AGH UNIVERSITY OF SCIENCE
AND TECHNOLOGY

Operating systems (2)

Department of Computer Science
Faculty of Computer Science, Electronics and Telecommunications
AGH University of Science and Technology, Krakow, POLAND

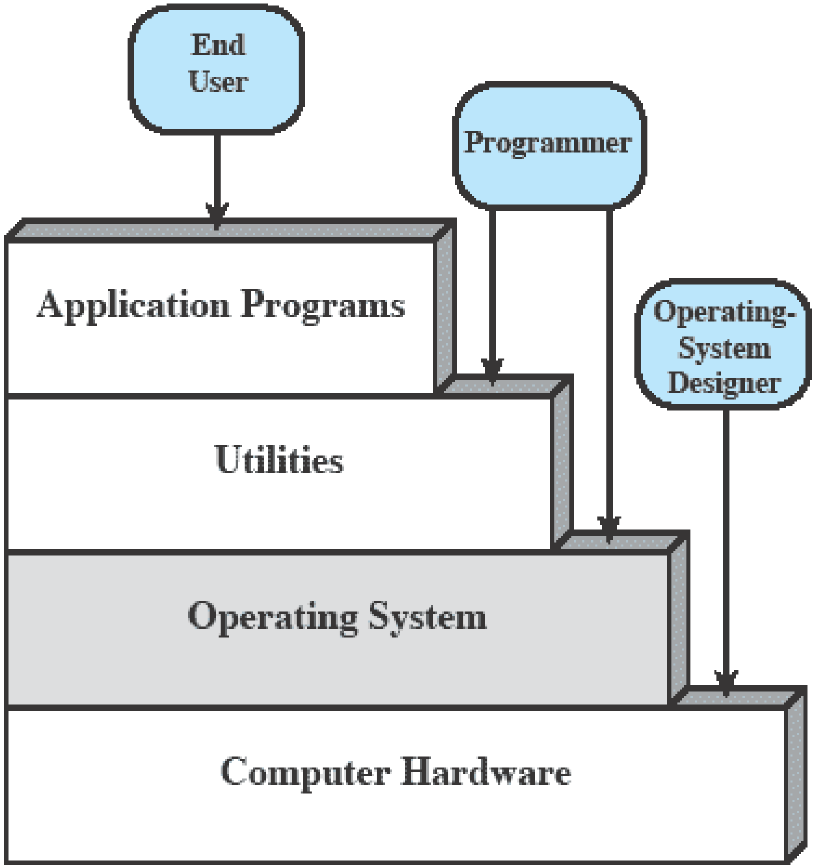
Roadmap

- Operating System Objectives/Functions
- The Evolution of Operating Systems
- Major Achievements
- Developments Leading to Modern Operating Systems
- Microsoft Windows Overview
- UNIX Systems
- Linux

Operating System

- A program that controls the execution of application programs
- An interface between applications and hardware
- Main objectives of an OS:
  - Convenience
  - Efficiency
  - Ability to evolve

Layers and Views



Services Provided by the Operating System

- Program development
  - Editors and debuggers.
- Program execution
  - OS handles scheduling of numerous tasks required to execute a program
- Access I/O devices
  - Each device will have unique interface
  - OS presents standard interface to users

Services cont…

- Controlled access to files
  - Accessing different media but presenting a common interface to users
  - Provides protection in multi-access systems
- System access
  - Controls access to the system and its resources

Services cont…

- Error detection and response
  - Internal and external hardware errors
  - Software errors
  - Operating system cannot grant request of application
- Accounting
  - Collect usage statistics
  - Monitor performance

The Role of an OS

- A computer is a set of resources for the movement, storage, and processing of data.
- The OS is responsible for managing these resources.

Operating System as Software

- The OS functions in the same way as an ordinary computer software
  - It is a program that is executed by the CPU
- Operating system relinquishes control of the processor

OS as
Resource Manager

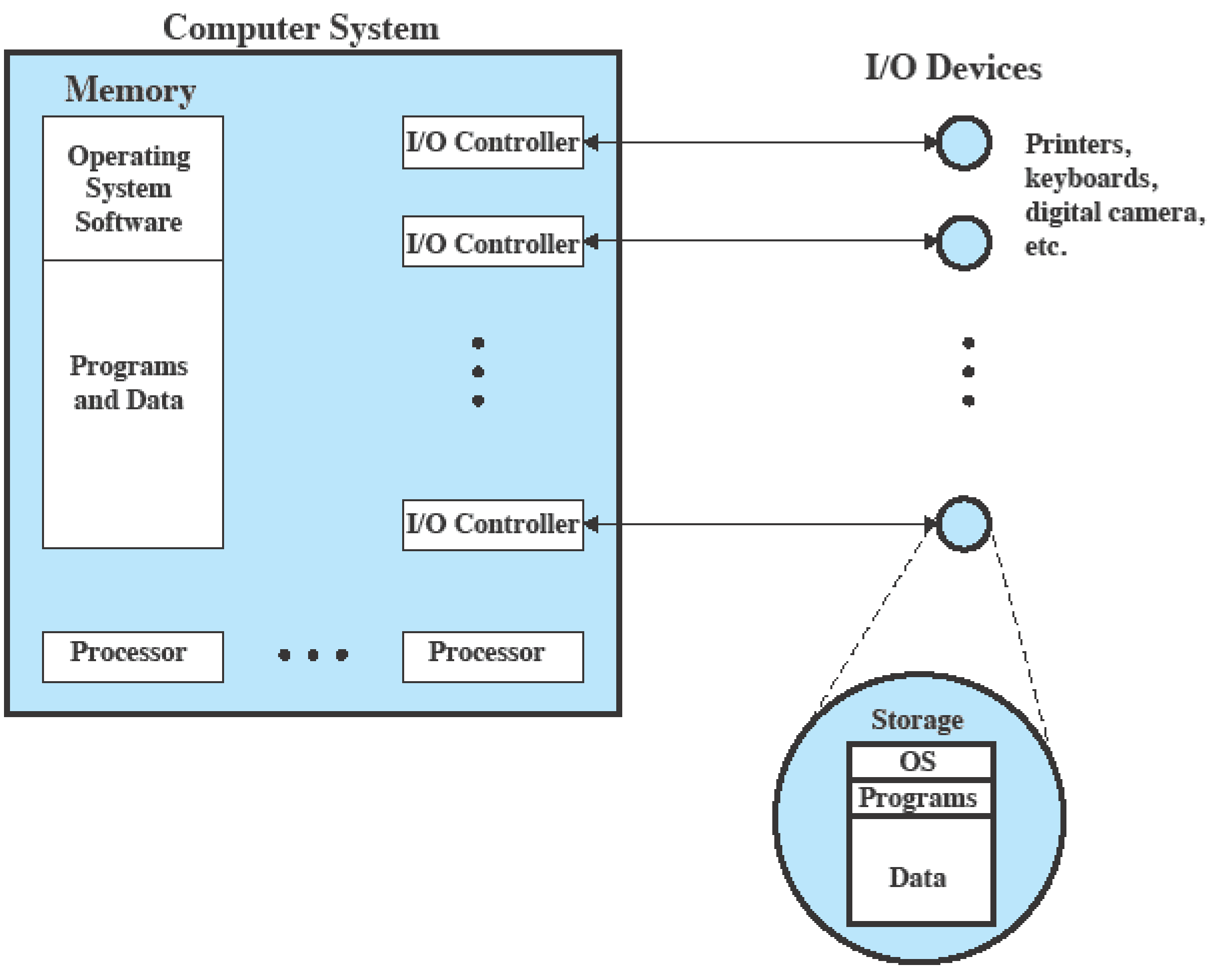


Figure 2.2 The Operating System as Resource Manager

Evolution of Operating Systems

- Operating systems will evolve over time
  - Hardware upgrades plus new types of hardware
  - New services
  - Fixes

Roadmap

- Operating System Objectives/Functions
- **The Evolution of Operating Systems**
- Major Achievements
- Developments Leading to Modern Operating Systems
- Microsoft Windows Overview
- UNIX Systems
- Linux

Evolution of Operating Systems

- It may be easier to understand the key requirements of an OS by considering the evolution of Operating Systems
- Stages include
  - Serial Processing
  - Simple Batch Systems
  - Multiprogrammed batch systems
  - Time Sharing Systems

Serial Processing

- No operating system
- Machines run from a console with display lights, toggle switches, input device, and printer
- Problems include:
  - Scheduling
  - Setup time

Simple batch system

- Early computers were extremely expensive
  - Important to maximize processor utilization
- Monitor
  - Software that controls the sequence of events
  - Batch jobs together
  - Program returns control to monitor when finished

Monitor's perspective

- Monitor controls the sequence of events
- *Resident Monitor* is software always in memory
- Monitor reads in job and gives control
- Job returns control to monitor

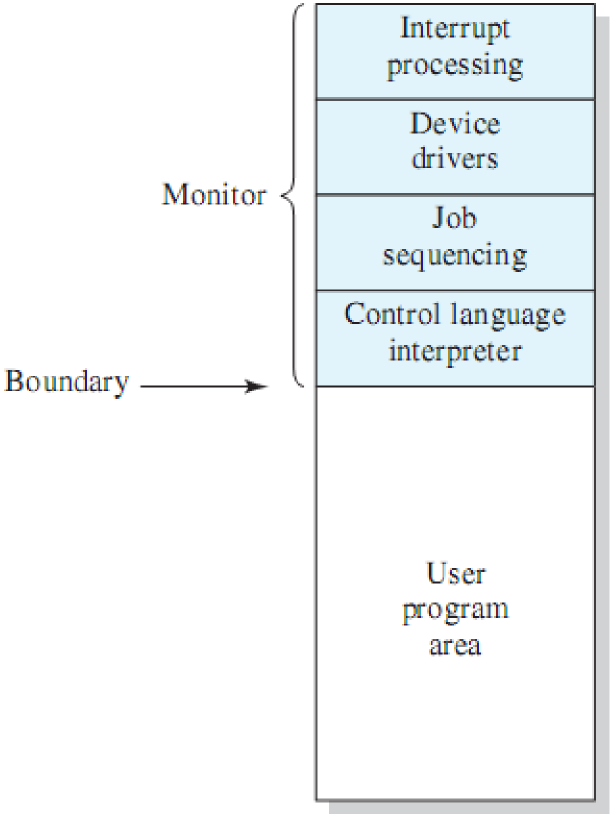


Figure 2.3 Memory Layout for a Resident Monitor

Job Control Language

- Special type of programming language to control jobs
- Provides instruction to the monitor
  - What compiler to use
  - What data to use

Desirable Hardware Features

- Memory protection for monitor
  - Jobs cannot overwrite or alter
- Timer
  - Prevent a job from monopolizing system
- Privileged instructions
  - Only executed by the monitor
- Interrupts

Modes of Operation

- User Mode
  - User program executes in user mode
  - Certain areas of memory protected from user access
  - Certain instructions may not be executed
- Kernel Mode
  - Monitor executes in kernel mode
  - Privileged instructions may be executed, all memory accessible.

Multiprogrammed Batch Systems

- CPU is often idle
  - Even with automatic job sequencing.
  - I/O devices are slow compared to processor

Read one record from file	15 μs
Execute 100 instructions	1 μs
Write one record to file	15 μs
TOTAL	31 μs

$$\text{Percent CPU Utilization} = \frac{1}{31} = 0.032 = 3.2\%$$

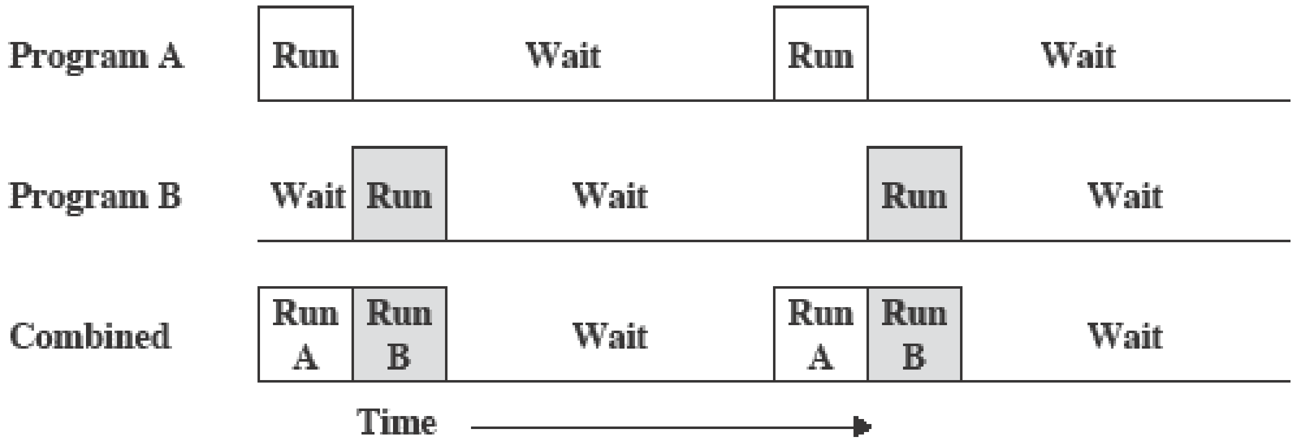
Uniprogramming

- Processor must wait for I/O instruction to complete before preceding

(a) Uniprogramming

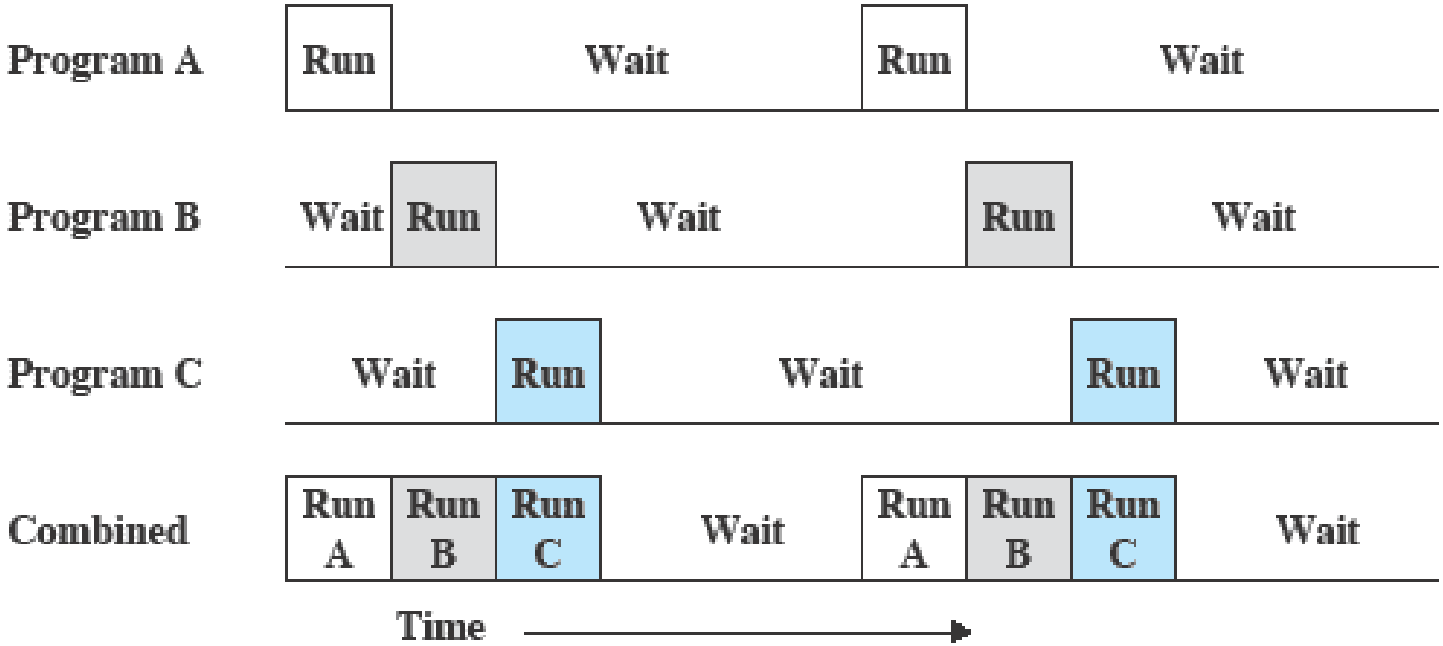
Multiprogramming

- When one job needs to wait for I/O, the processor can switch to the other job



(b) Multiprogramming with two programs

Multiprogramming



(c) Multiprogramming with three programs

Example

Table 2.1 Sample Program Execution Attributes

	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min	15 min	10 min
Memory required	50 M	100 M	75 M
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes

Utilization Histograms

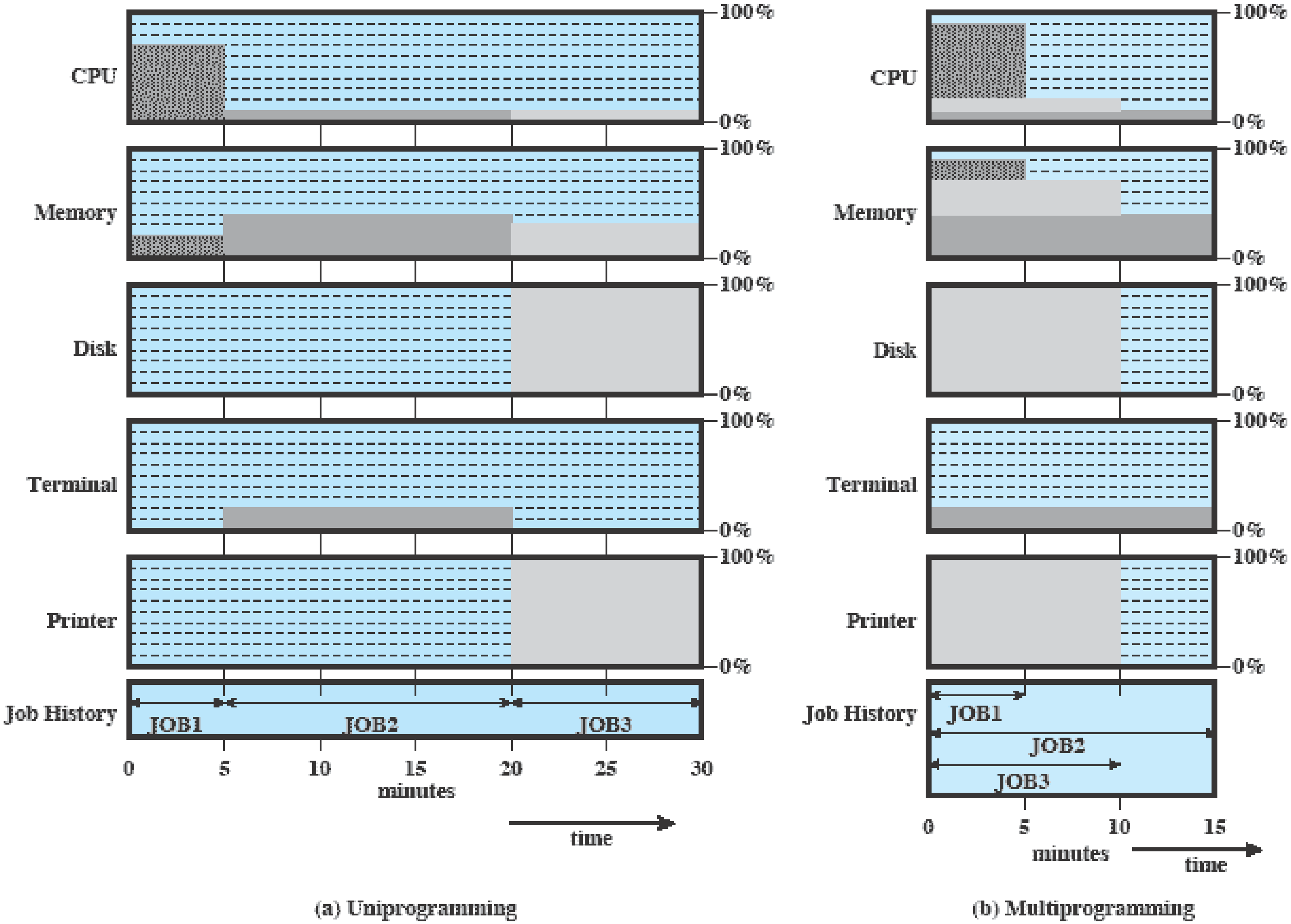


Figure 2.6 Utilization Histograms

Time Sharing Systems

- Using multiprogramming to handle multiple interactive jobs
- Processor’s time is shared among multiple users
- Multiple users simultaneously access the system through terminals

Batch Multiprogramming vs. Time Sharing

	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

Problems and Issues

- Multiple jobs in memory must be protected from each other’s data
- File system must be protected so that only authorised users can access
- Contention for resources must be handled
  - Printers, storage etc

Roadmap

- Operating System Objectives/Functions
- The Evolution of Operating Systems
- **Major Achievements**
- Developments Leading to Modern Operating Systems
- Microsoft Windows Overview
- UNIX Systems
- Linux

Major Advances

- Operating Systems are among the most complex pieces of software ever developed
- Major advances include:
  - Processes
  - Memory management
  - Information protection and security
  - Scheduling and resource management
  - System

Process

- Fundamental to the structure of OS’s
- A *process* is:
  - A program in execution
  - An instance of a running program
  - The entity that can be assigned to and executed on a processor
  - A single sequential thread of execution, a current state, and an associated set of system resources.

Causes of Errors when Designing System Software

- Error in designing an OS are often subtle and difficult to diagnose
- Errors typically include:
  - Improper synchronization
  - Failed mutual exclusion
  - Non-determinate program operation
  - Deadlocks

Components of a Process

- A process consists of
  - An executable program
  - Associated data needed by the program
  - Execution context of the program (or “process state”)
- The execution context contains all information the operating system needs to manage the process

Process Management

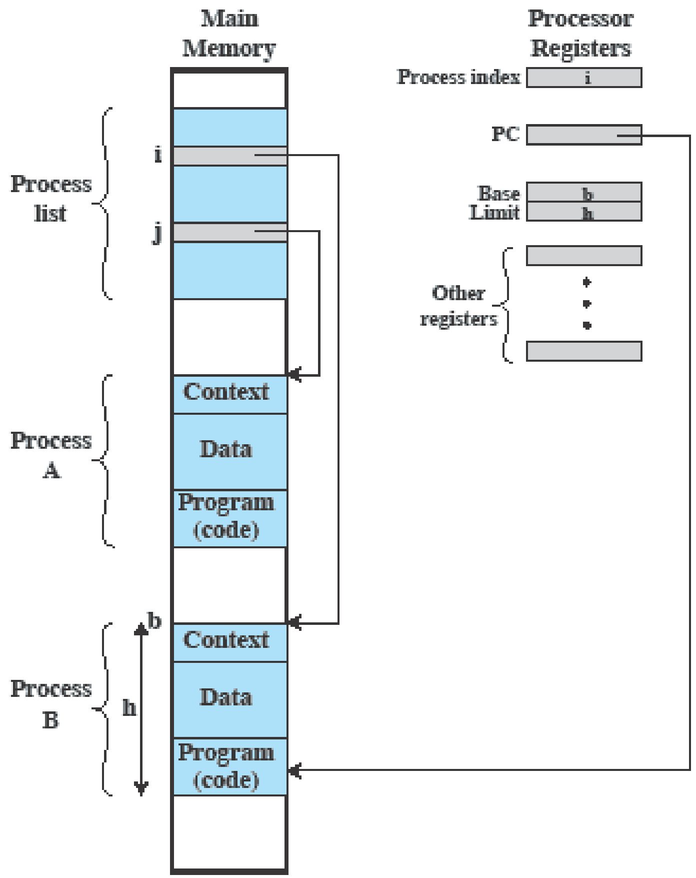


Figure 2.8 Typical Process Implementation

Memory Management

- The OS has 5 principal storage management responsibilities
  - Process isolation
  - Automatic allocation and management
  - Support of modular programming
  - Protection and access control
  - Long-term storage

Virtual Memory

- File system implements long-term store
- Virtual memory allows programs to address memory from a logical point of view
  - Without regard to the limits of physical memory

Paging

- Allows process to be comprised of a number of fixed-size blocks, called pages
- Virtual address is a page number and an offset within the page
- Each page may be located any where in main memory

Virtual Memory

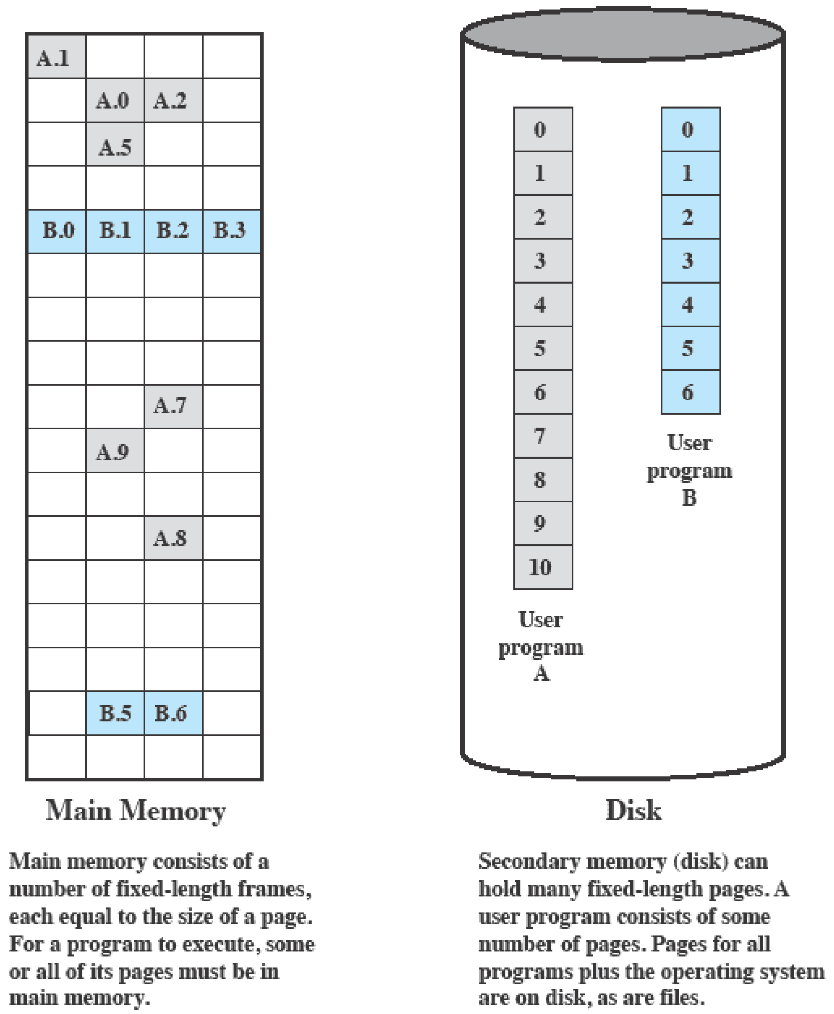
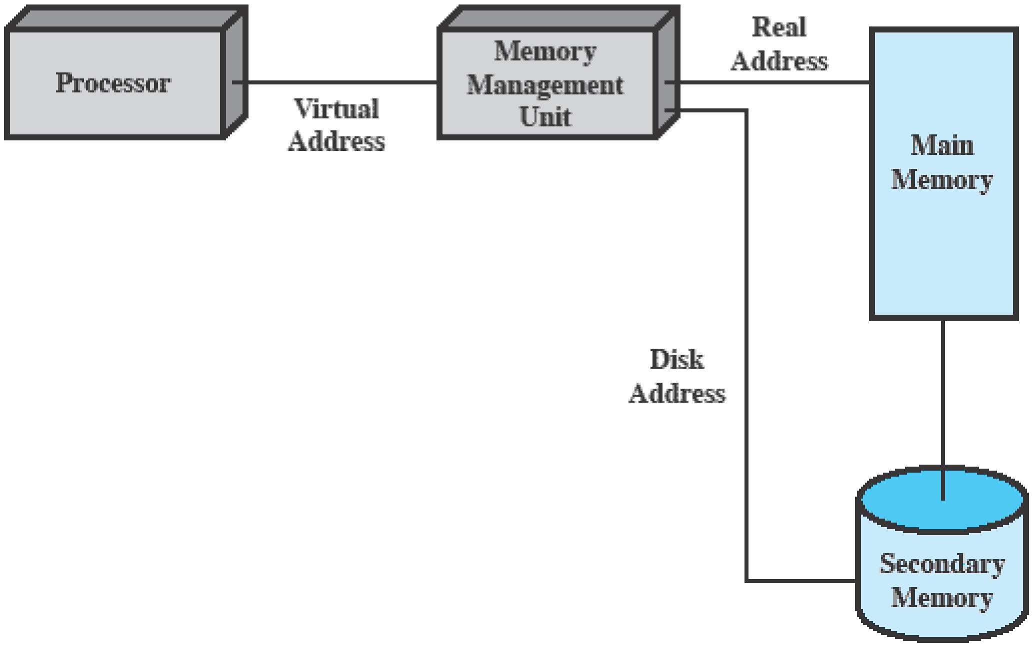


Figure 2.9 Virtual Memory Concepts

Virtual Memory Addressing



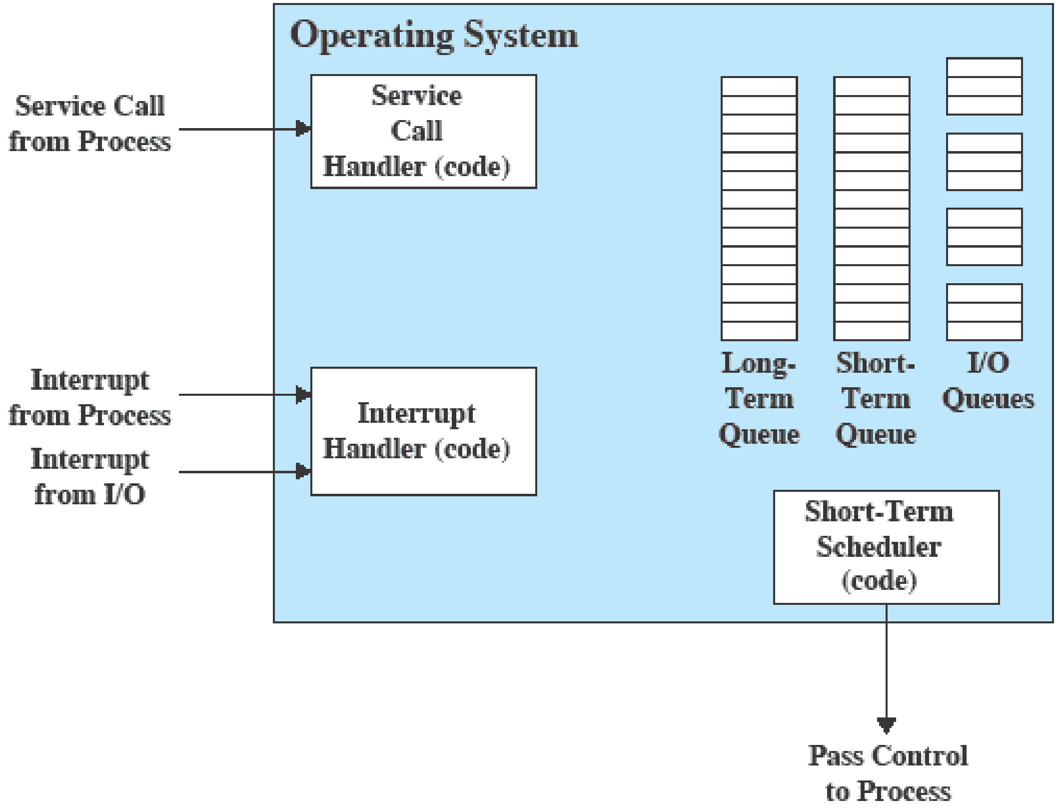
Information Protection and Security

- The problem involves controlling access to computer systems and the information stored in them.
- Main issues are:
  - Availability
  - Confidentiality
  - Data integrity
  - Authenticity

Scheduling and Resource Management

- Key responsibility of an OS is managing resources
- Resource allocation policies must consider:
  - Fairness
  - Differential responsiveness
  - Efficiency

Key Elements of an Operating System



System Structure

- View the system as a series of levels
- Each level performs a related subset of functions
- Each level relies on the next lower level to perform more primitive functions
- This decomposes a problem into a number of more manageable subproblems

OS Design Hierarchy

Level	Name	Objects	Example Operations
13	Shell	User programming environment	Statements in shell language
12	User processes	User processes	Quit, kill, suspend, resume
11	Directories	Directories	Create, destroy, attach, detach, search, list
10	Devices	External devices, such as printers, displays, and keyboards	Open, close, read, write
9	File system	Files	Create, destroy, open, close, read, write
8	Communications	Pipes	Create, destroy, open, close, read, write
7	Virtual memory	Segments, pages	Read, write, fetch
6	Local secondary store	Blocks of data, device channels	Read, write, allocate, free
5	Primitive processes	Primitive processes, semaphores, ready list	Suspend, resume, wait, signal
4	Interrupts	Interrupt-handling programs	Invoke, mask, unmask, retry
3	Procedures	Procedures, call stack, display	Mark stack, call, return
2	Instruction set	Evaluation stack, microprogram interpreter, scalar and array data	Load, store, add, subtract, branch
1	Electronic circuits	Registers, gates, buses, etc.	Clear, transfer, activate, complement

Gray shaded area represents hardware.

Roadmap

- Operating System Objectives/Functions
- The Evolution of Operating Systems
- Major Achievements
- **Developments Leading to Modern Operating Systems**
- Microsoft Windows Overview
- UNIX Systems
- Linux

Different Architectural Approaches

- Various approaches have been tried, categories include:
  - Microkernel architecture
  - Multithreading
  - Symmetric multiprocessing
  - Distributed operating systems
  - Object-oriented design

Microkernel Architecture

- Most early OS are a monolithic kernel
  - Most OS functionality resides in the kernel.
- A microkernel assigns only a few essential functions to the kernel
  - Address spaces
  - Interprocess communication (IPC)
  - Basic scheduling

Multithreading

- Process is divided into threads that can run concurrently
- Thread
  - Dispatchable unit of work
  - executes sequentially and is interruptible
- Process is a collection of one or more threads

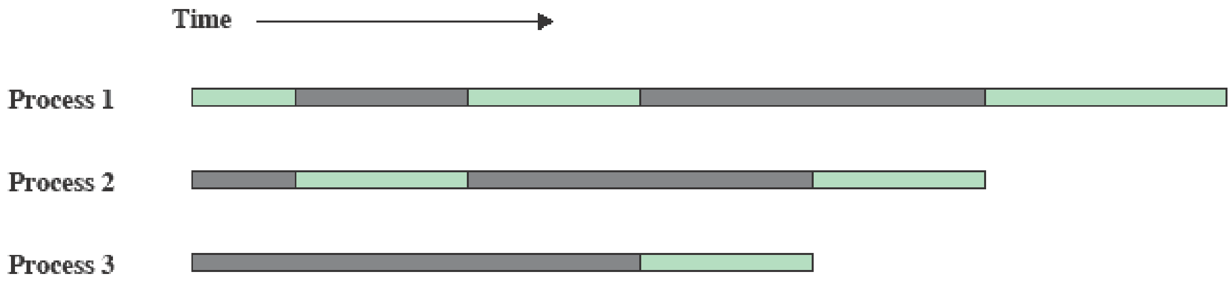
Symmetric multiprocessing (SMP)

- An SMP system has
  - multiple processors
  - These processors share same main memory and I/O facilities
  - All processors can perform the same functions
- The OS of an SMP schedules processes or threads across all of the processors.

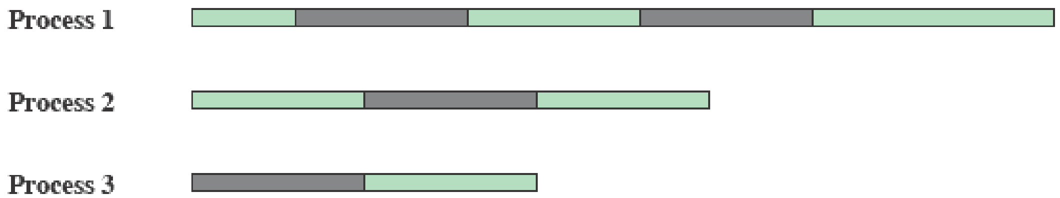
SMP Advantages

- Performance
  - Allowing parallel processing
- Availability
  - Failure of a single process does not halt the system
- Incremental Growth
  - Additional processors can be added.
- Scaling
  - Vendors can offer a range of products based on the number of processors configured in the system

Multiprogramming and Multiprocessing



(a) Interleaving (multiprogramming, one processor)



(b) Interleaving and overlapping (multiprocessing; two processors)

Distributed Operating Systems

- Provides the illusion of a **single main memory space** and a **single secondary memory space** plus other unified access facilities, such as **a distributed file system**
- State of the art for distributed operating systems lags that of uniprocessor and SMP operating systems

Object-oriented design

- Used for adding modular extensions to a small kernel
- Enables programmers to customize an operating system without disrupting system integrity
- Also eases the development of distributed tools and full-blown distributed operating systems

Fault tolerance

- Ability of a system or component to continue normal operation despite the presence of hardware or software faults
- Typically involves some degree of redundancy
- Intended to increase the reliability of a system
  - Typically comes with a cost in financial terms or performance
- The extent adoption of fault tolerance measures must be determined by how critical the resource is

Fault tolerance - Fundamental Concepts

- Basic measures are:
  - **Reliability**
    - *R(t)*
    - Defined as the probability of its correct operation up to time *t* given that the system was operating correctly at time *t=o*
  - **Mean time to failure (MTTF)**
    - Mean time to repair (MTTR) is the average time it takes to repair or replace a faulty element
  - **Availability**
    - Defined as the fraction of time the system is available to service users' requests

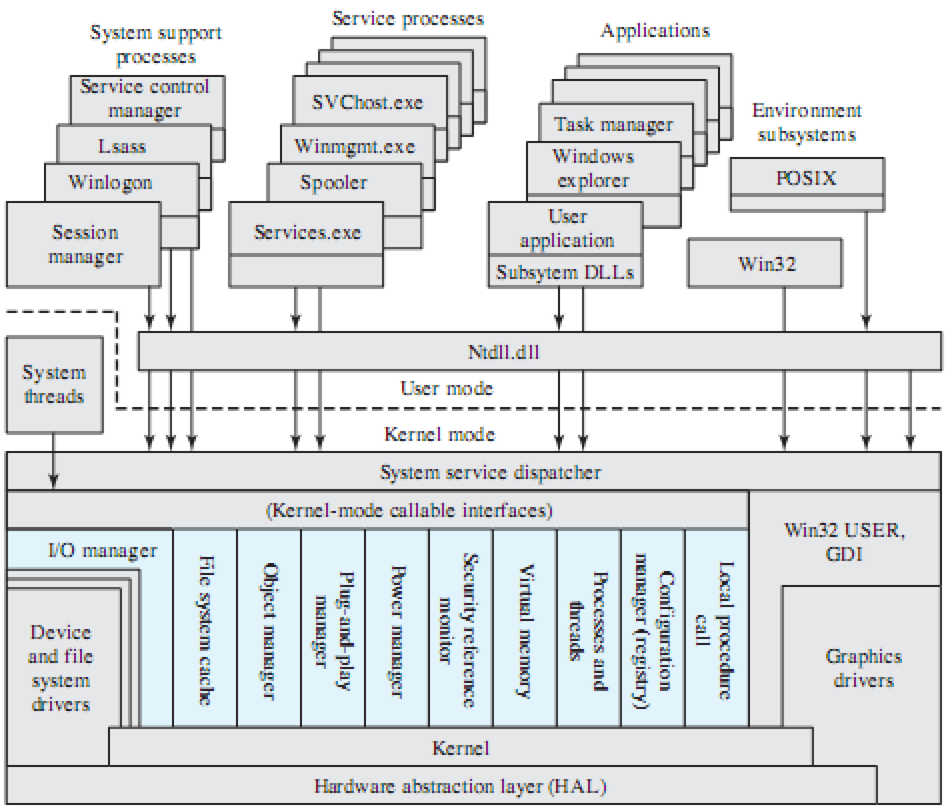
Roadmap

- Operating System Objectives/Functions
- The Evolution of Operating Systems
- Major Achievements
- Developments Leading to Modern Operating Systems
- **Microsoft Windows Overview**
- UNIX Systems
- Linux

Single-User Multitasking

- From Windows 2000 on Windows development developed to exploit modern 32-bit and 64-bit microprocessors
- Designed for single users who run multiple programs
- Main drivers are:
  - Increased memory and speed of microprocessors
  - Support for virtual memory

Windows Architecture



Lsass = local security authentication server
POSIX = portable operating system interface
GDI = graphics device interface
DLL = dynamic link libraries

Colored area indicates Executive

Figure 2.13 Windows and Windows Vista Architecture [RUSS05]

Kernel-Mode Components of Windows

- Executive
  - Contains the core OS services, such as memory management, process and thread management, security, I/O, and interprocess communication
- Kernel
  - Controls execution of the processors. The Kernel manages thread scheduling, process switching, exception and interrupt handling, and multiprocessor synchronization
- Hardware Abstraction Layer (HAL)
  - Maps between generic hardware commands and responses and those unique to a specific platform and isolates the OS from platform-specific hardware differences

Kernel-Mode Components of Windows (2)

- Device Drivers
  - Dynamic libraries that extend the functionality of the Executive. These include hardware device drivers that translate user I/O function calls into specific hardware device I/O requests and software components for implementing file systems, network protocols, and any other system extensions that need to run in kernel mode

- Windowing and Graphics System
  - Implements the GUI functions, such as dealing with windows, user interface controls, and drawing

Windows Executive

- **I/O manager**
  - Provides a framework through which I/O devices are accessible to applications, and is responsible for dispatching to the appropriate device drivers for further processing
- **Cache manager**
  - Improves the performance of file-based I/O by causing recently referenced file data to reside in main memory for quick access, and by deferring disk writes by holding the updates in memory for a short time before sending them to the disk in more efficient batches
- **Object manager**
  - Creates, manages, and deletes Windows Executive objects that are used to represent resources such as processes, threads, and synchronization objects and enforces uniform rules for retaining, naming, and setting the security of objects

Windows Executive (2)

- **Plug-and-play manager**
  - Determines which drivers are required to support a particular device and loads those drivers
- **Power manager**
  - Coordinates power management among devices
- **Security reference monitor**
  - Enforces access-validation and audit-generation rules
- **Virtual memory manager**
  - Manages virtual addresses, physical memory, and the paging files on disk and controls the memory management hardware and data structures which map virtual addresses in the process’s address space to physical pages in the computer’s memory

Windows Executive (3)

- **Process/thread manager**
  - Creates, manages, and deletes process and thread objects
- **Configuration manager**
  - Responsible for implementing and managing the system registry, which is the repository for both system-wide and per-user settings of various parameters
- **Advanced local procedure call (ALPC) facility**
  - Implements an efficient cross-process procedure call mechanism for communication between local processes implementing services and subsystems

Client/Server Model

- Windows OS, protected subsystem, and applications all use a client/server model
  - Common in distributed systems, but can be used internal to a single system
- Processes communicate via RPC

Windows Objects

- Windows draws heavily on the concepts of object-oriented design.
- Key Object Oriented concepts used by Windows are:
  - Encapsulation
  - Object class and instance

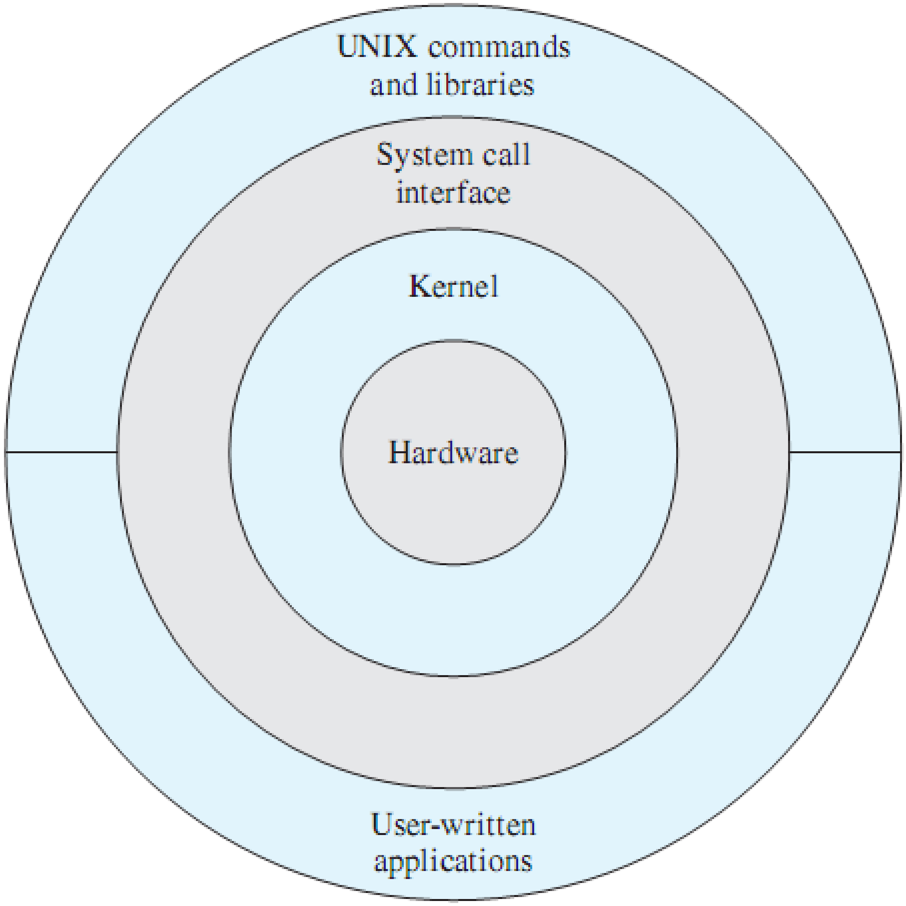
User-Mode Processes

- Windows supports four basic types of user-mode processes
- Special System Processes - User-mode services needed to manage the system
- Service Processes - The printer spooler, event logger, and user-mode components that cooperate with device drivers, and various network services
- Environment Subsystems - Provide different OS personalities (environments)
- User Applications - Executables (EXEs) and DLLs that provide the functionality users run to make use of the system

Roadmap

- Operating System Objectives/Functions
- The Evolution of Operating Systems
- Major Achievements
- Developments Leading to Modern Operating Systems
- Microsoft Windows Overview
- **UNIX Systems**
- Linux

Description of UNIX – General Unix Architecture



Traditional UNIX Kernel

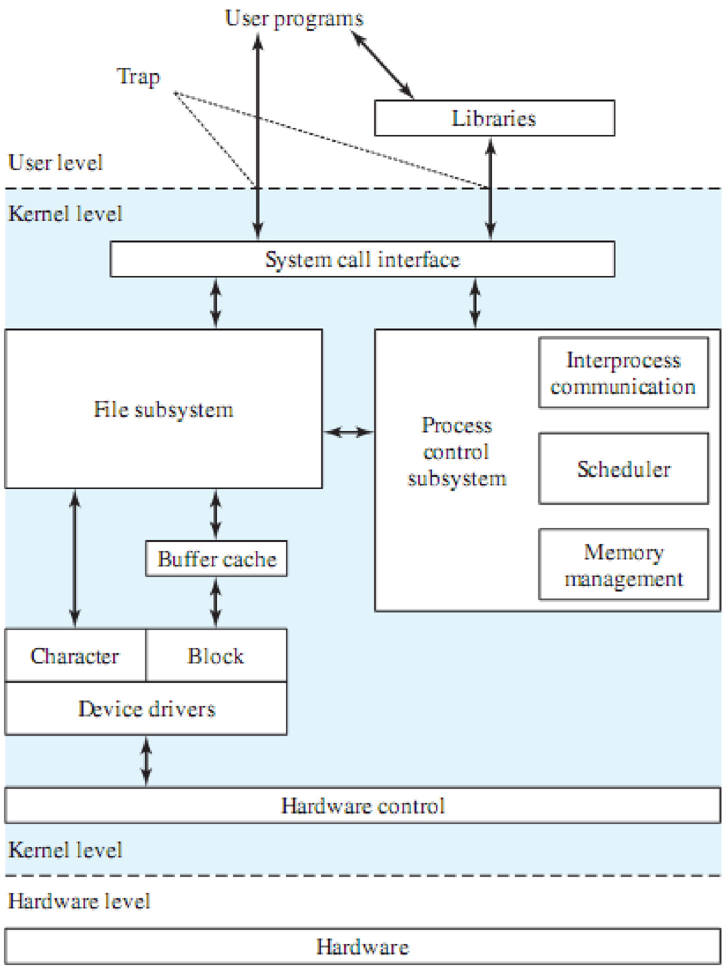
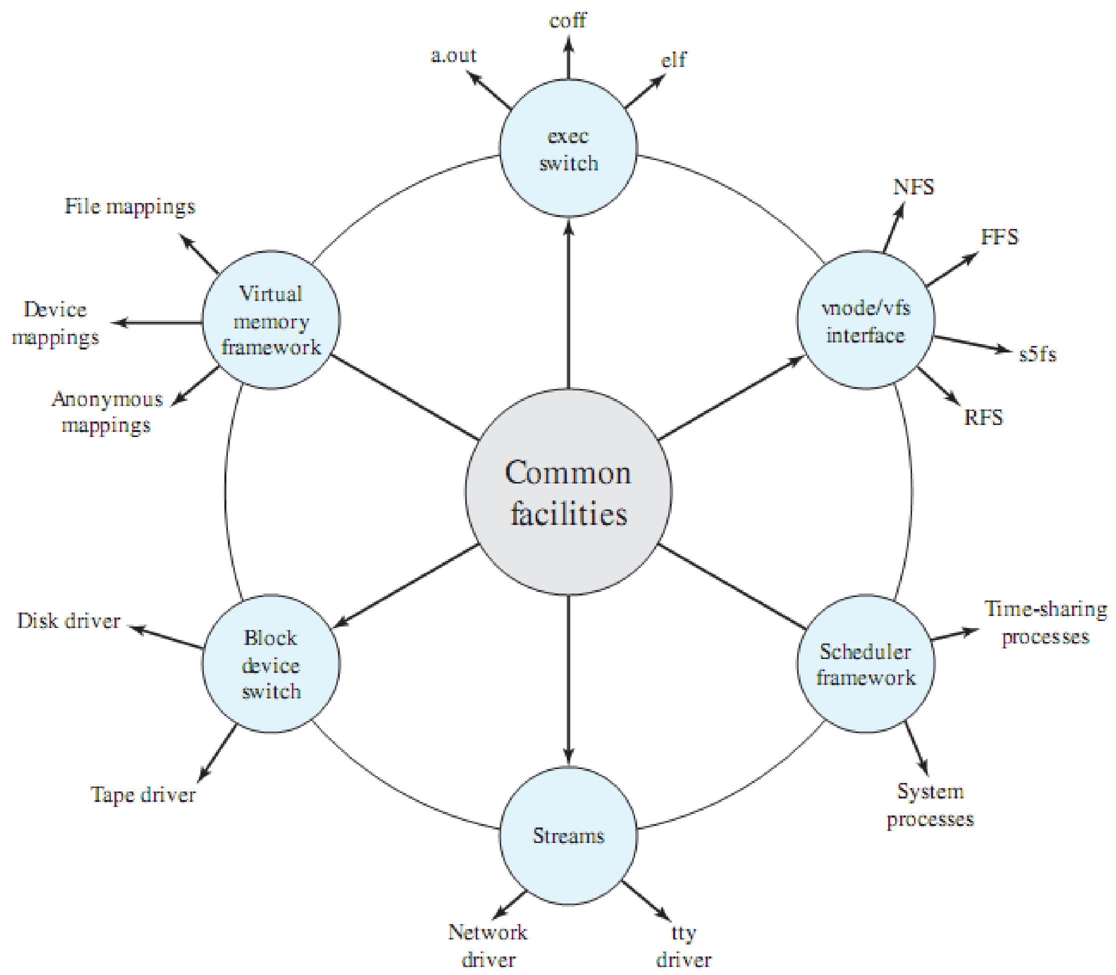
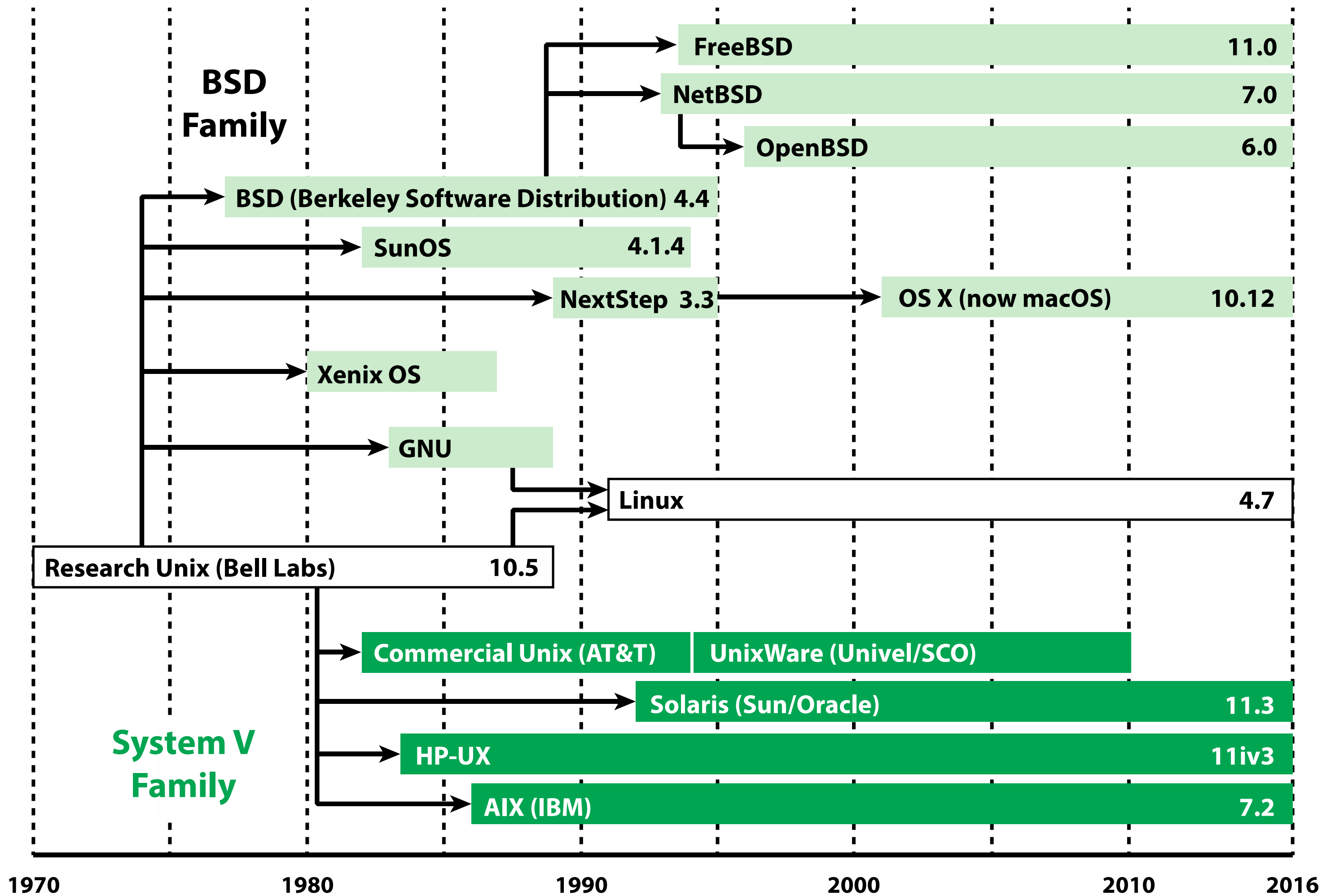


Figure 2.15 **Traditional UNIX Kernel**

System V Release 4 (SVR4) – Modern Unix Kernel



Unix Family Tree



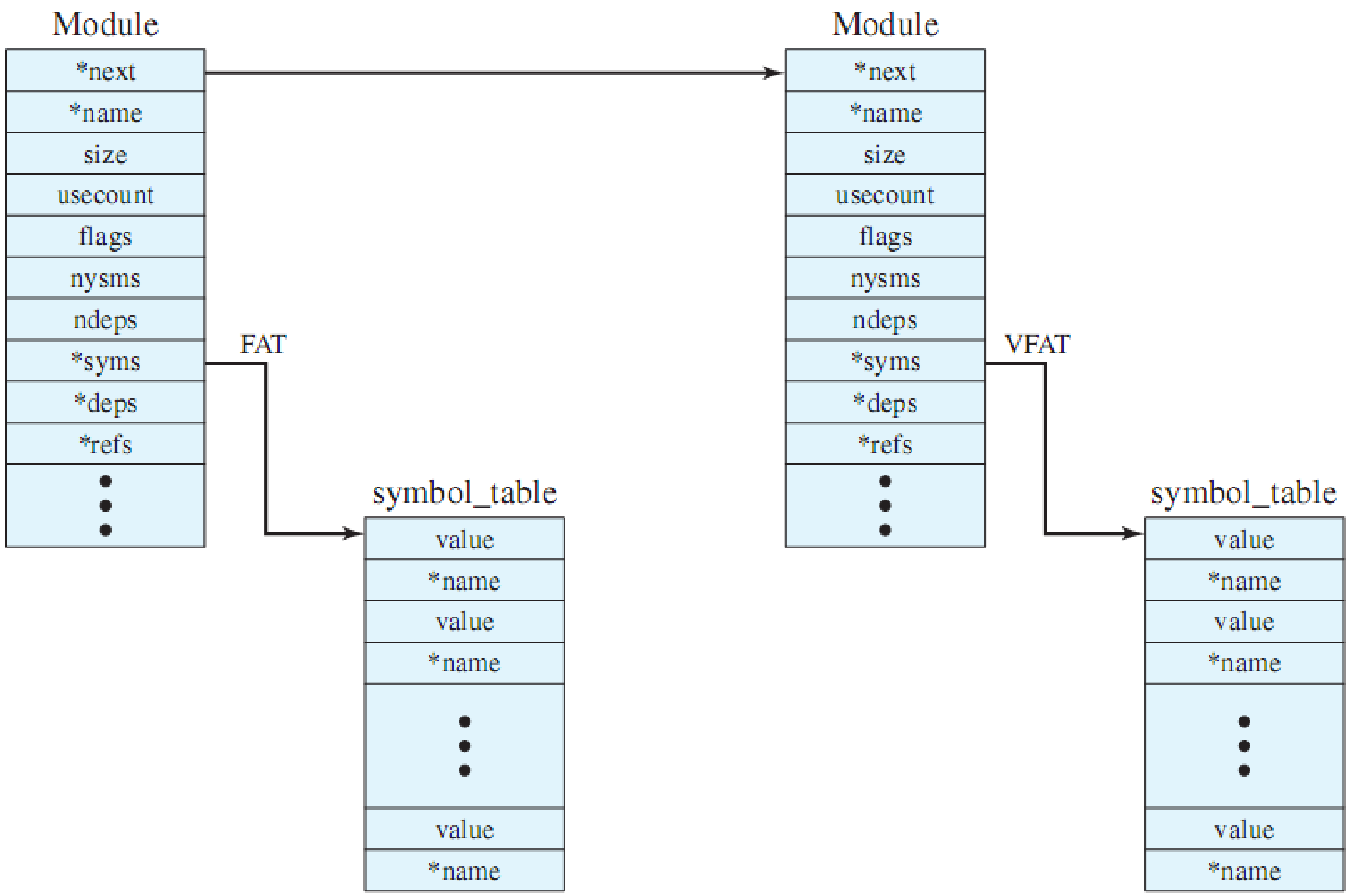
Roadmap

- Operating System Objectives/Functions
- The Evolution of Operating Systems
- Major Achievements
- Developments Leading to Modern Operating Systems
- Microsoft Windows Overview
- UNIX Systems
- **Linux**

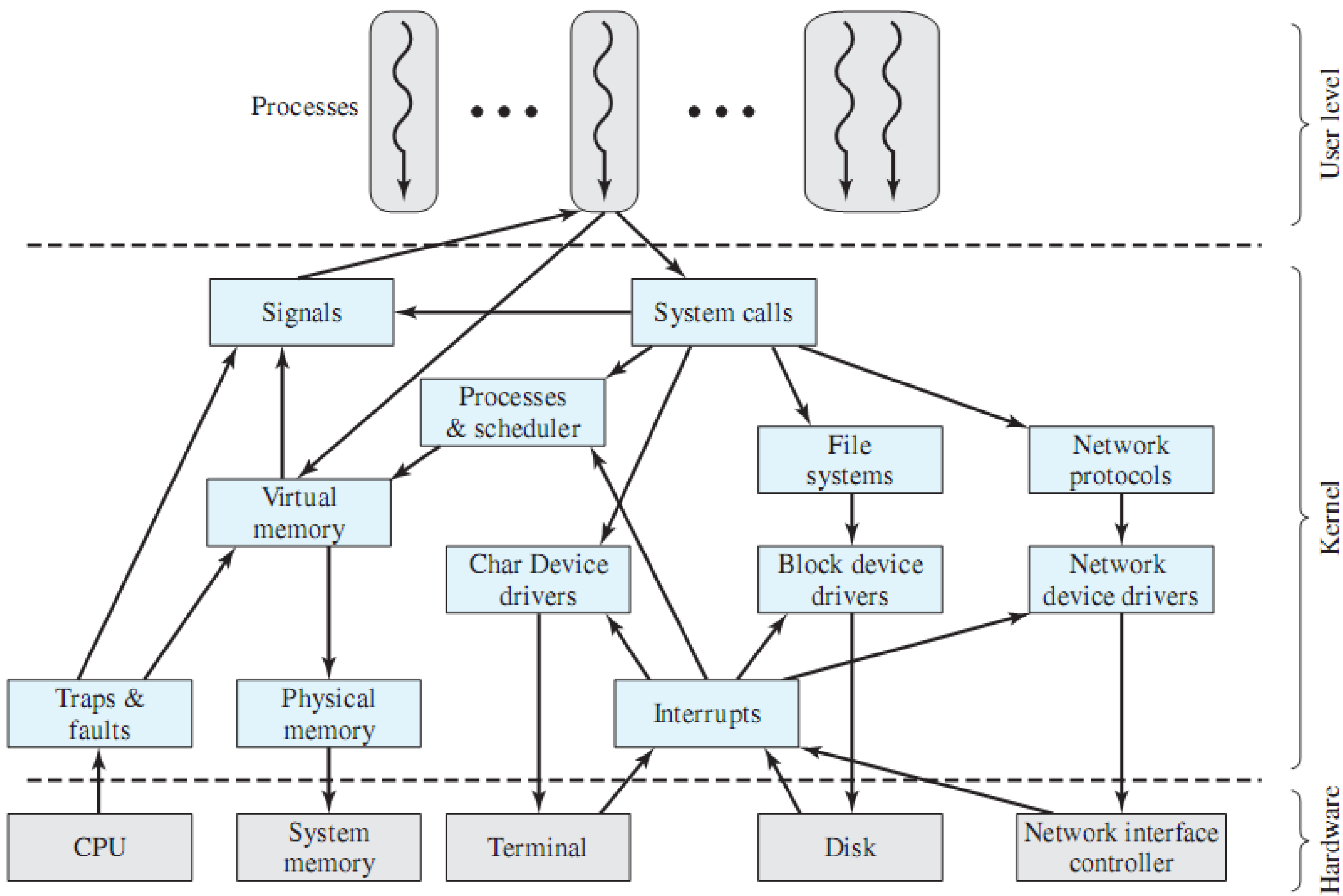
Modular Monolithic Kernel

- Although monolithic, the kernel is structures as a collection of modules
  - Loadable modules
  - An object file which can be linked and unlinked at run time
- Characteristics:
  - Dynamic Linking
  - Stackable modules

Linux Kernel Modules



Linux Kernel Components



Bibliography

- William Stallings, „Operating Systems. Internals and Design Principles”. Ninth Edition, Pearson Prentice Hall, 2017
- Dave Bremer, Otago Polytechnic, N.Z., Prentice Hall