

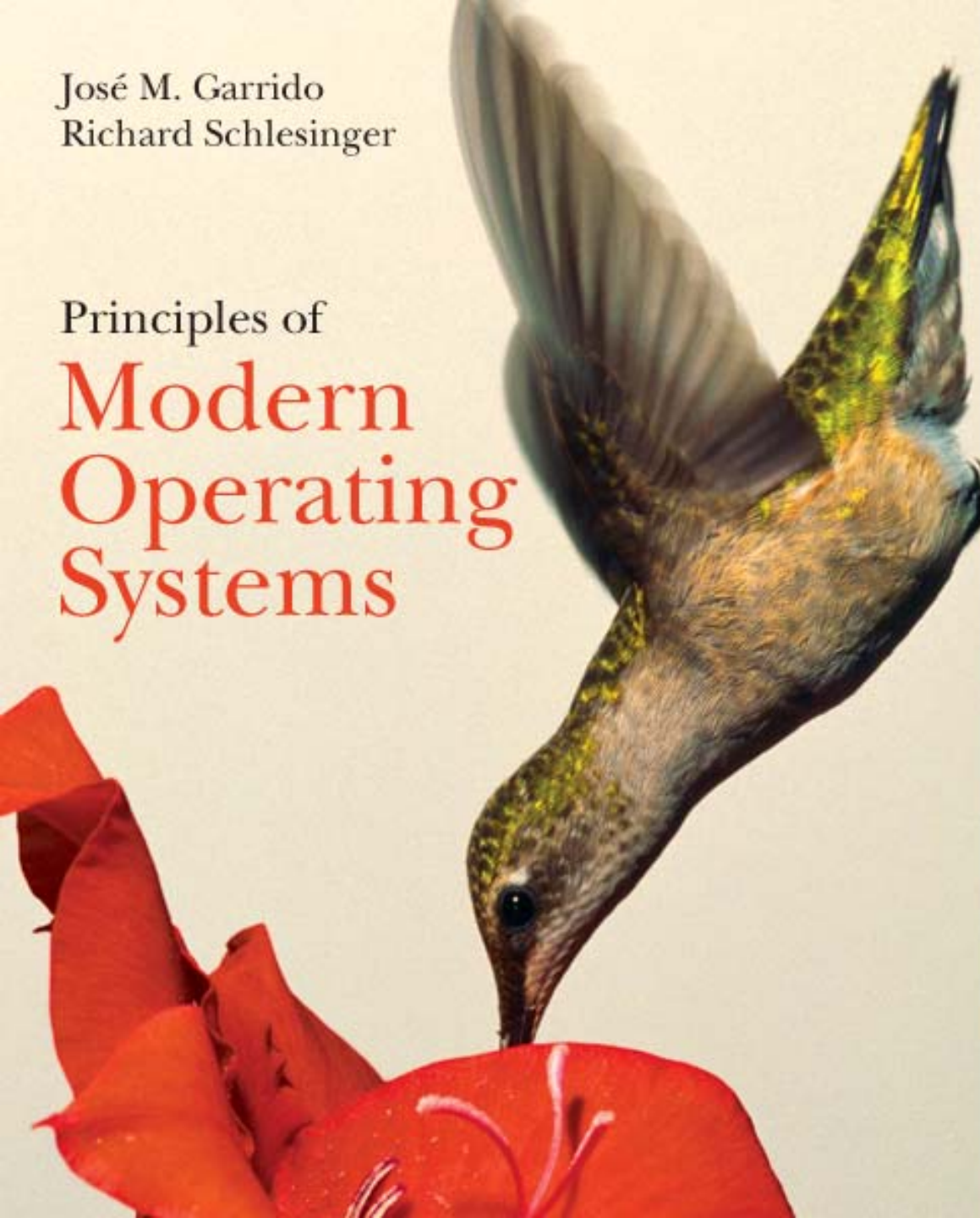
CS 3530 Operating Systems

L03 OS Intro Part 2

Dr. Ken Hoganson

José M. Garrido
Richard Schlesinger

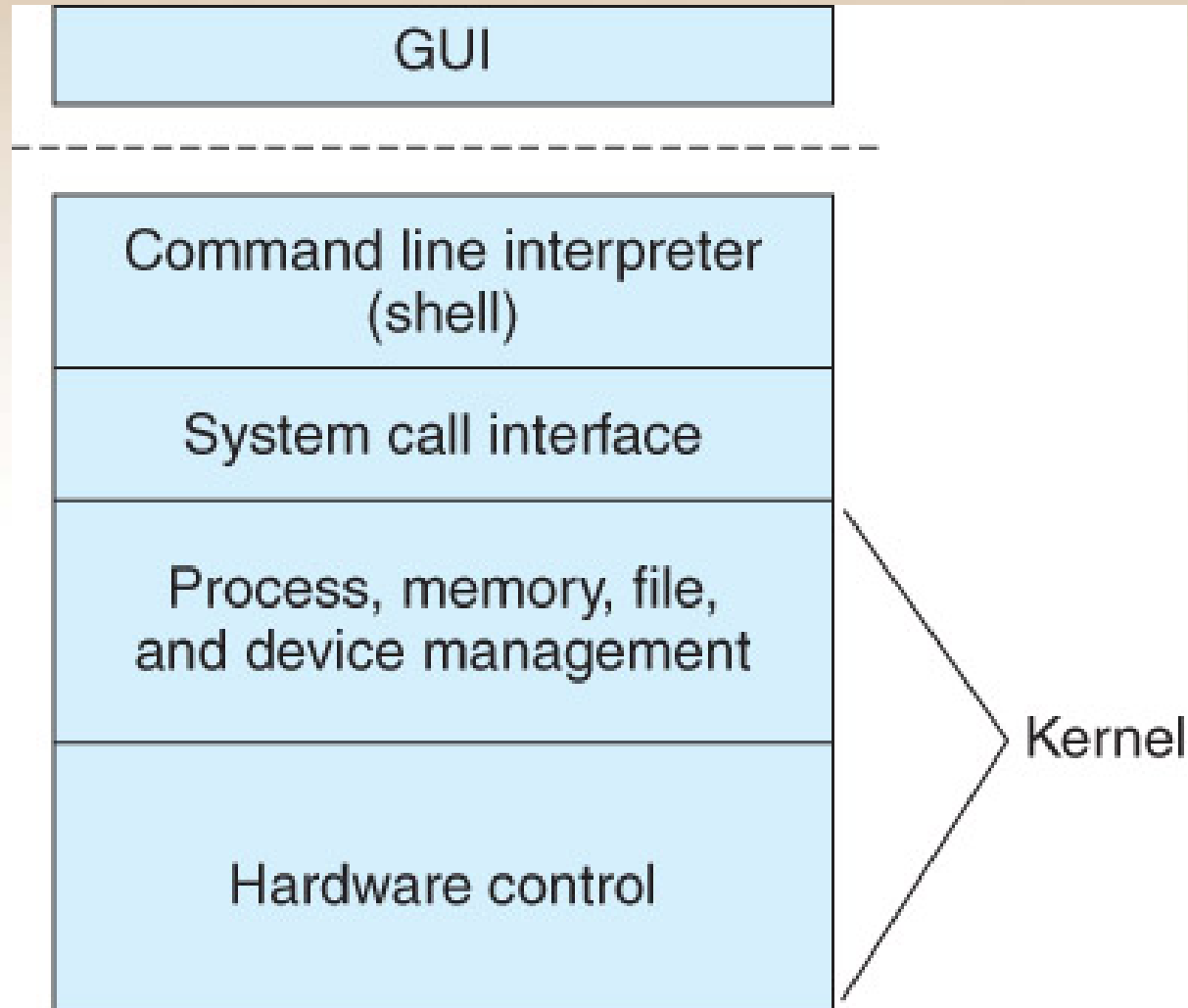
Principles of
**Modern
Operating
Systems**



Chapter 1

Basic Concepts of Operating Systems

Basic Structure of an OS



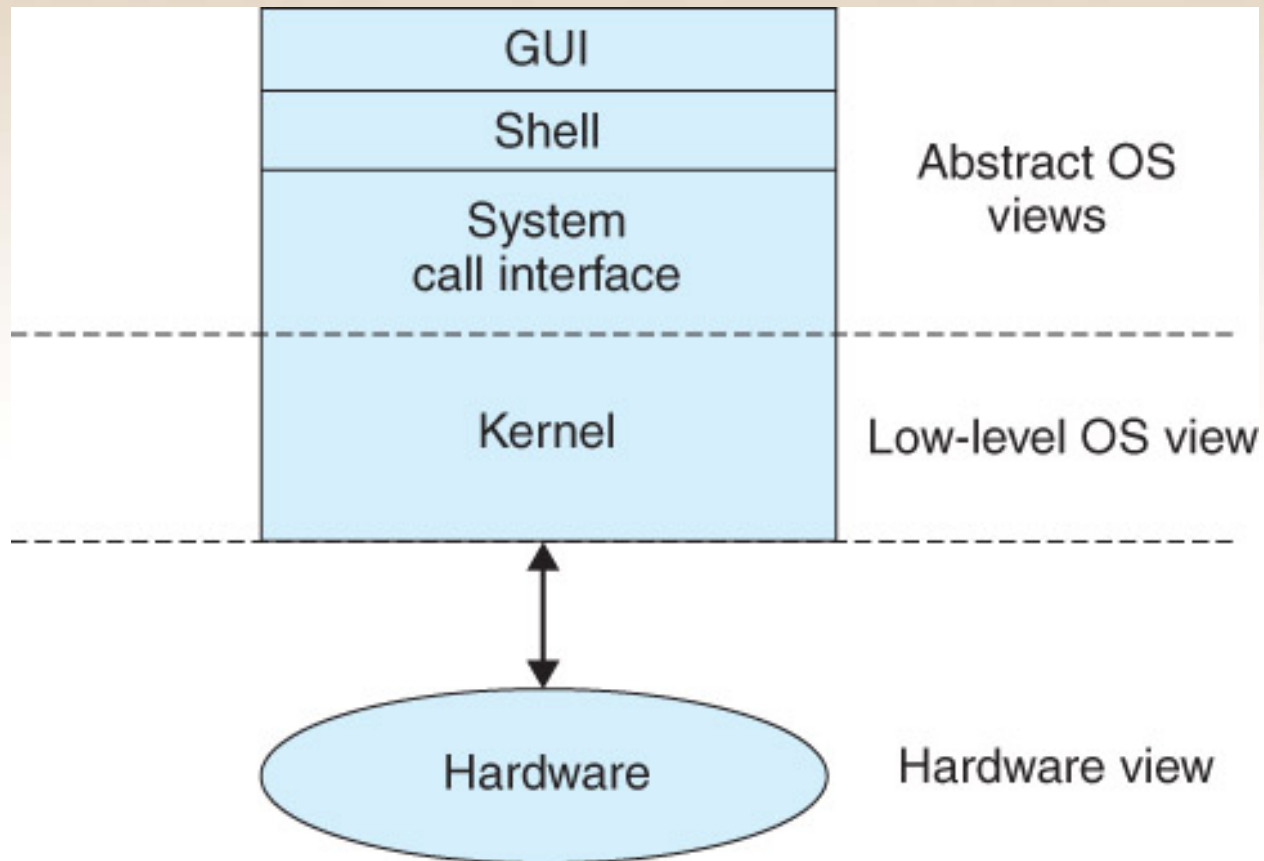
Abstract Views

- The overall structure of an operating system is divided into the various software components using a top-down (layered) approach.
- The top layer provides the easiest interface to the human operators and users interacting with the system.
- Any layer uses the services or functions provided by the next lower layer.

Operating Systems Abstract Views

- External views
 - As an user interface of the computer system
 - As a layer of software on top of the hardware
- Internal view
 - Resource manager - It controls and manages CPU, memory, I/O devices, etc.

Abstract Views of an OS



Layered Structure of an OS

- Users (top layer)
- Application User Interface (AUI): shell, commands, application programs
- Application program Interface (API): libraries, system calls
- OS kernel

System Programs

- The Operating System media will include programs that are not part of the operating system kernel.
- Examples
 - Web Browser
 - Email program
- Most users' view of the Operating System is defined by System Programs, not the OS itself

Internal View of an Operating System

- The system the system call interface separates the kernel from the application layer and the kernel is located above the hardware
- The kernel is the core, and most critical part, of the operating system and needs to be always resident in memory.
- A detailed knowledge about the different components including these lower-level components of the operating system, corresponds to an internal view of the system.

Functional Components of an OS

The most important components of an operating system are:

- Process manager
- Memory manager
- Resource manager
- File manager
- Device manager

Services Provided by the OS

- Process Control, execution, scheduling, etc.
- Communication between processes
- File Manipulation
- Device Manipulation
- Information Maintenance
- Memory Management

Jobs and Processes

A job is a unit of work submitted by a user to the operating system. A typical job consists of the parts listed below:

- A sequence of commands to the operating system
- A program either in a source language or in binary form
- A set of input data used by the program when it executes

A process basically refers to an execution instance of a program.

History of Operating Systems

- First generation - No operating system, bare hardware, machine language.
- Second generation
 - Batch systems, assemblers, linkers, loaders, compilers
 - Batch systems with Automatic Job Sequencing

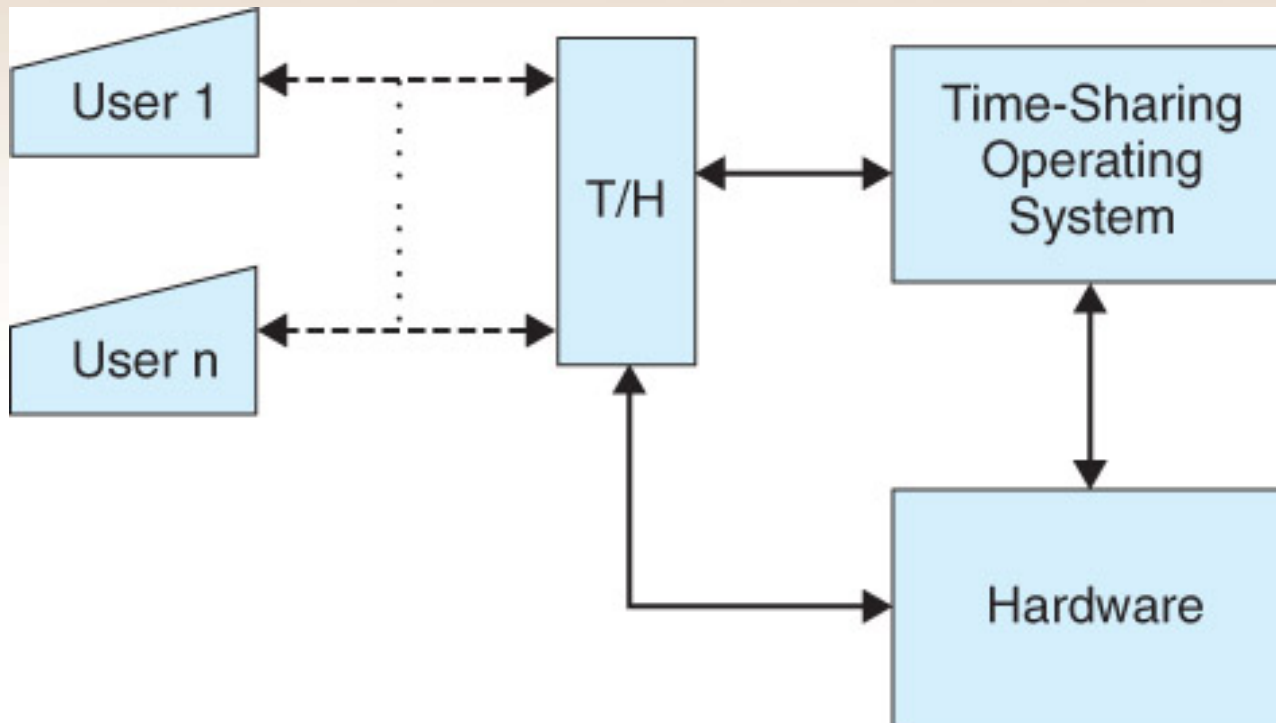
History of Operating Systems(2)

- Third generation -- O.S. for complete families of computers (OS/360)
 - Batch with Multiprogramming
 - Spooling
 - Timesharing (MULTICS, UNIX, ...)
- Fourth generation
 - Network and distributed operating systems

Categories of Operating Systems

- Batch systems, in which a set of jobs are submitted in sequence for processing.
- Interactive systems, which support computing for on-line users. The most common type of operating systems that support interactive computing is time-sharing, which are multi-user systems.
- Real-time systems, which support application programs with very tight timing constraints.
- Hybrid systems, which support batch and interactive computing.

A Time-Sharing System



Modern Operating Systems

- Windows (Microsoft Corporation) these include a family of systems: 98, Me, CE, 2000, XP, Vista, and others
- Linux (Linus Torvalds, OSF GNU)
- OSF-1 (OSF, DEC)
- Solaris (Sun Microsystems)
- IRIX (Silicon Graphics)
- OS2 (IBM)
- OS/390 (IBM)
- VMS (Dec/Compaq/HP)
- MacOS (Apple)

Mechanisms and Policies

- Mechanisms determine the implementation of some technique, policies decide what type of service is provided.
- The separation of policy from mechanism is a very important principle, it allows maximum flexibility if policy decisions are to be changed later.

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.

End of Lecture

End

Of

Today's

Lecture.