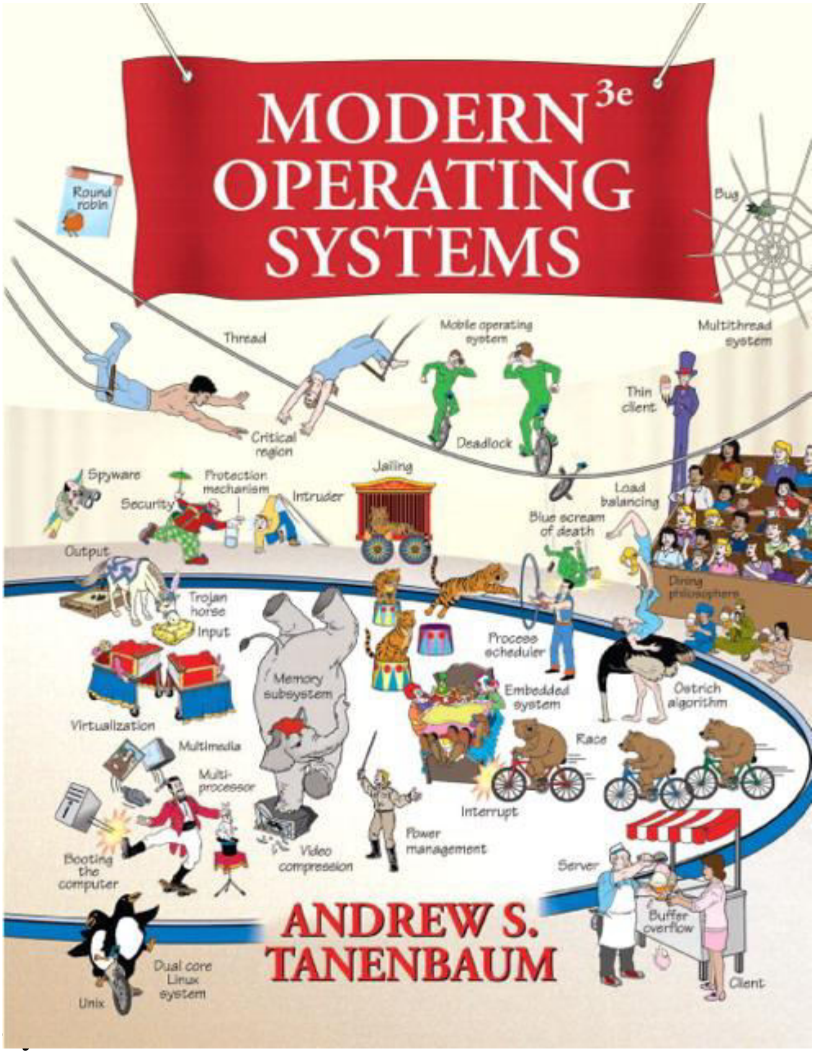
Main Bibliography

W. Stallings, "Operating Systems: Internals and Design Principles", 8th ed, Pearson, 2015. http://williamstallings.com/OperatingSystems/



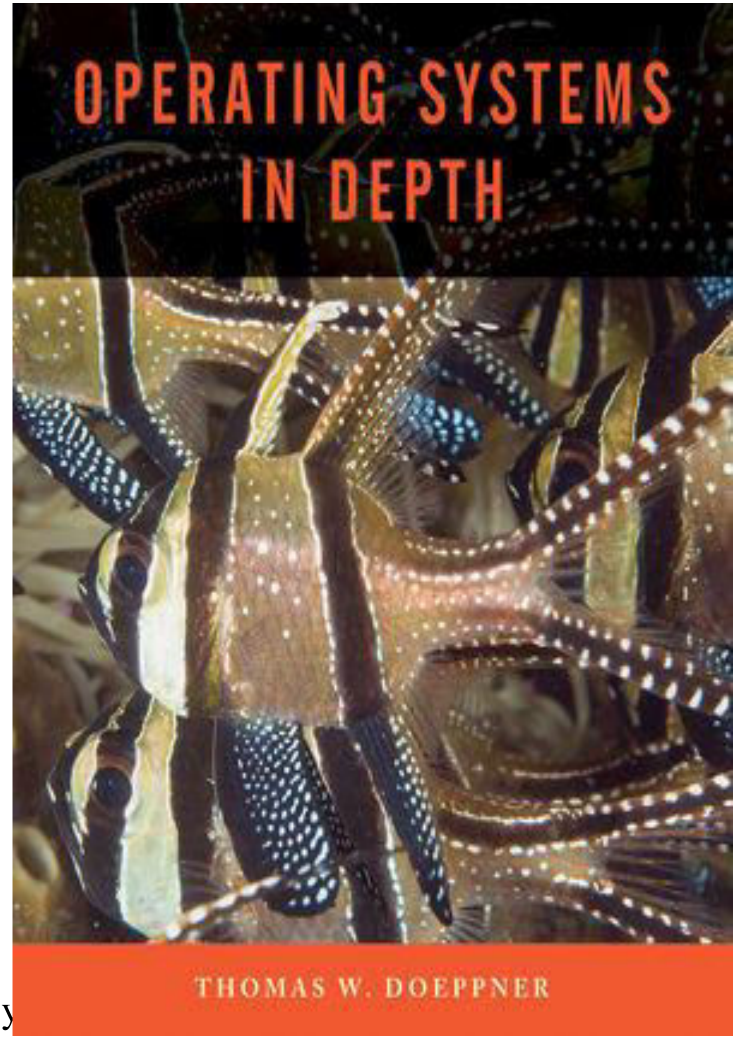
Main Bibliography

A. S. Tanenbaum, “Modern Operating Systems”, 4th ed, Pearson, 2015. http://www.pearsonhighered.com/educator/academic/product/0,,0136006639,00%2ben-USS_01DBC.html

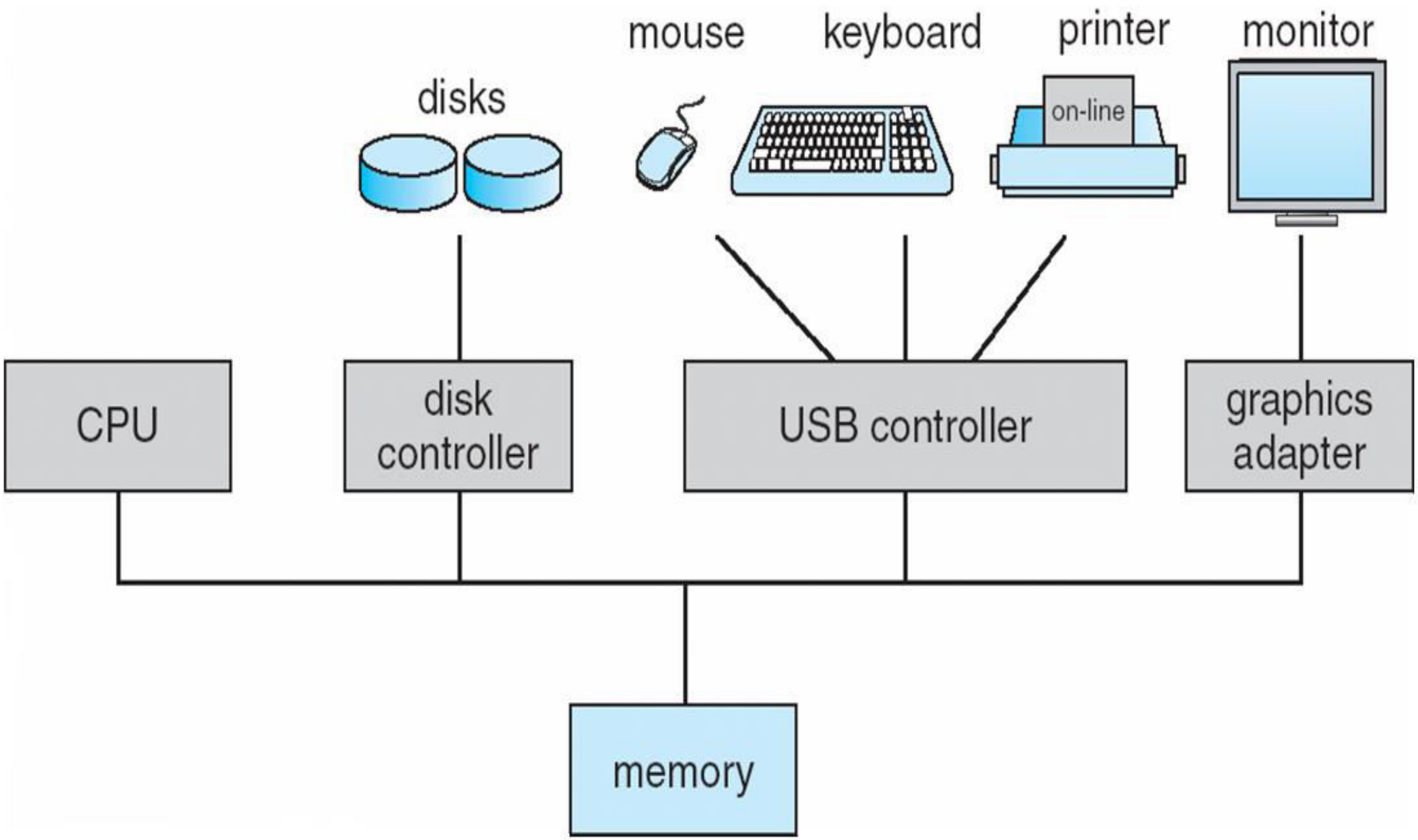


Main Bibliography

T. W. Doeppner, Operating Systems in Depth, John Wiley & Sons, 2011, http://eu.wiley.com/WileyCDA/WileyTitle/productCd-EHEP001803.html



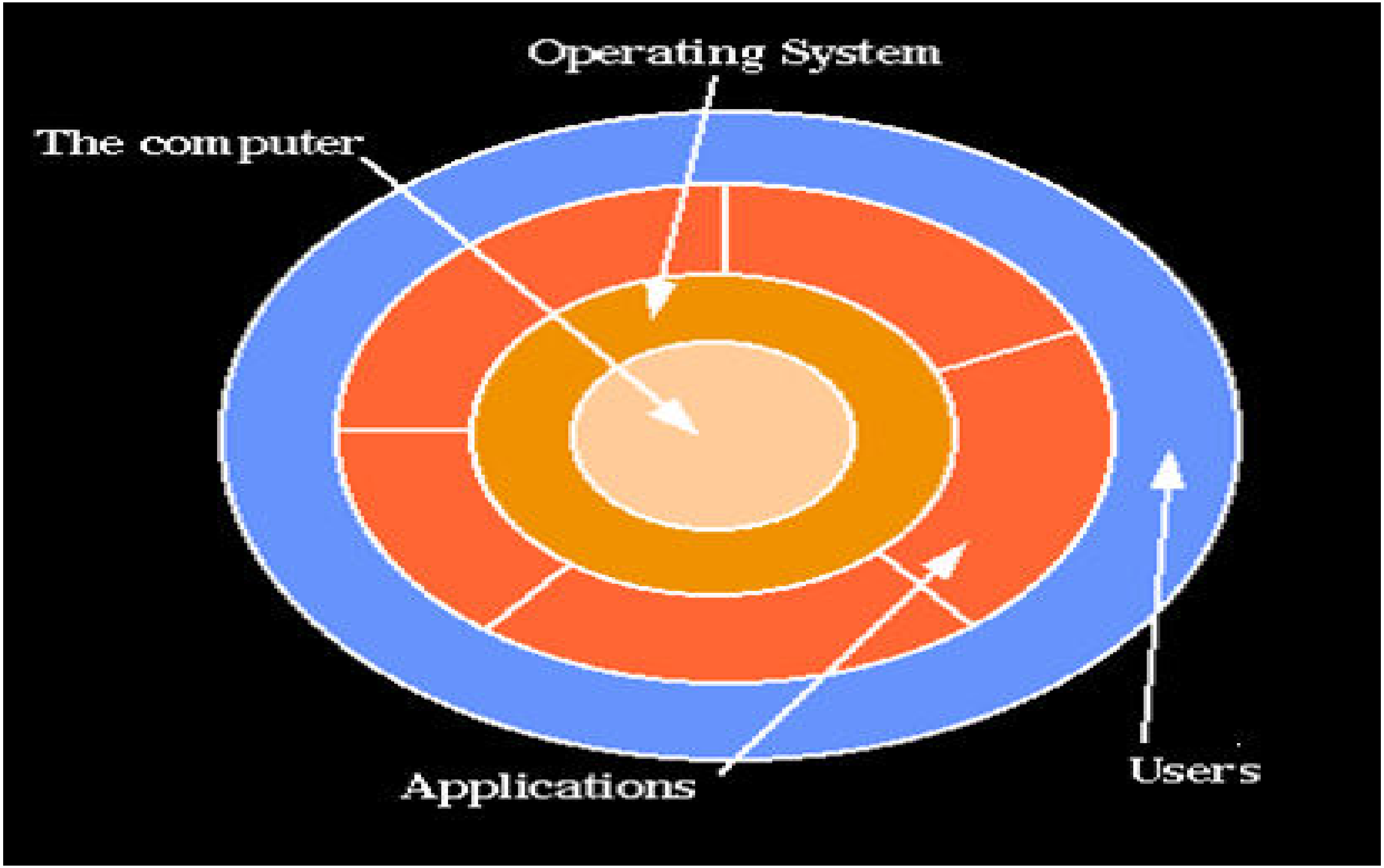
Computer Hardware Organization



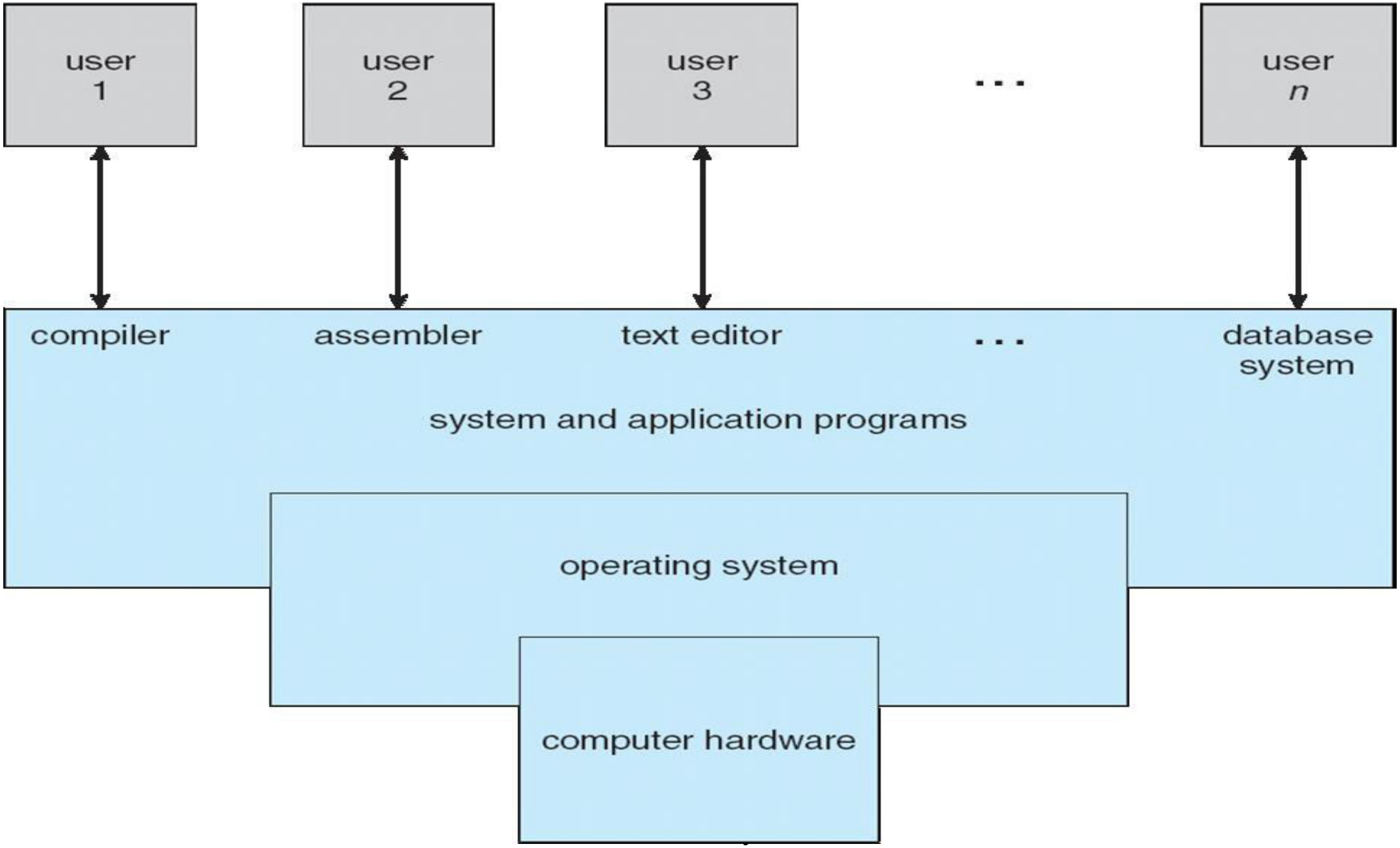
Computer System Components

1. Hardware – provides basic computing resources (CPU, Memory, I/O devices, Communication).
2. Operating System – controls and coordinates use of the hardware among various application programs for various users.
3. System & Application Programs – ways in which the system resources are used to solve computing problems of the users (Word processors, Compilers, Web browsers, Database systems, Video games).
4. Users – (People, Machines, other computers).

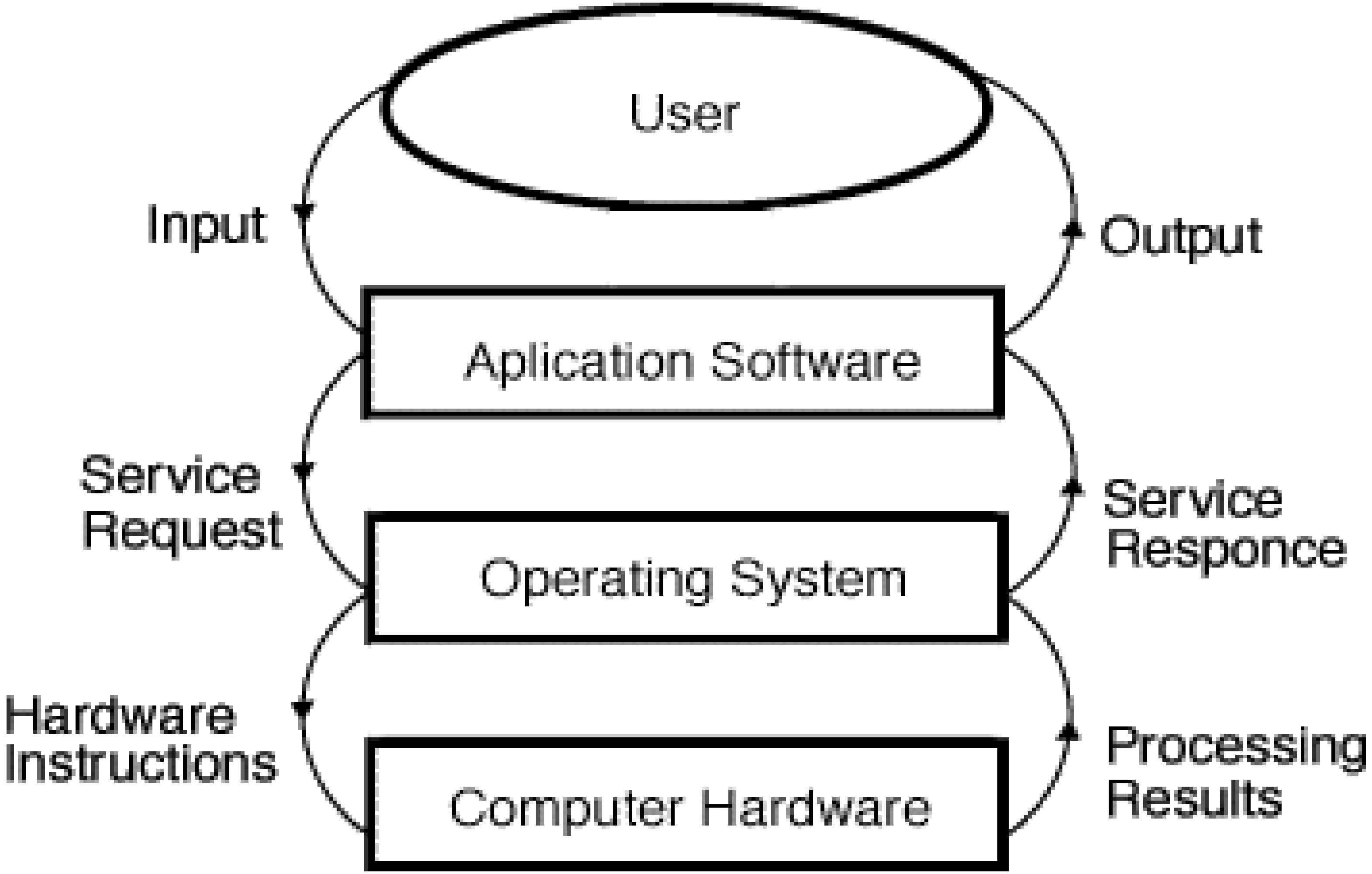
Hierarchical view of computer system



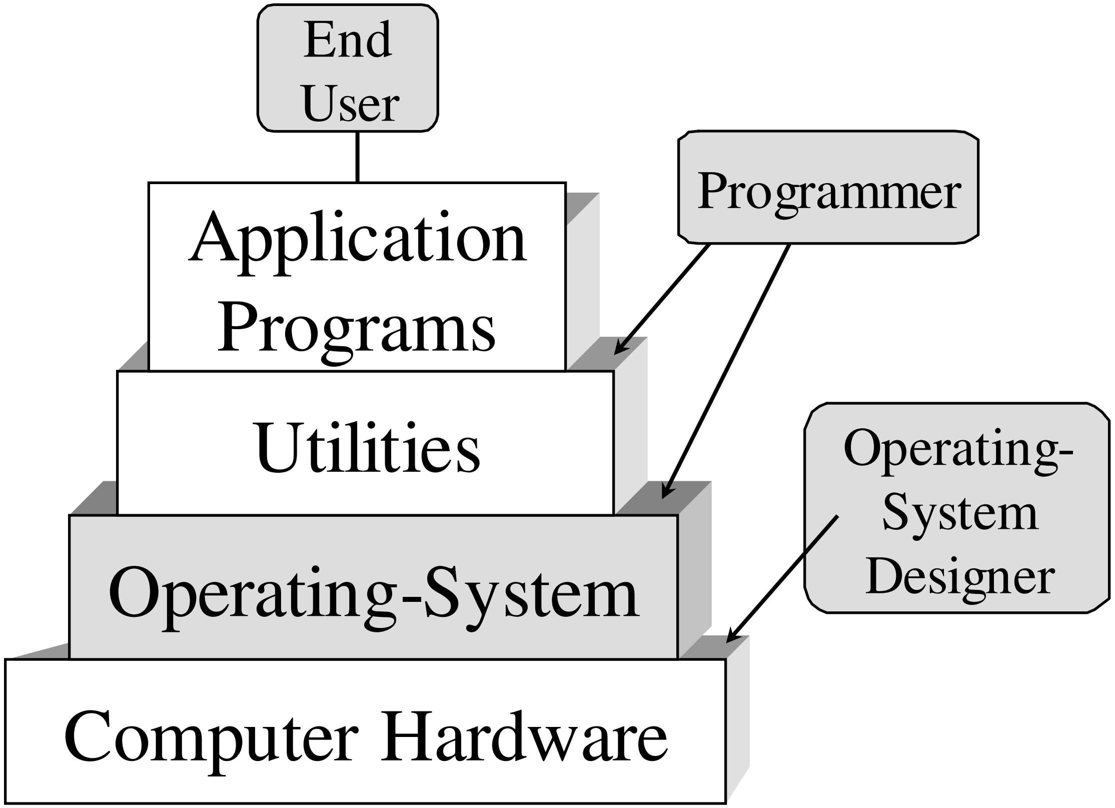
Static View of System Components



Dynamic View of System Components



Layers of a Computer System


End User

Programmer

Operating-System Designer

Application Programs

Utilities

Operating-System

Computer Hardware

What Operating Systems Do

- Depends on the point of view.
- Users want convenience, ease of use and good performance
  - Don’t care about resource utilization.
- But a shared computer such as mainframe or minicomputer must keep all users happy.
- Users of dedicate systems such as workstations have dedicated resources but frequently use shared resources from servers.
- Handheld computers are resource poor, optimized for usability and battery life.
- Some computers have little or no user interface, such as embedded computers in devices and automobiles.

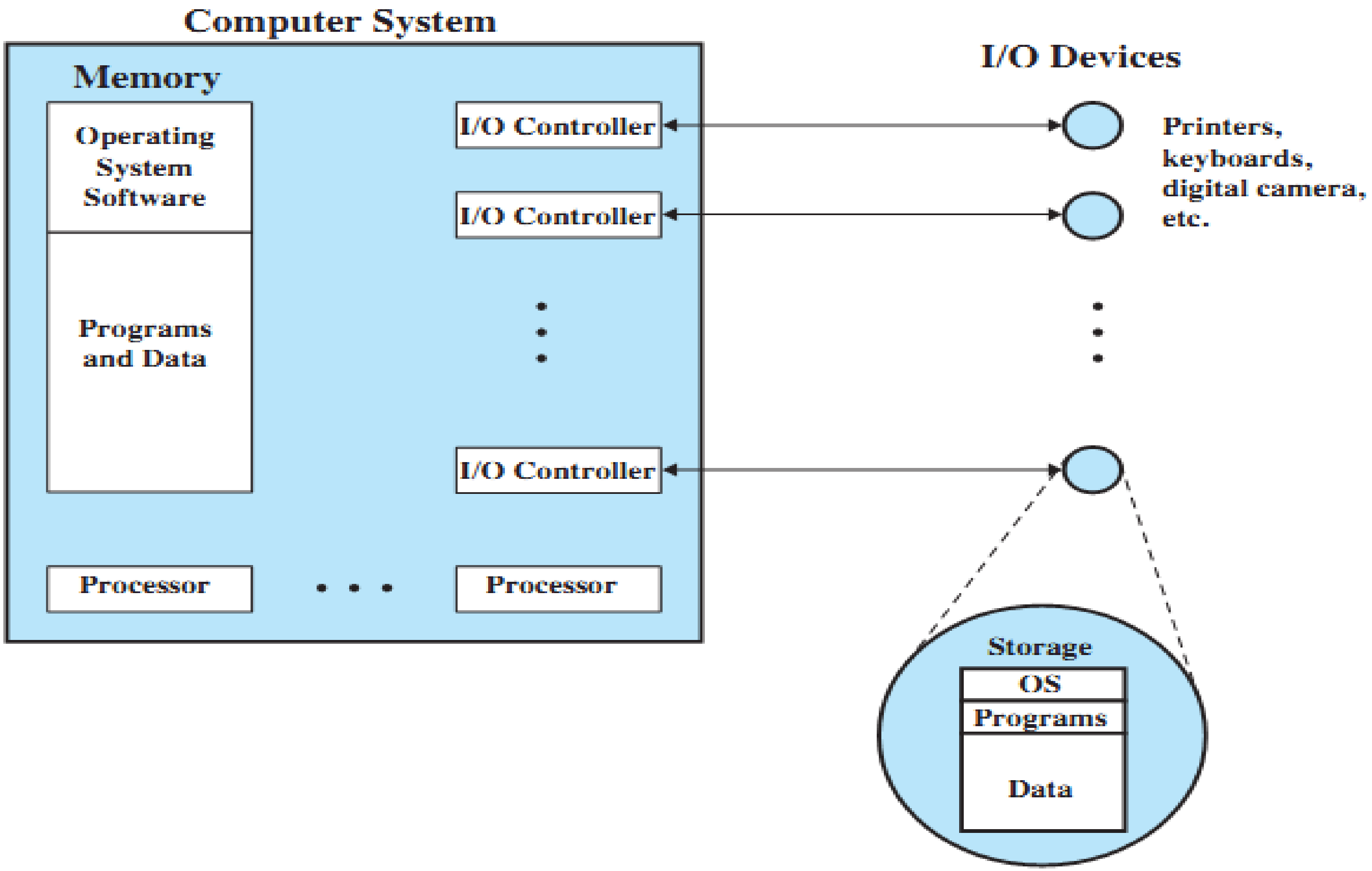
Views of an Operating System

- There are three classical views (in literature):
  1. Resource Manager – manages and allocates resources.
  2. Control program – controls the execution of user programs and operations of I/O devices.
  3. Command Executer – Provides an environment for running user commands.
- But one more modern view: the Operating System as a Virtual Machine.

1. Resource Manager

- Resource Manager:
  - Manages and protects multiple computer resources: CPU, Processes, Internal/External memory, Tasks, Applications, Users, Communication channels, etc…
  - Handles and allocates resources to multiple users or multiple programs running at the same time and space (e.g., processor time, memory, I/O devices).
  - Decides between conflicting requests for efficient and fair resource use (e.g., maximize throughput, minimize response time).
- Sort of a bottom-up view.

OS as a Resource Manager

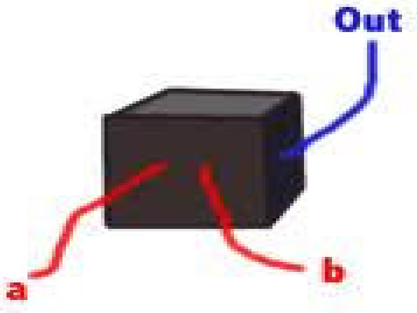


Resource Manager oriented OS names

- DEC RSX – Resource Sharing eXecutive
- MIT Multics – MULTiplexed Information and Computing Services
- IBM MFT/MVT – Multiple Fixed/Variable Tasks
- IBM MVS – Multiple Virtual Storage
- DEC VMS – Virtual Memory System
- MVS TSO – Time Sharing Option
- CTSS – Compatible Time Sharing System
- IBM VM – Virtual machine

2. Control Program

- Control Program:
  - Manages all the components of a complex computer system in an integrated manner.
  - Controls the execution of user programs and I/O devices to prevent errors and improper use of computer resources.
  - Looks over and protects the computer: Monitor, Supervisor, Executive, Controller, Master, Coordinator ….
- Sort of a black box view.



Control program oriented OS names

- Unisys MCP – Master Control Program
- DR CP/M – Control Program/Microcomputer
- IBM VM/CP – VM Control Program
- IBM AIX – Advanced Interactive eXecutive
- DEC RSX – Resource Sharing eXecutive

3. Command Executer

- Command Executer:
  - Interfaces between the users and machine.
  - Supplies services/utilities to users.
  - Provides the users with a convenient CLI (Command Language Interface), also called a Shell (in UNIX), for entering the user commands.
- Sort of a top-down view.

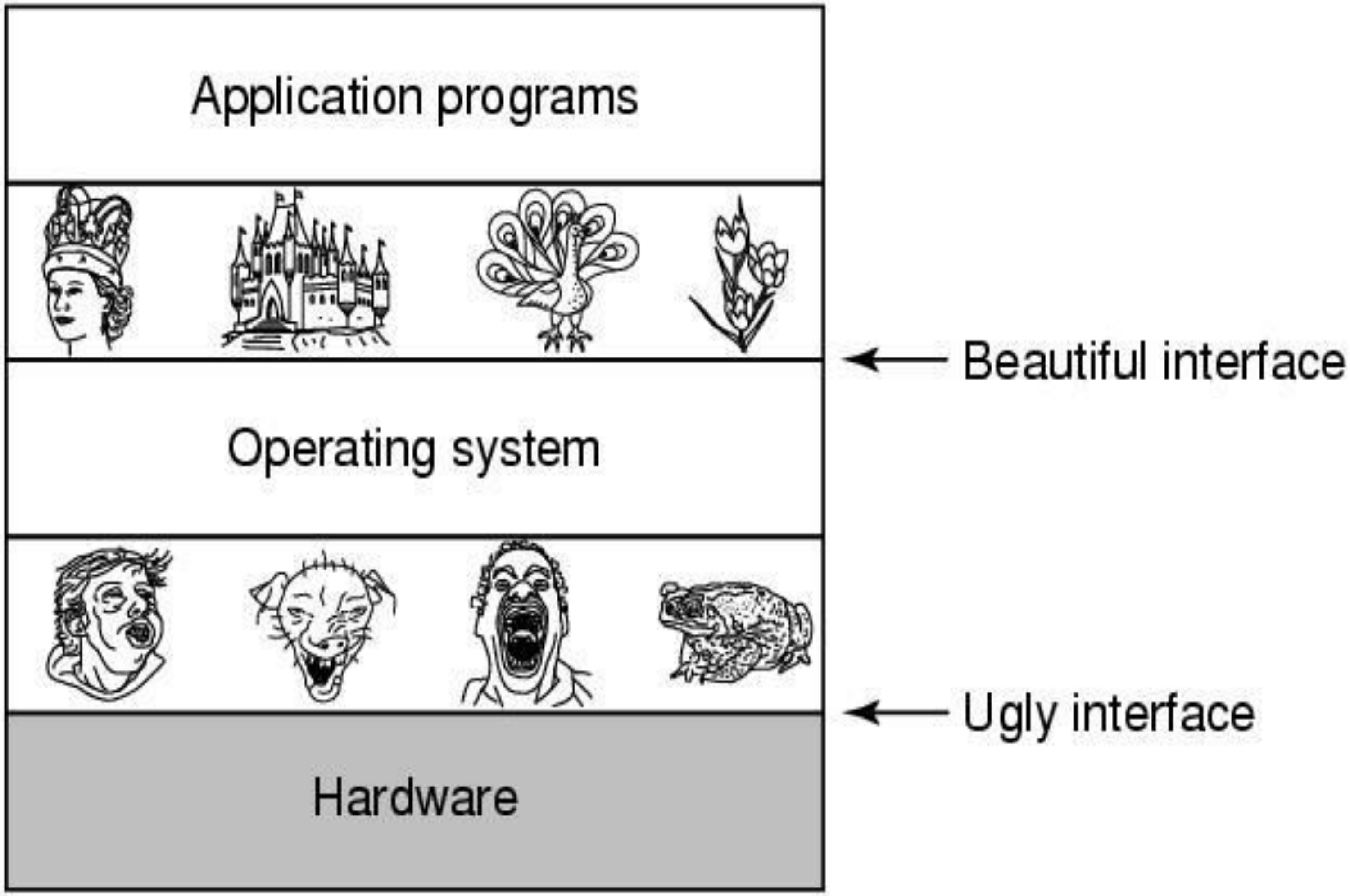
Command Executer oriented OS names

- IBM AIX – Advanced Interactive Executive
- IBM VM/CMS – Conversational monitor System

Modern view: Virtual Machine (1)

- Operating System as a Virtual Machine:
  - An interface between the user and hardware that hides the details of the hardware (e.g., I/O).
  - Constructs higher-level (virtual) resources out of lower-level (physical) resources (e.g., files).
  - **Definition**: OS is a collection of software enhancements, executed on the bare hardware, culminating in a high-level virtual machine that serves as an advanced programming environment.
    - virtual machine = software enhancement = extended machine = abstract machine = layer = level = ring.

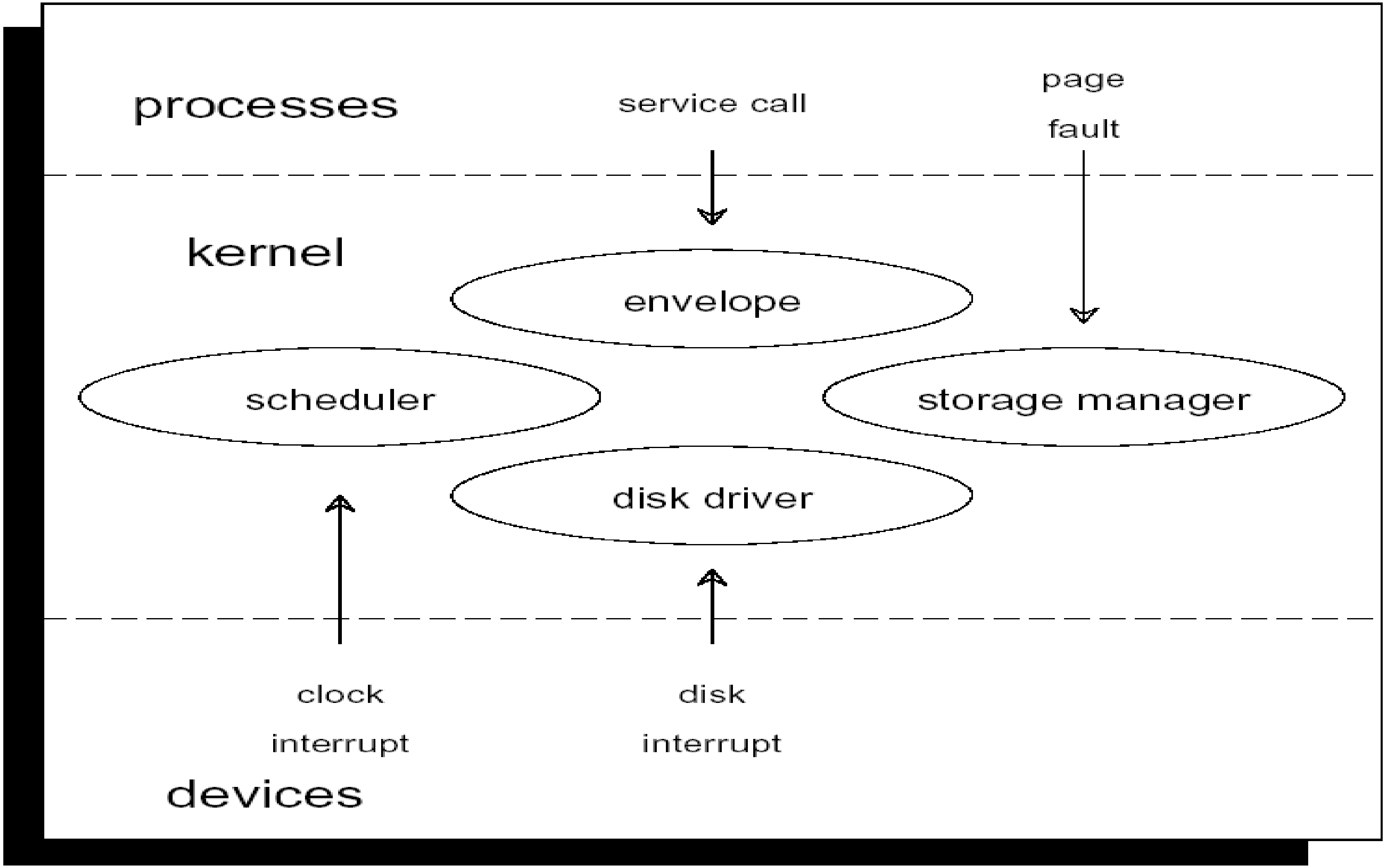
Modern view: Virtual Machine (2)



Definition of Operating System

- There is no universally accepted definition.
- “Everything a vendor ships when you order an operating system” is good approximation but varies widely.
- “The one program running at all times on the computer” is the **Kernel.**
- Everything else is either a system program (ships with the operating system) or an application program.

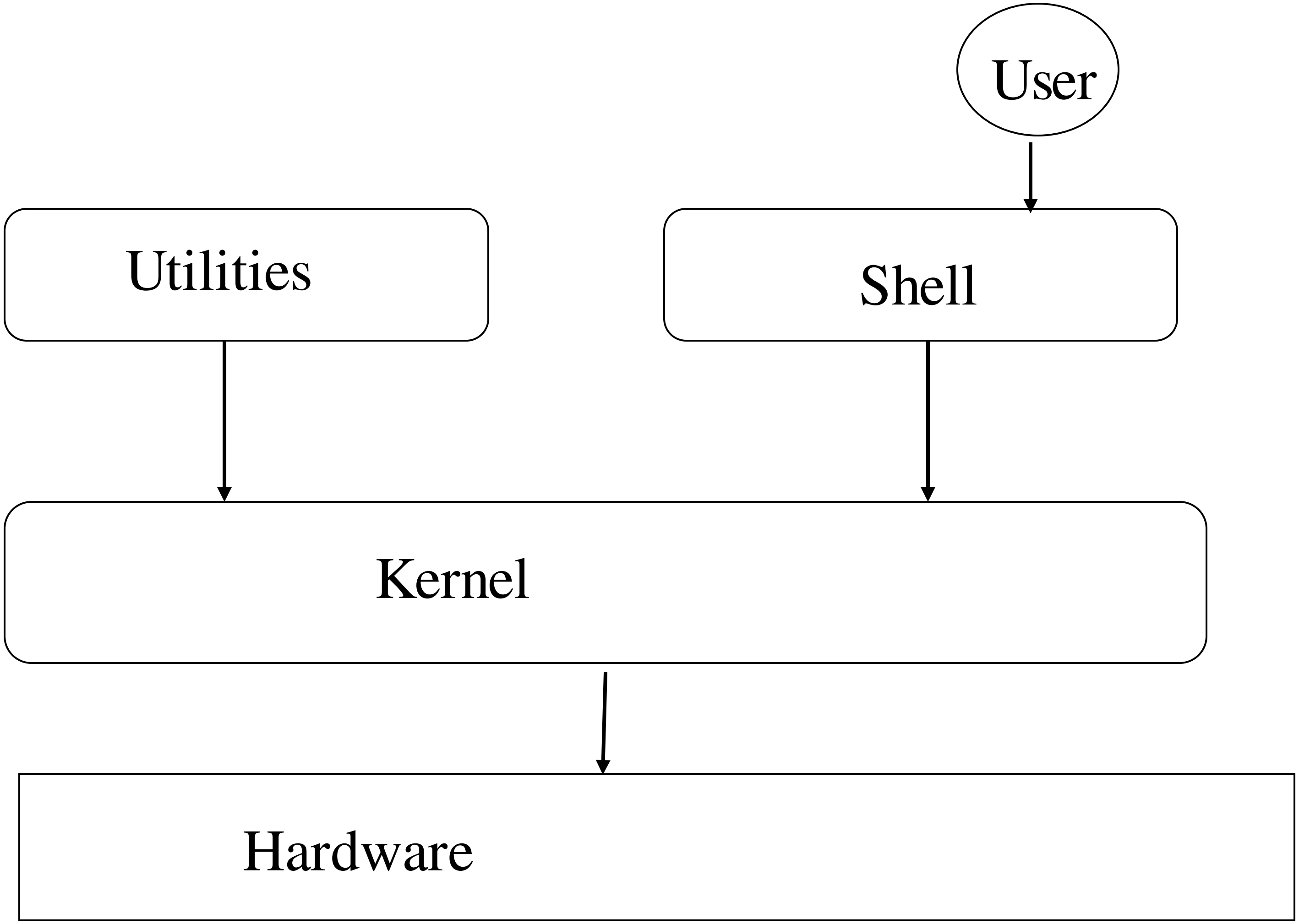
One Kernel Point of View



What is the OS/Kernel?

- Is the Operating System just the Kernel (not the utilities and application programs)?!
- The Command Line Interface (CLI) (or command layer/interpreter or shell) allows direct command entry by the user.
- The shell used to be in the kernel but now is a (first between equals) utility outside of it:
  - Easy to change/debug
  - Many of them (sh, bsh, csh, ksh, tcsh, wsh, bash)
  - Possible to switch between them (chsh)

UNIX Shell and Utilities

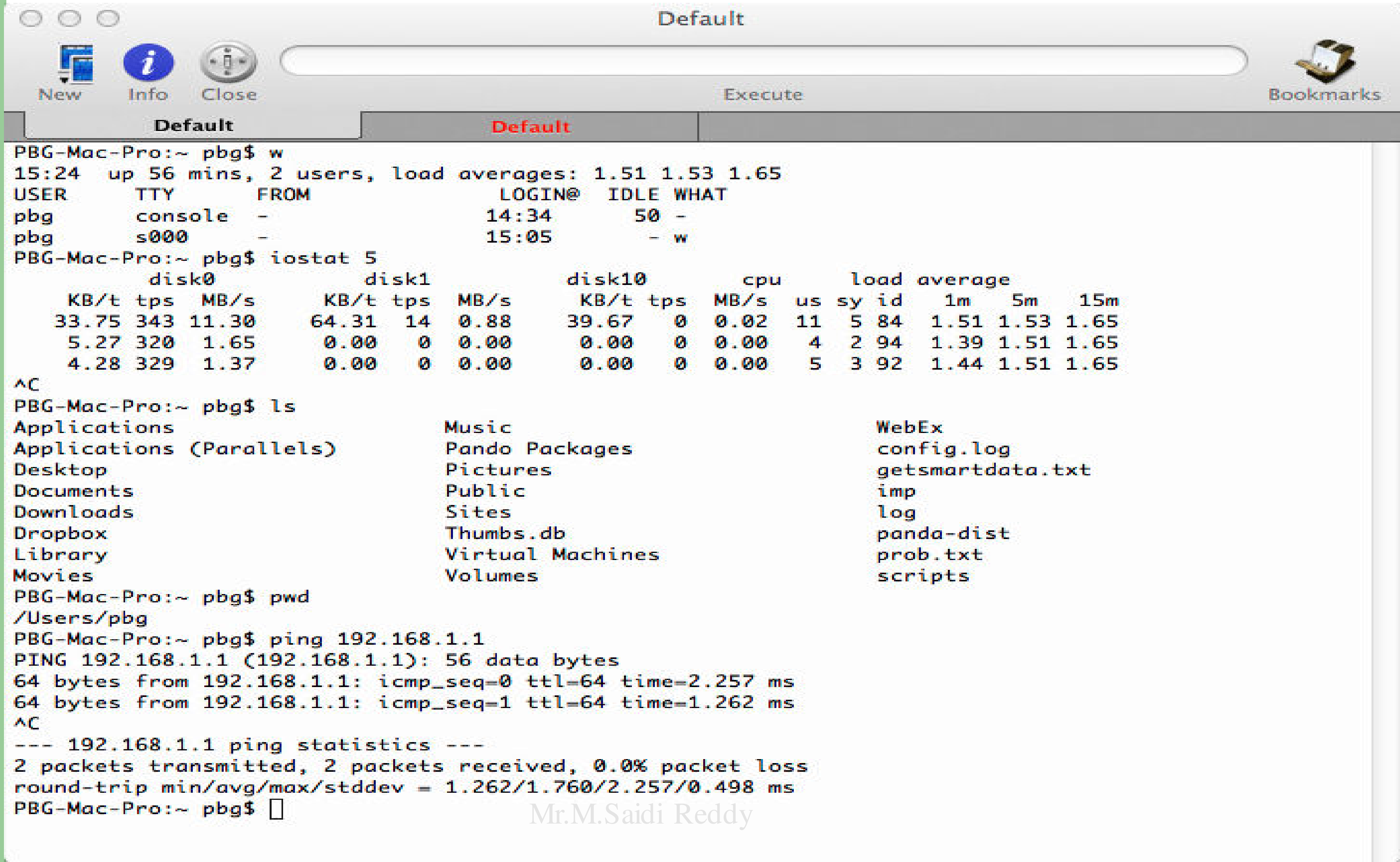


CLI is the User OS Interface

CLI allows direct command entry:

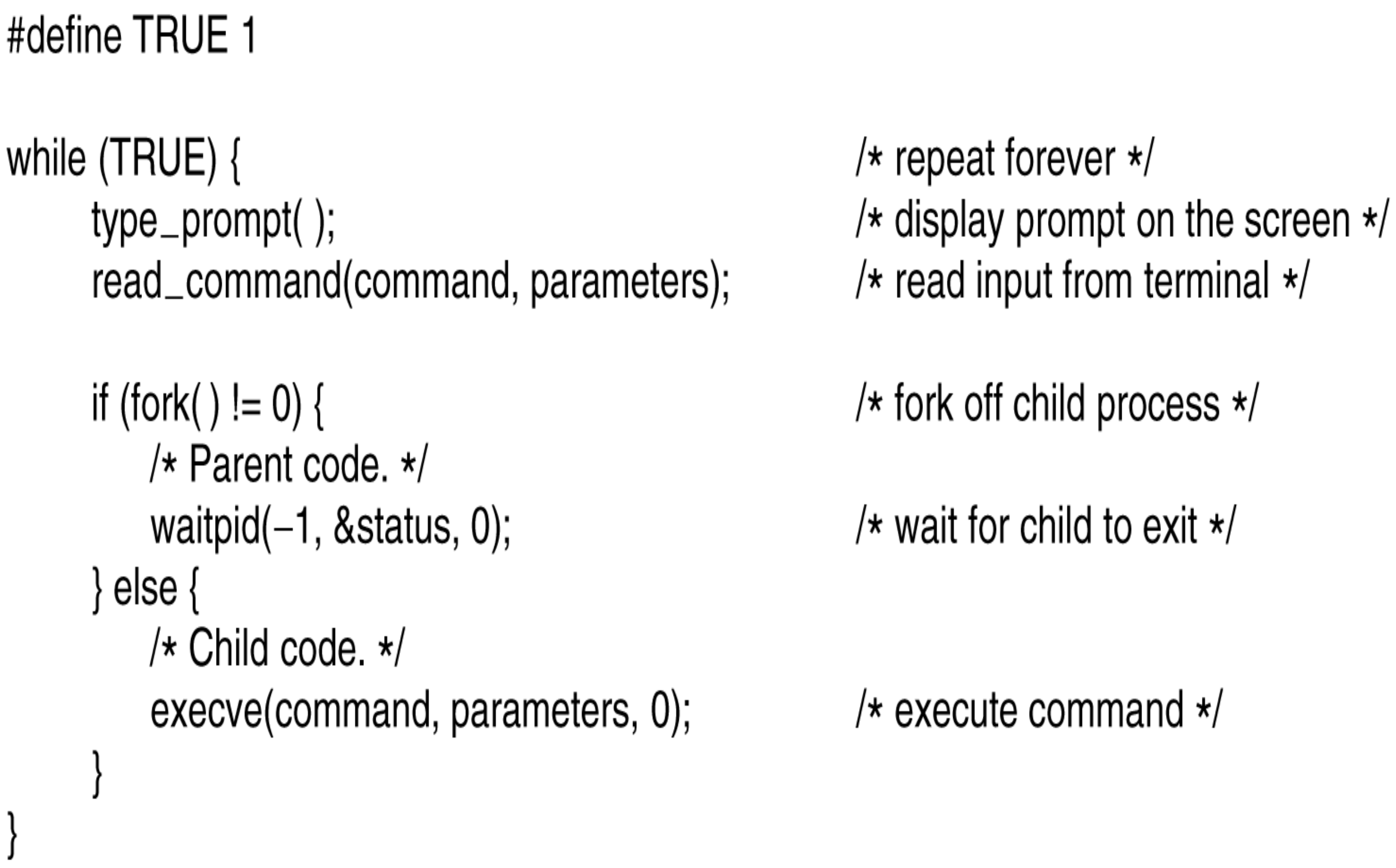
- Sometimes implemented in kernel, sometimes by systems program.
- Sometimes multiple flavors implemented – shells.
- Primarily fetches a command from user and executes it.
- Sometimes commands built-in, sometimes just names of programs; if the latter, adding new features doesn't require shell modification.

Bourne Shell (bsh)



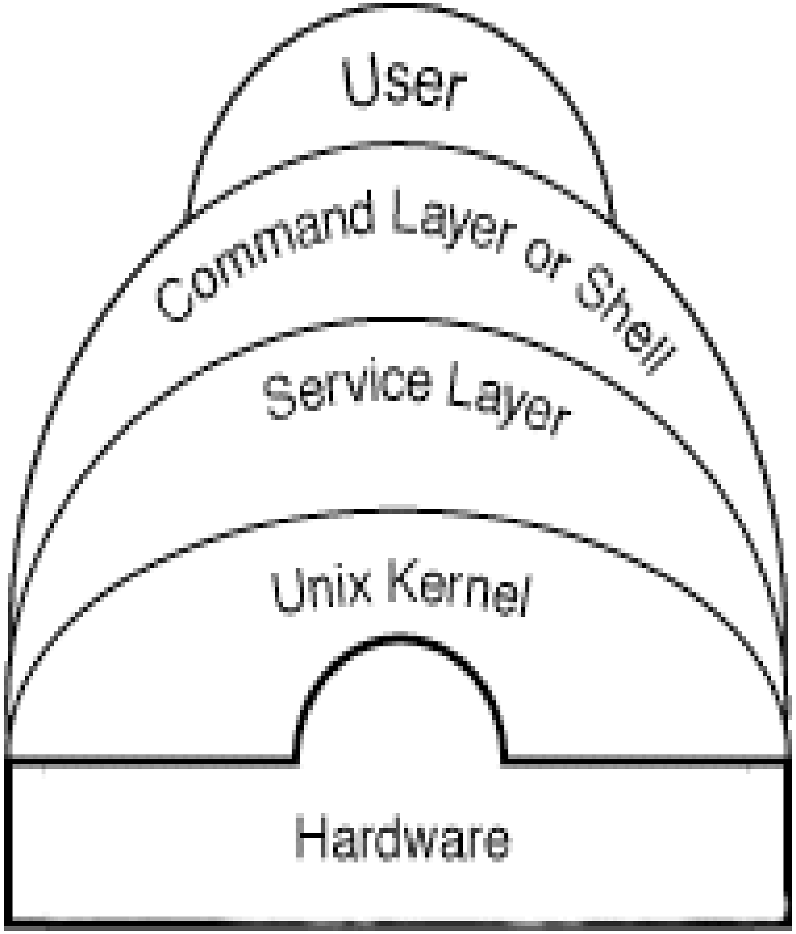
```
PBG-Mac-Pro:~ pbg$ w
15:24  up 56 mins, 2 users, load averages: 1.51 1.53 1.65
USER     TTY      FROM              LOGIN@  IDLE WHAT
pbg      console  -                14:34      50 -
pbg      s000     -                15:05       - w
PBG-Mac-Pro:~ pbg$ iostat 5
          disk0           disk1          disk10       cpu     load average
    KB/t tps  MB/s     KB/t tps  MB/s     KB/t tps  MB/s  us sy id   1m   5m   15m
   33.75 343 11.30    64.31  14  0.88    39.67   0  0.02  11  5 84  1.51 1.53 1.65
    5.27 320  1.65     0.00   0  0.00     0.00   0  0.00   4  2 94  1.39 1.51 1.65
    4.28 329  1.37     0.00   0  0.00     0.00   0  0.00   5  3 92  1.44 1.51 1.65
^C
PBG-Mac-Pro:~ pbg$ ls
Applications                    Music                           WebEx
Applications (Parallels)        Pando Packages                  config.log
Desktop                         Pictures                        getsmartdata.txt
Documents                       Public                          imp
Downloads                       Sites                           log
Dropbox                         Thumbs.db                       panda-dist
Library                         Virtual Machines                prob.txt
Movies                          Volumes                         scripts
PBG-Mac-Pro:~ pbg$ pwd
/Users/pbg
PBG-Mac-Pro:~ pbg$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=2.257 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=1.262 ms
^C
--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 1.262/1.760/2.257/0.498 ms
PBG-Mac-Pro:~ pbg$
```

A very simplified Shell

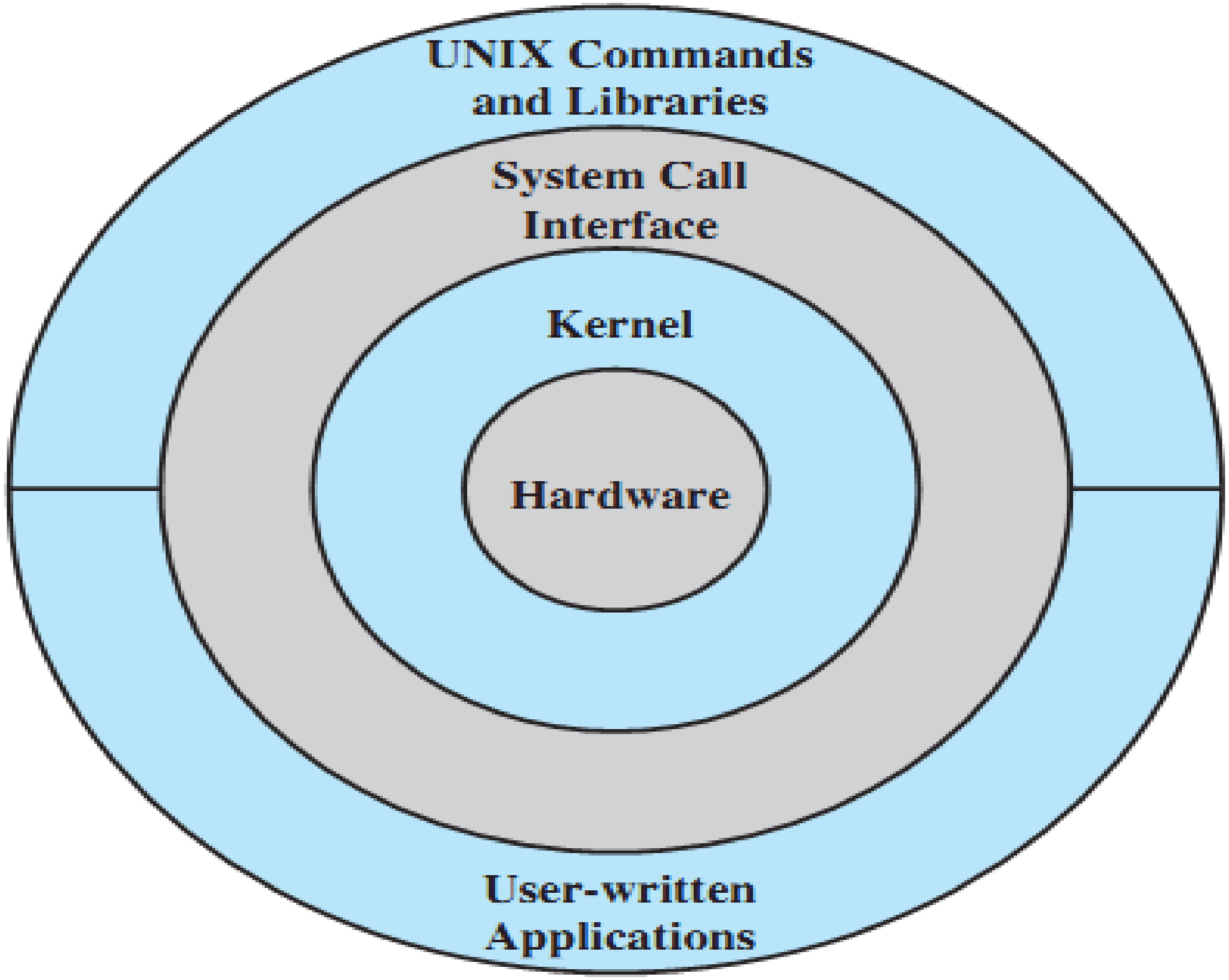


```
#define TRUE 1

while (TRUE) {                                    /* repeat forever */
    type_prompt( );                               /* display prompt on the screen */
    read_command(command, parameters);            /* read input from terminal */

    if (fork( ) != 0) {                           /* fork off child process */
        /* Parent code. */
        waitpid(−1, &status, 0);                  /* wait for child to exit */
    } else {
        /* Child code. */
        execve(command, parameters, 0);           /* execute command */
    }
}
```

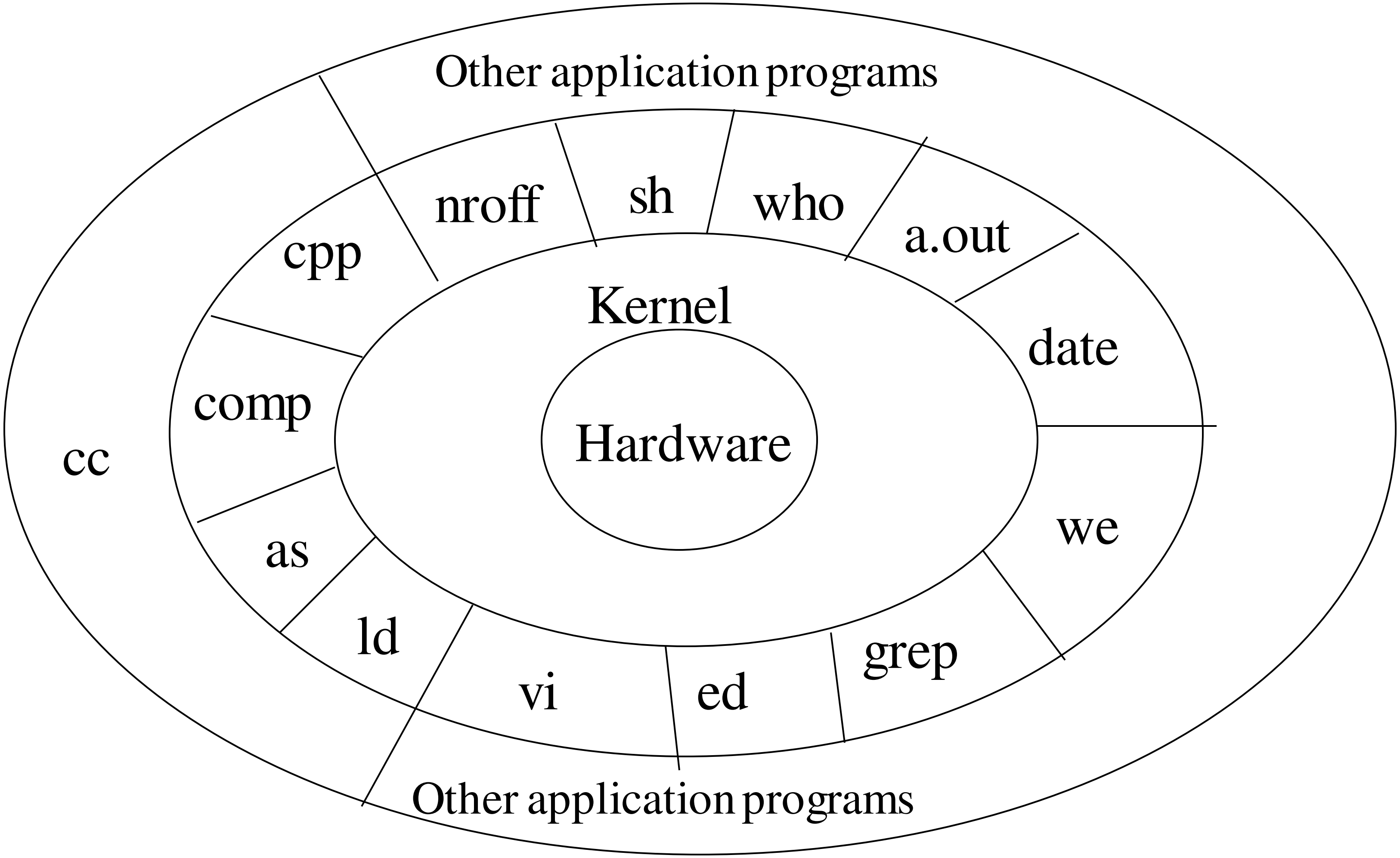
UNIX System Layout



General UNIX Architecture (1)



General UNIX Architecture (2)


Other application programs
nroff
sh
who
a.out
cpp
Kernel
date
comp
cc
Hardware
as
we
ld
grep
vi
ed
Other application programs