# Chapter 2 Lectures

**Stallings - 9th Ed**

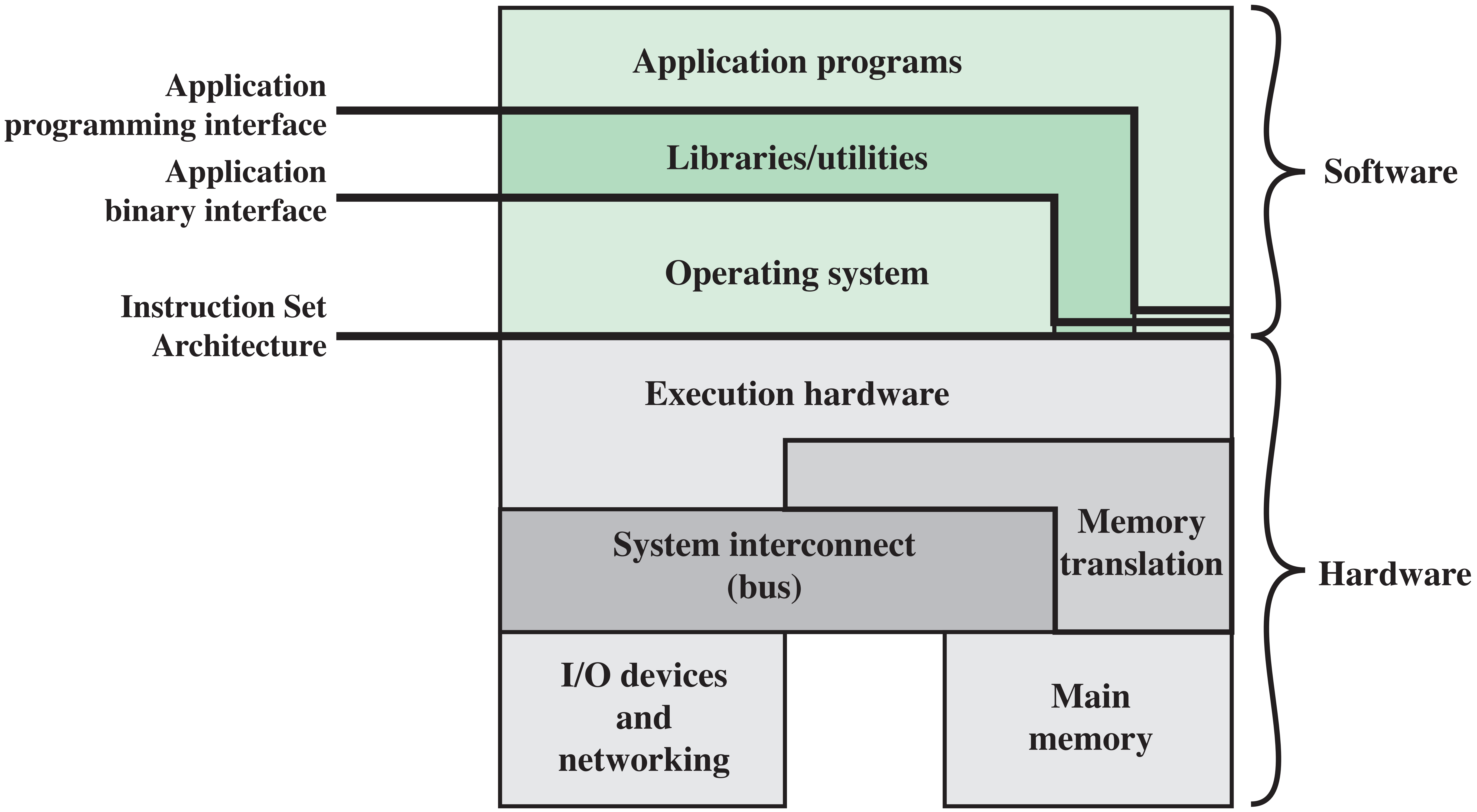

Application programs
Application programming interface
Libraries/utilities
Software
Application binary interface
Operating system
Instruction Set Architecture
Execution hardware
Memory translation
System interconnect (bus)
Hardware
I/O devices and networking
Main memory

**Figure 2.1 Computer Hardware and Software Structure**

# Operating System

- A program that controls the execution of application programs
- An interface between applications and hardware

# Operating System Objectives

- Convenience
  - Makes the computer more convenient to use
- Efficiency
  - Allows computer system resources to be used in an efficient manner
- Ability to evolve
  - Permit effective development, testing, and introduction of new system functions without interfering with service

# Services Provided by the Operating System

- Program development
  - Editors and debuggers
- Program execution
- Access to I/O devices
- Controlled access to files
- Memory management
- System access
- Network support
- …

# Services Provided by the Operating System (2)

- Error detection and response
  - Internal and external **hardware** errors
    - Memory error
    - Device failure
  - **Software** errors
    - Arithmetic overflow
    - Access forbidden memory locations
  - Operating system cannot grant request of application

# Services Provided by the Operating System (3)

- Accounting
  - Collect usage statistics
  - Monitor performance
  - Used to anticipate future enhancements
  - Used for billing purposes
  - check out the log files of a unix system
    - where do you find this information?

# Operating System

- Responsible for managing resources
- An OS is just a program (or set of programs) that is executed

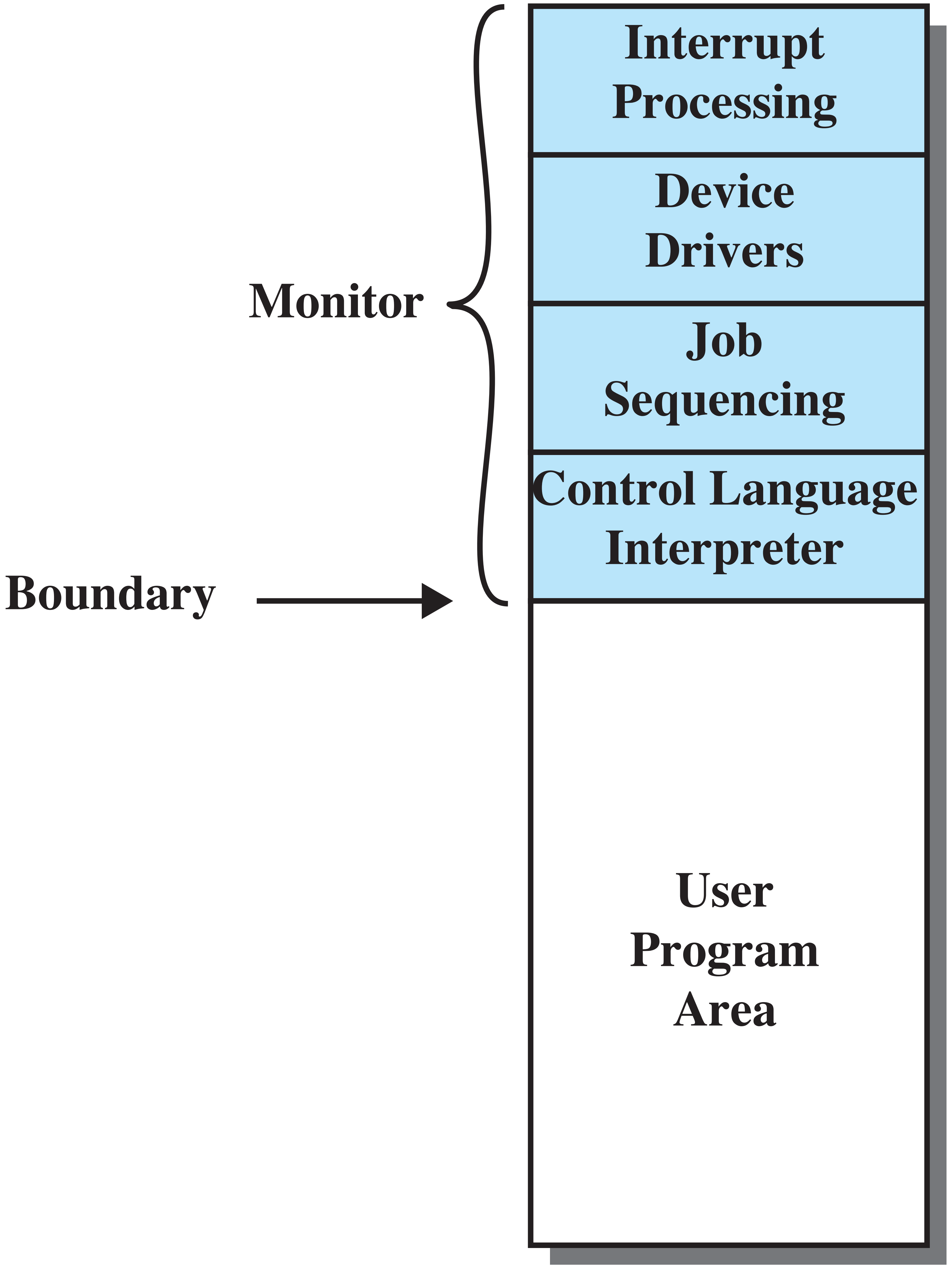


Figure 2.3 Memory Layout for a Resident Monitor

# Evolution of Operating Systems

- Serial Processing
  - No operating system
    - Machines run from a console with display lights, toggle switches, input device, and printer
    - Schedule time, e.g. sign up
    - Setup included loading the compiler and source program, saving compiled program, loading and linking

# Evolution of Operating Systems

- Simple Batch Systems
  - Monitor
    - Software that controls the sequence of events
    - Batch jobs together
    - Program branches back to monitor when finished

# Job Control Language (JCL)

- Special type of programming language
- Provides instruction to the monitor, e.g.
  - What compiler to use
  - What data to use

# Hardware Features of Interest in Operating Systems

- User/Supervisor Mode
- Interrupts/Traps/Exceptions
- Real-time Clock (Timer) Operation
- Support for Virtual Memory and Memory Protection

# Hardware Features

- Memory protection
  - Do not allow the memory area containing the monitor to be altered
- Timer
  - Prevents a job from monopolizing the system

# Hardware Features

- Privileged instructions
  - Certain machine level instructions can only be executed by the monitor
- Interrupts
  - Early computer models did not have this capability

# Memory Protection

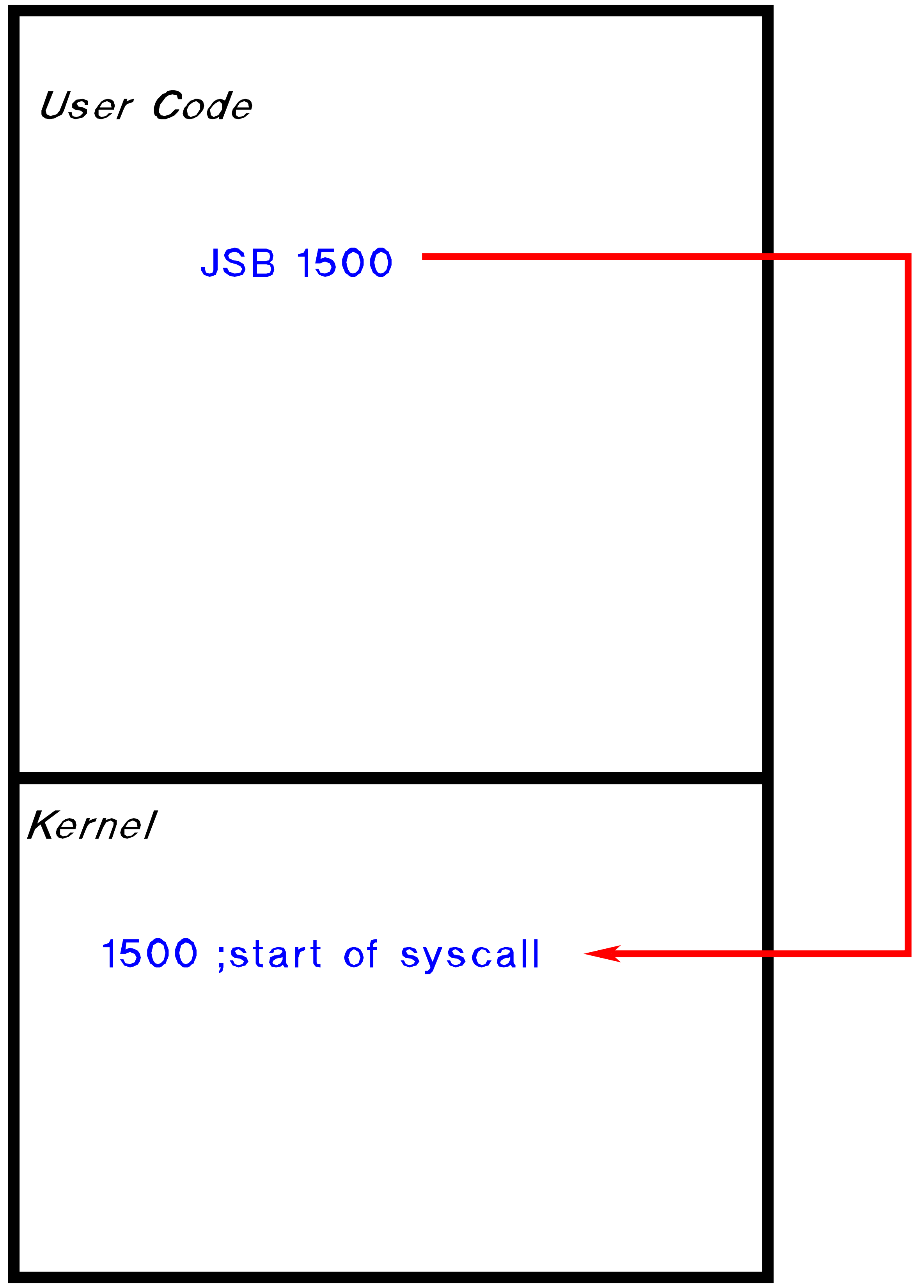
- User program executes in **user mode**
  - Certain instructions may not be executed
- Monitor executes in **system mode**
  - Kernel mode
  - Privileged instructions are executed
  - Protected areas of memory may be accessed

# Traps and Interrupts and Exceptions (Oh My!)

- Trap - Caused by executing a “trap” instruction. Used to request an operation from the operating system. Also called a “syscall” or “synchronous trap.”
- Interrupt - caused by a hardware device when it needs service. Also called an “asynchronous trap.”
- Exception - caused when an illegal operation (eg, divide by zero or memory reference out of bounds) is attempted.

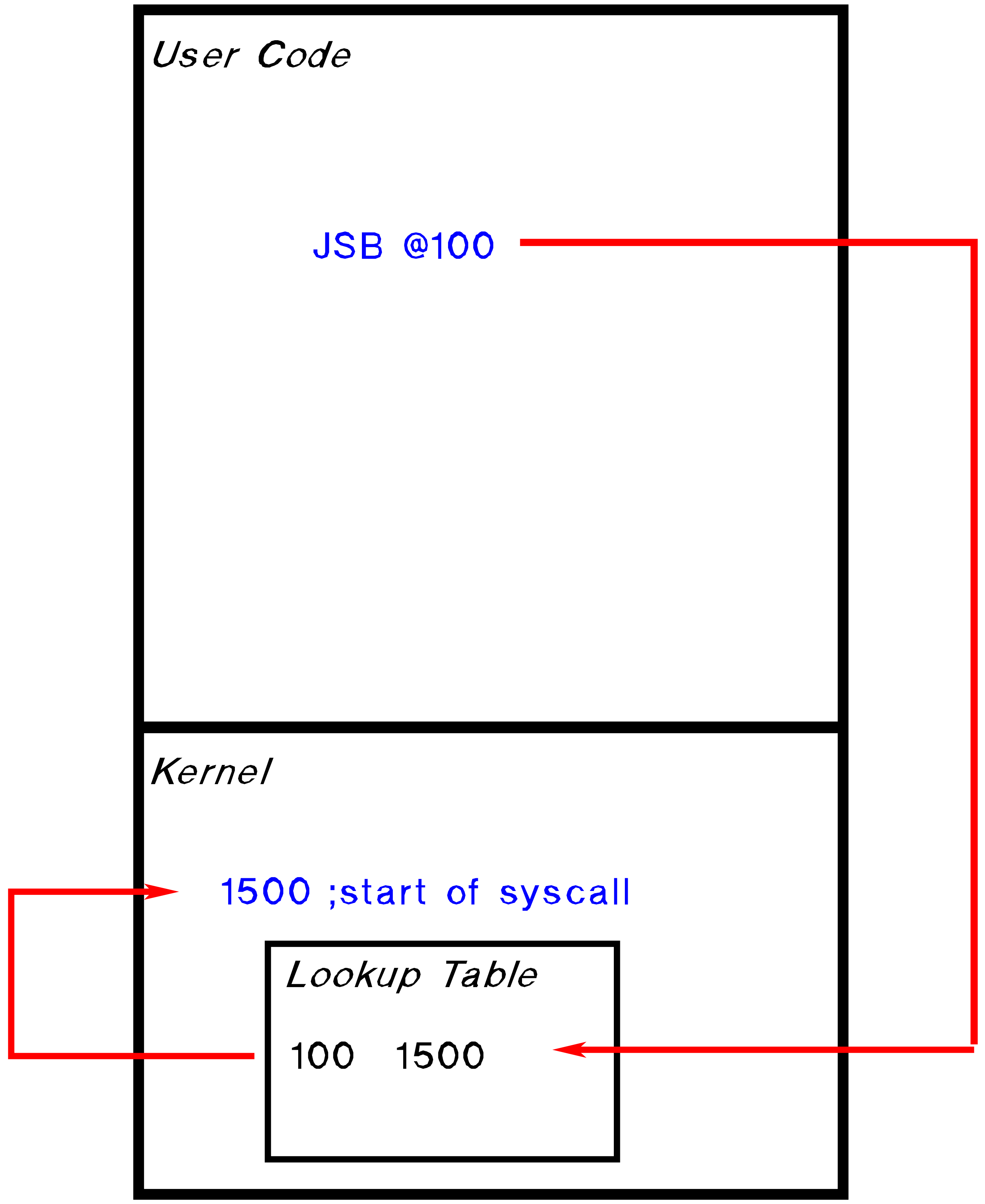
**With all three, the processor response is the same - an “indirect subroutine call” through the vector table, along with a change from user to kernel mode.**

# System Calls Using Subroutines



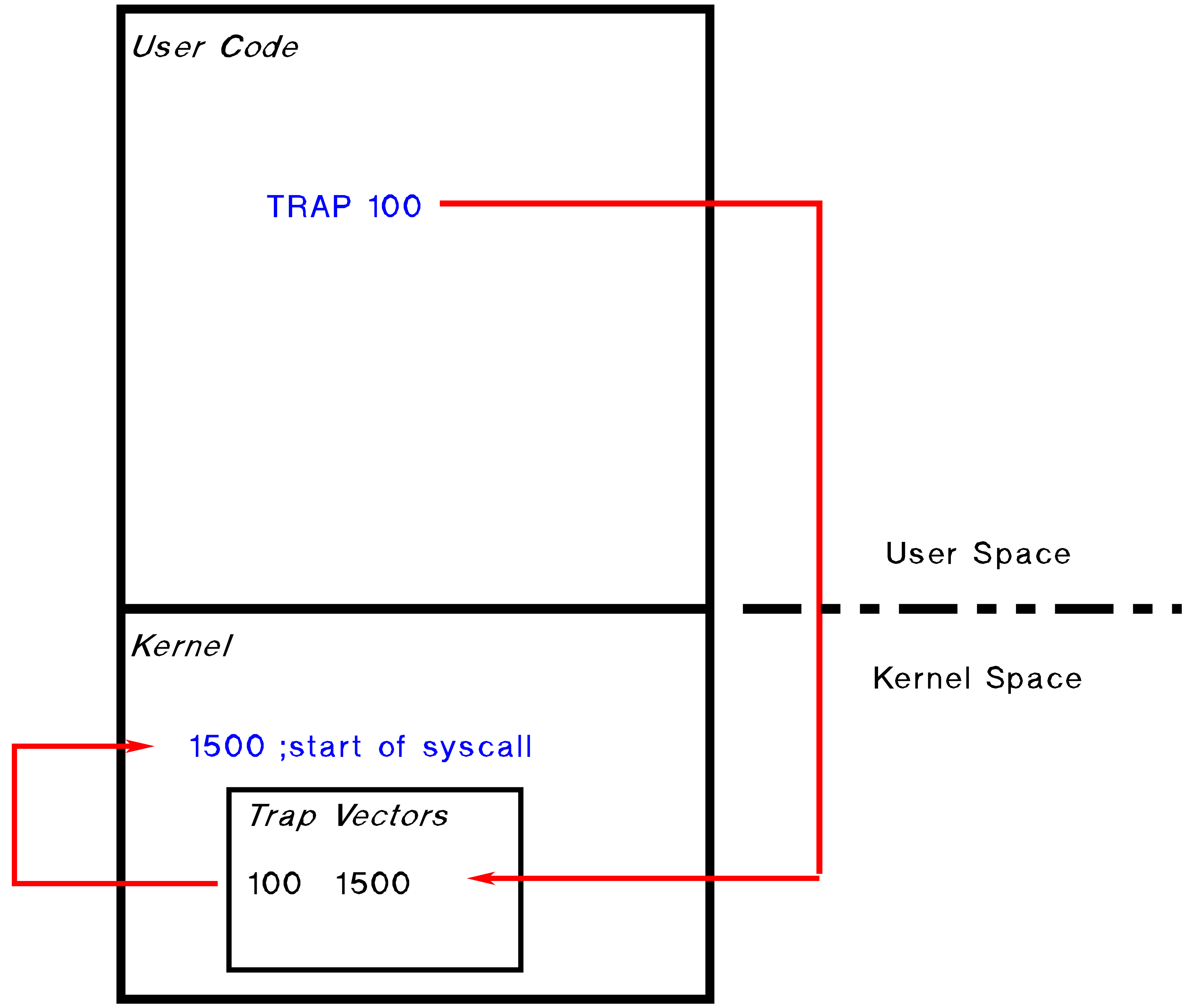
TRAP010

# System Calls Using Lookup Tables



TRAP020

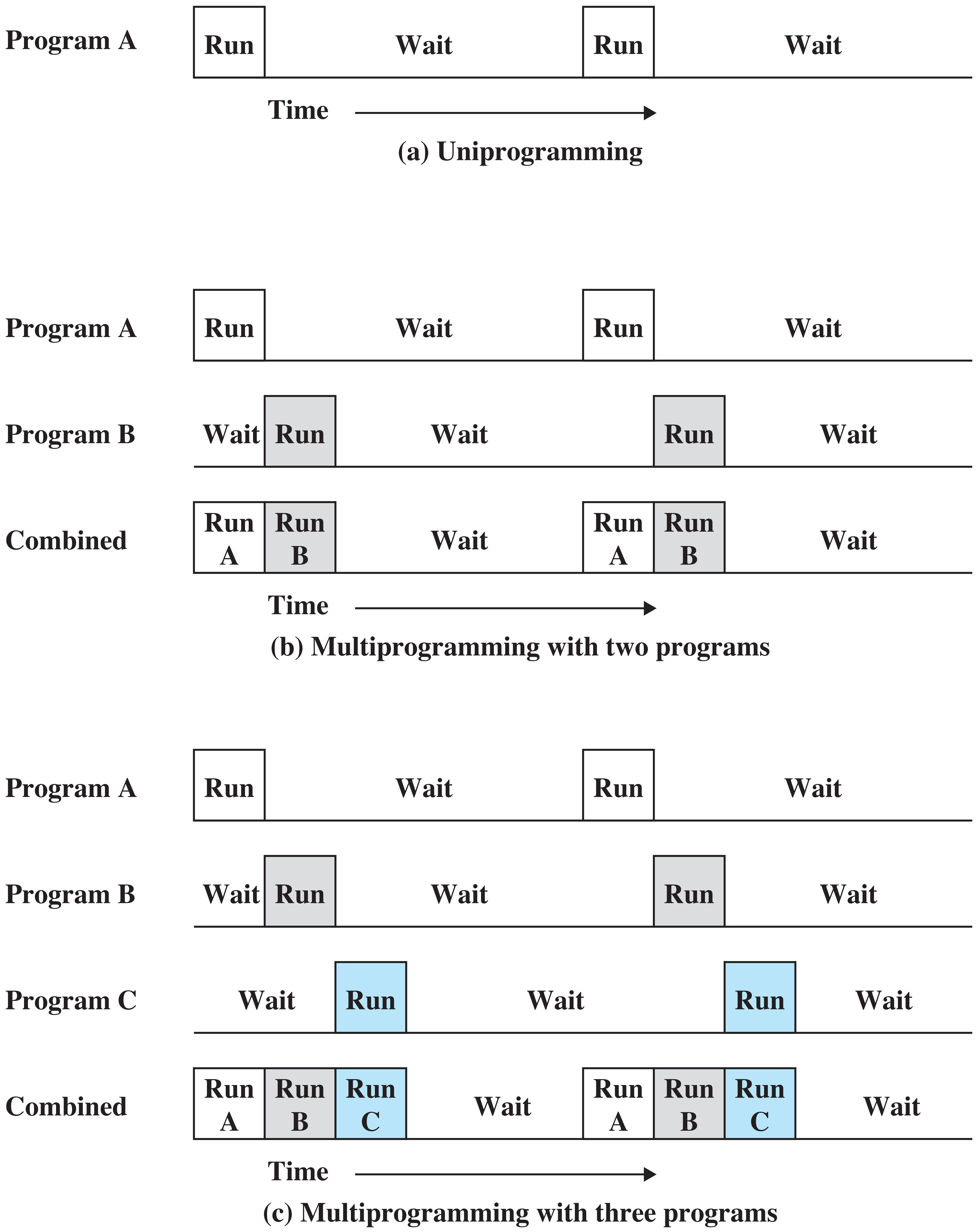
# System Calls Using the Trap Instruction



| | |
|---|---|
| Read one record from file | 15 $\mu$s |
| Execute 100 instructions | 1 $\mu$s |
| Write one record to file | $\underline{15\ \mu\text{s}}$ |
| TOTAL | 31 $\mu$s |

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

**Figure 2.4 System Utilization Example**



(a) Uniprogramming

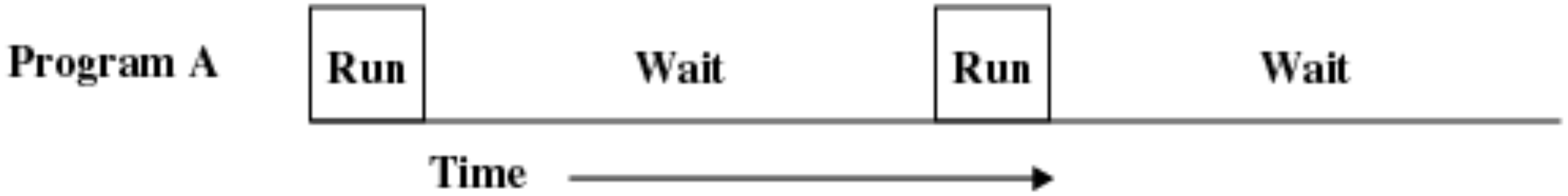
(b) Multiprogramming with two programs

(c) Multiprogramming with three programs

Figure 2.5 Multiprogramming Example

# Uniprogramming

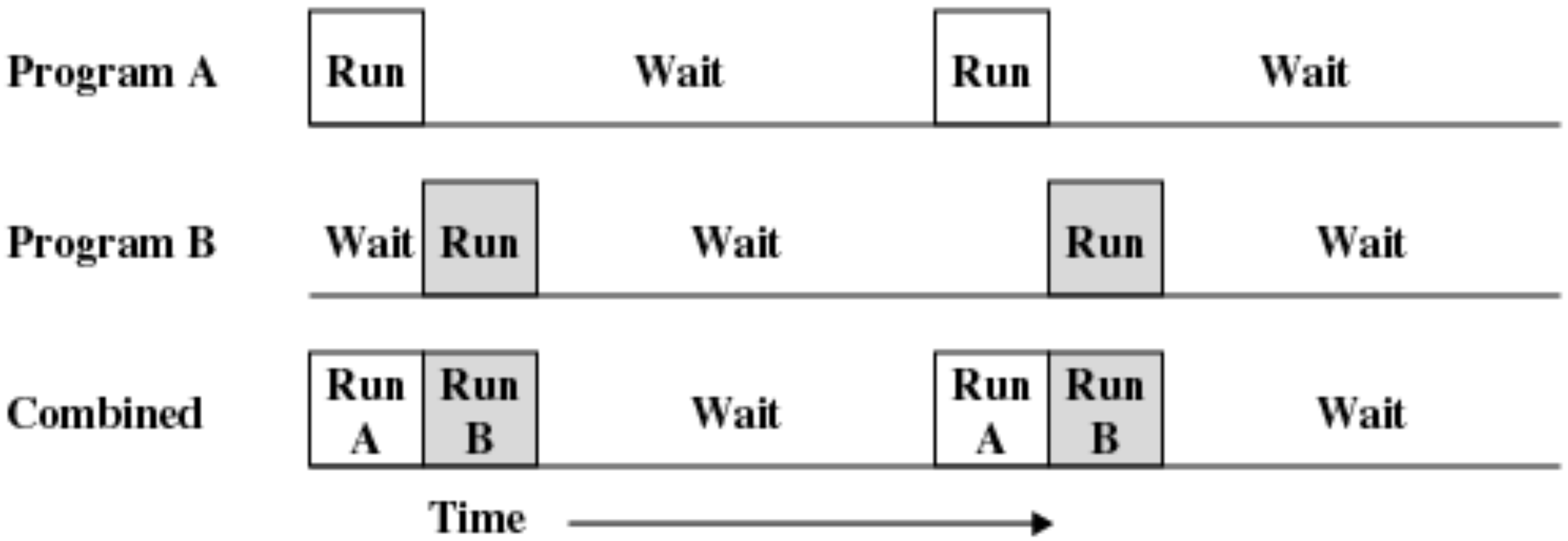
- Processor must wait for I/O instruction to complete before preceding



(a) Uniprogramming

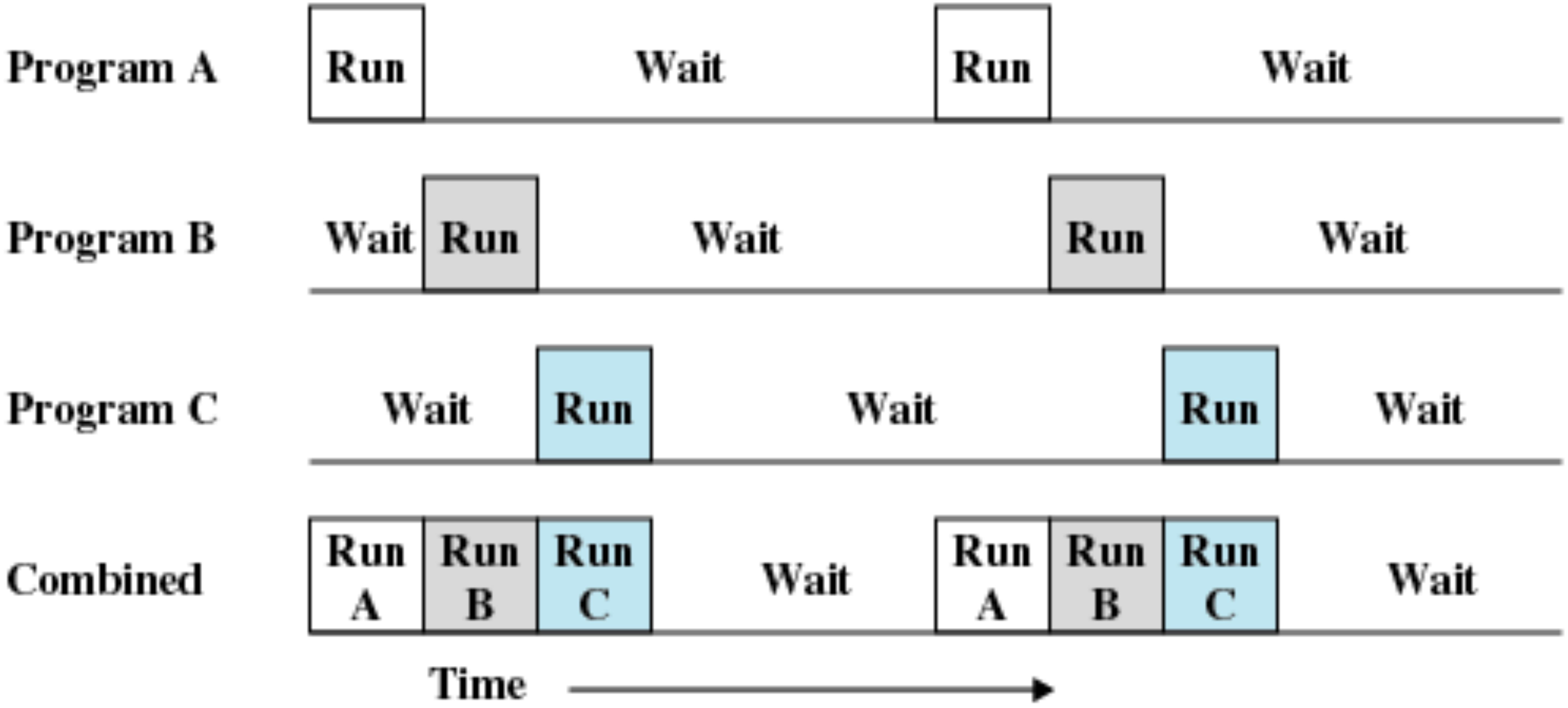
# Multiprogramming

- 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

# Time Sharing

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

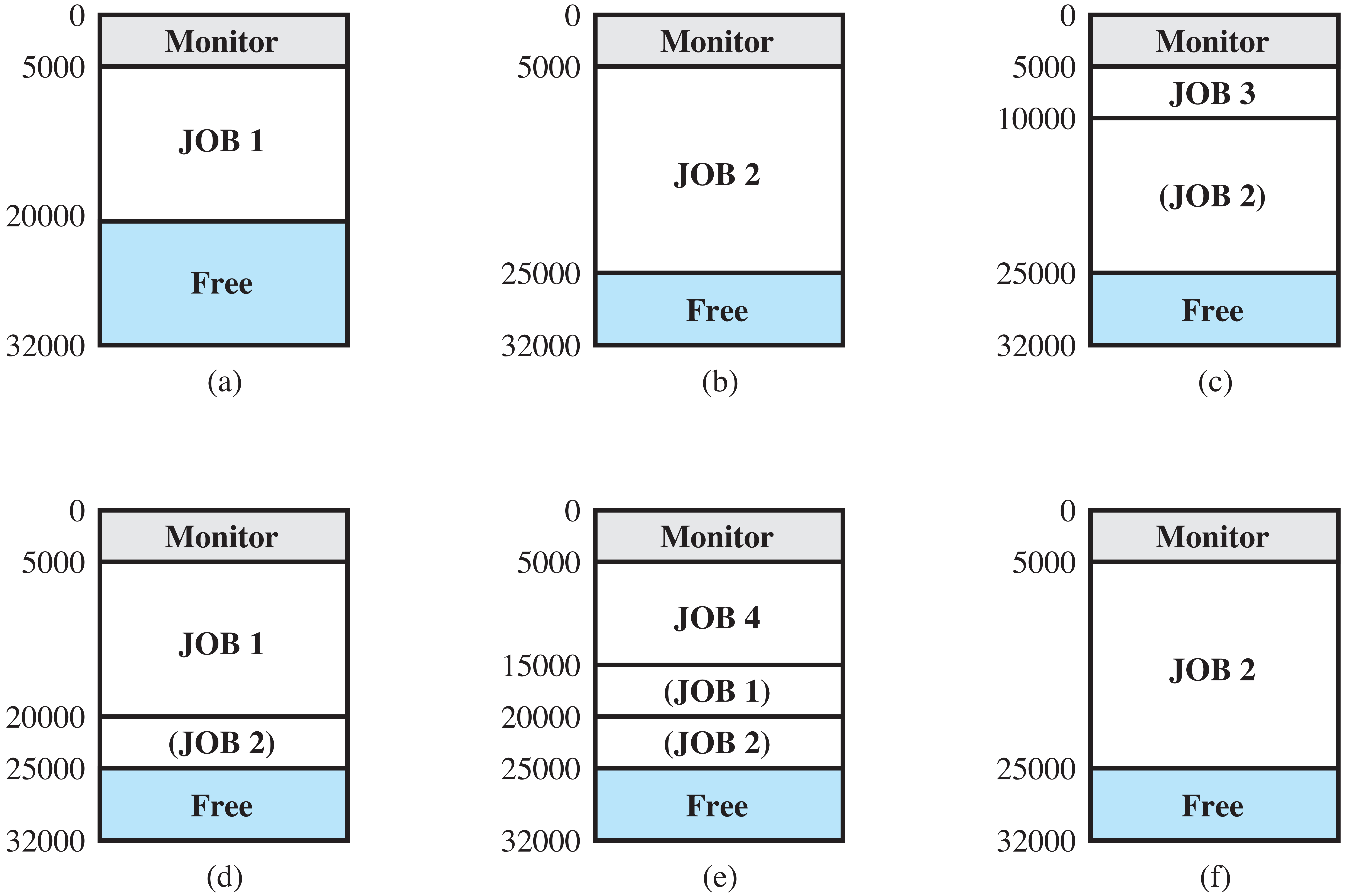


**Figure 2.7 CTSS Operation**

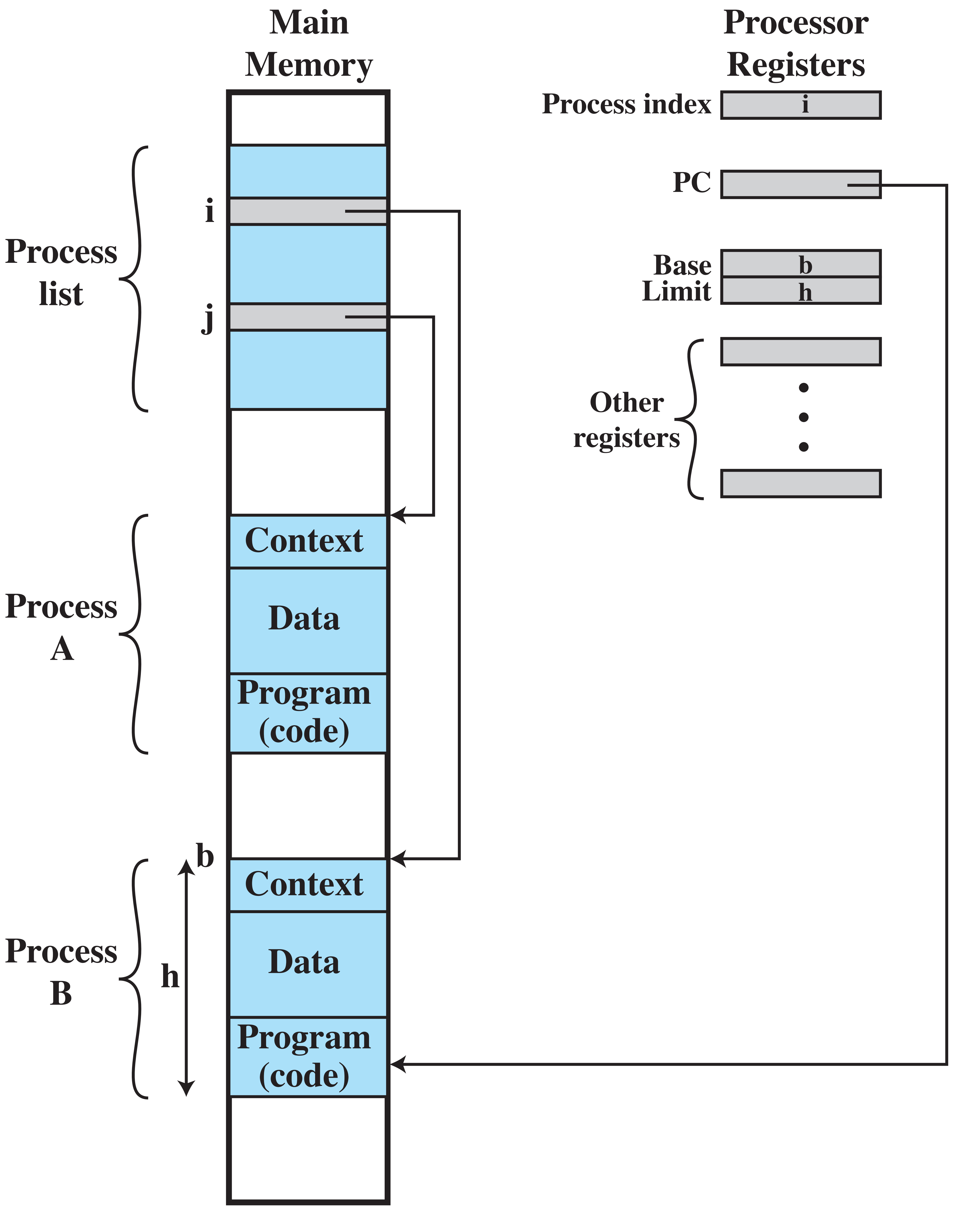


Figure 2.8 Typical Process Implementation

# Major Achievements

- Denning et.al. [DENN80a] point out 5 major OS advances:
  - Processes
  - Memory Management
  - Information protection and security
  - Scheduling and resource management
  - System structure
- Let’ s look at each one...

# Processes

- A program in execution
- An instance of a program running on a computer
- The entity that can be assigned to and executed on a processor
- A unit of activity characterized by a single sequential thread of execution, a current state, and an associated set of system resources

# Memory Management

- Process isolation
  - non-interference between independent procs.
- Automatic allocation and management
  - should be transparent to programmer
- Support of modular programming
- Protection and access control
- Long-term storage
  - after computer has been powered down

# Virtual Memory

- Allows programmers to address memory from a logical point of view
- No hiatus between the execution of successive processes while one process was written out to secondary store and the successor process was read in

# Virtual Memory and File System

- Implements long-term store
- Information stored in named objects called files

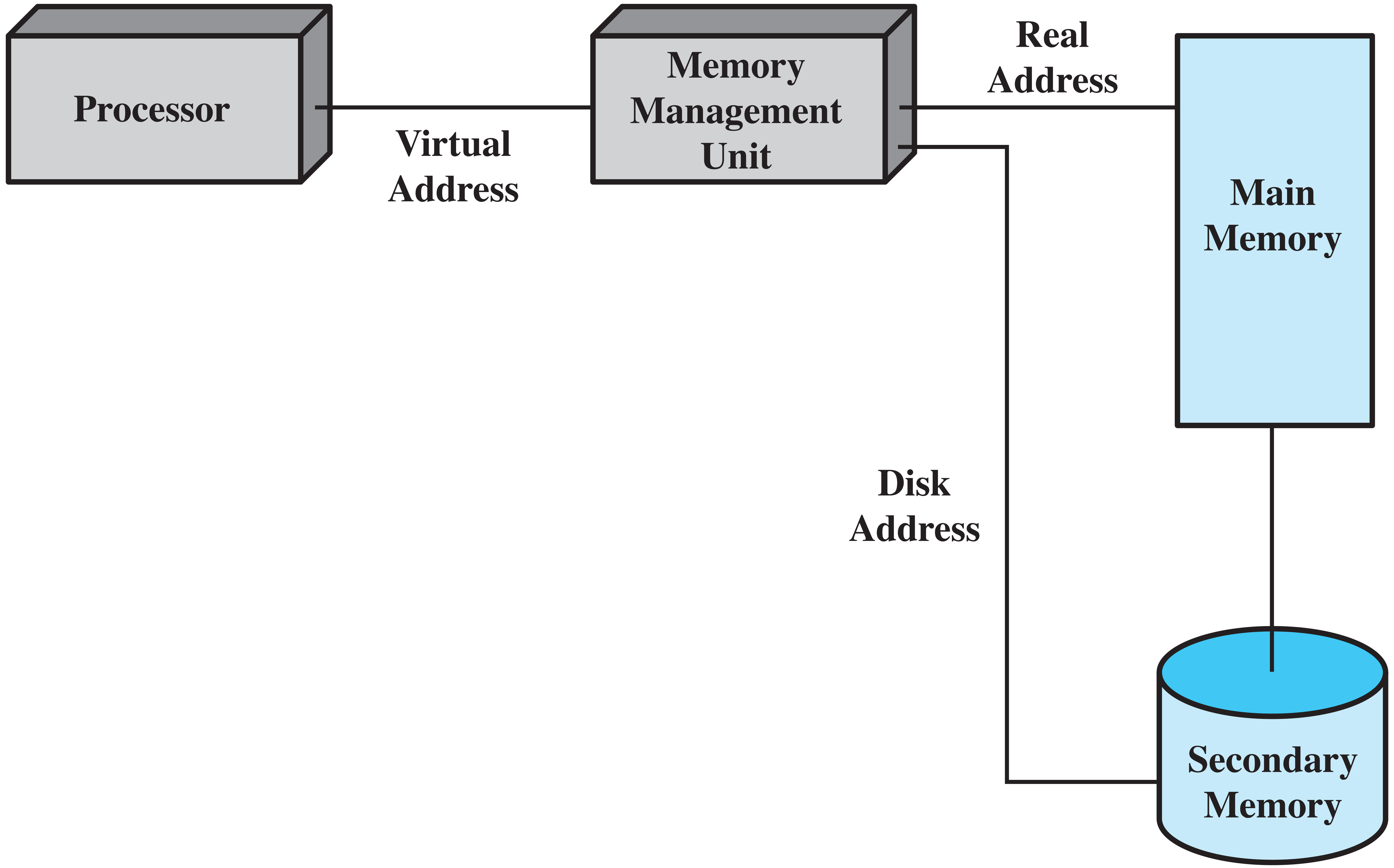
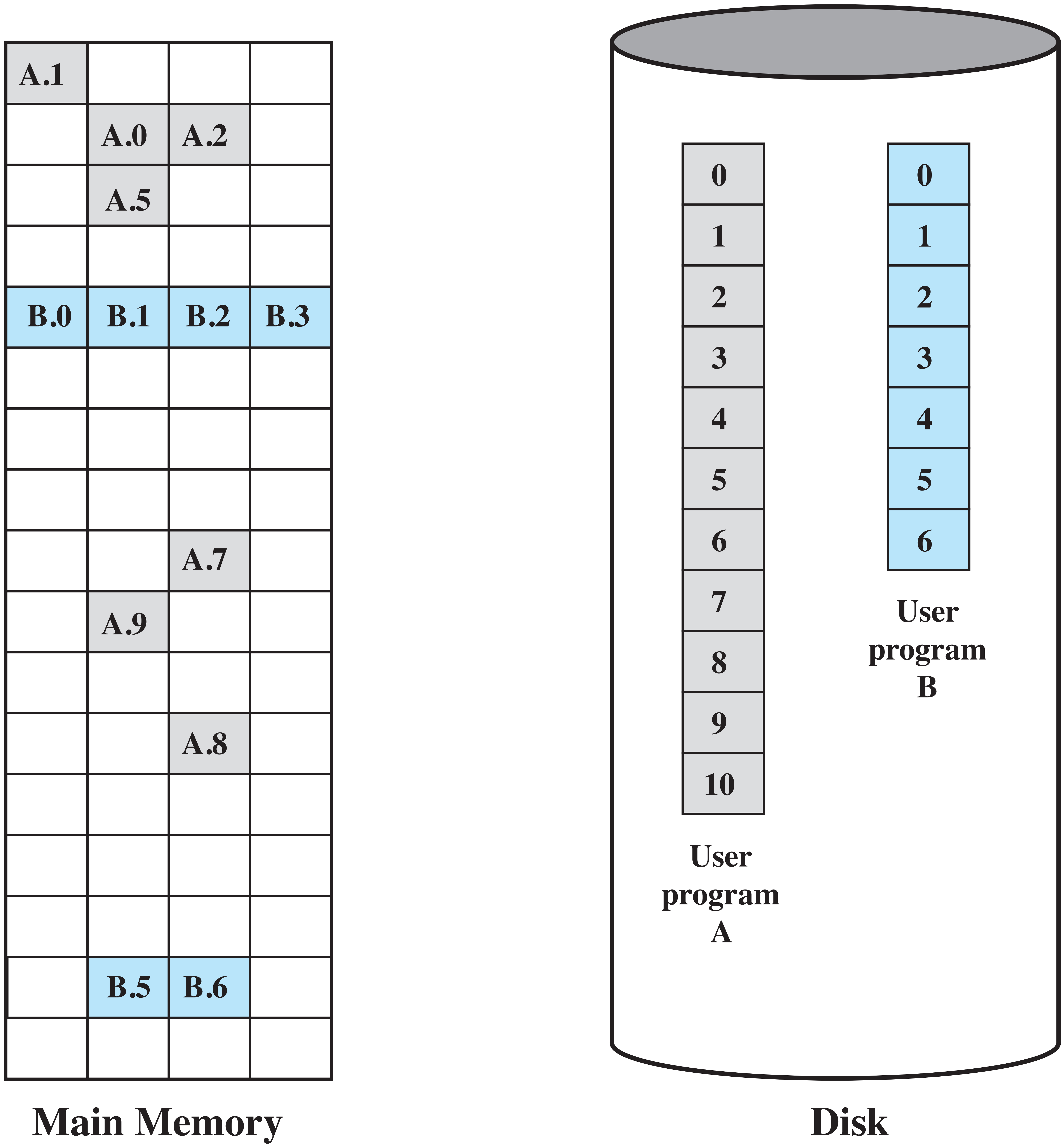


Figure 2.10 Virtual Memory Addressing



Main memory consists of a number of fixed-length frames, each equal to the size of a page. For a program to execute, some or all of its pages must be in main memory.

Secondary memory (disk) can hold many fixed-length pages. A user program consists of some number of pages. Pages of all programs plus the operating system are on disk, as are files.

Figure 2.9 Virtual Memory Concepts

# Information Protection and Security

- Availability
  - Concerned with protecting the system against interruption
- Confidentiality
  - Assuring that users cannot read data for which access is unauthorized

Sequence 3

# Information Protection and Security

- Data integrity
  - Protection of data from unauthorized modification
- Authenticity
  - Concerned with the proper verification of the identity of users and the validity of messages or data

# Scheduling and Resource Management

- Fairness
  - Give equal and fair access to resources
- Differential responsiveness
  - ...but, OS also needs to discriminate among different classes of jobs
- Efficiency
  - Maximize throughput, minimize response time, and accommodate as many uses as possible

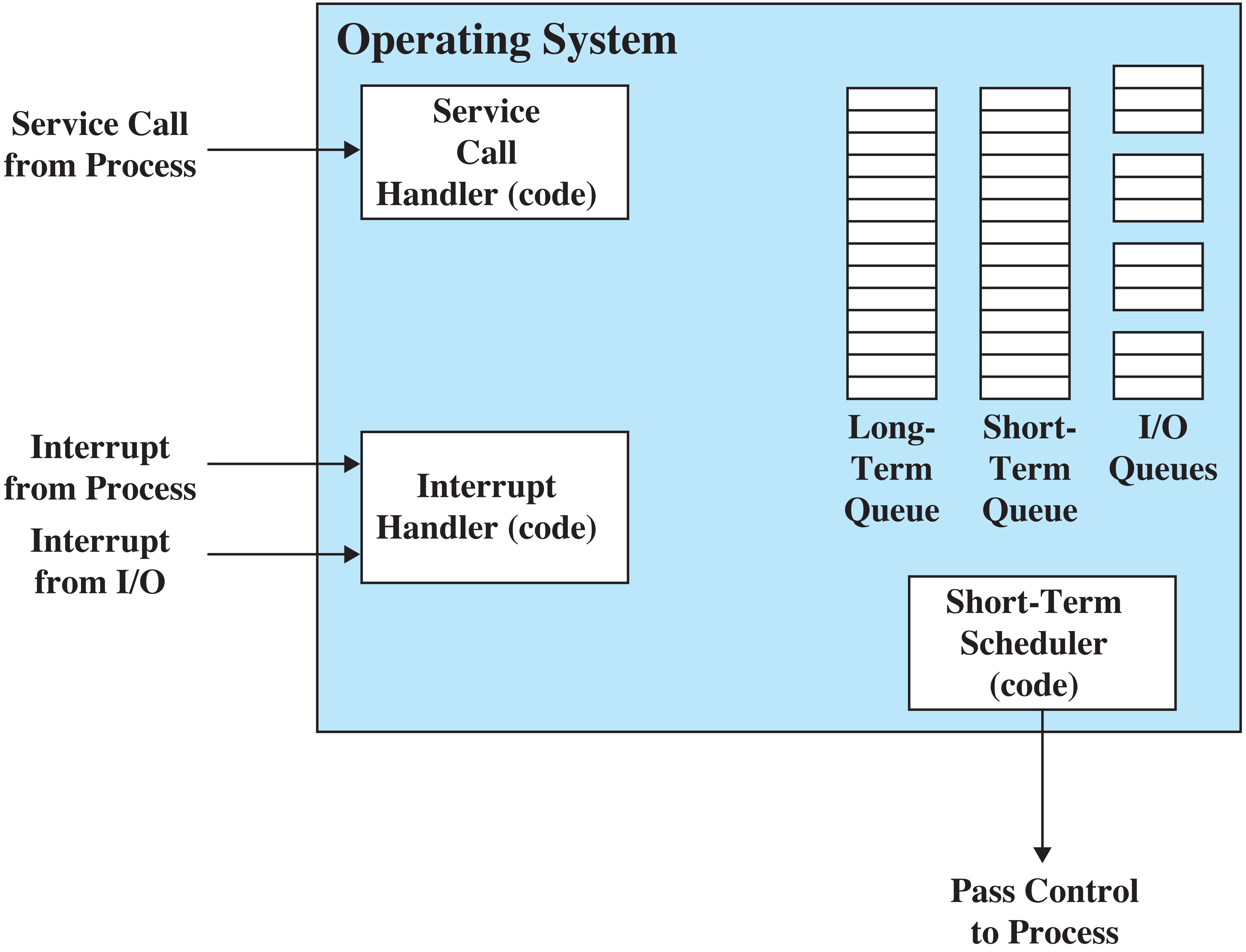
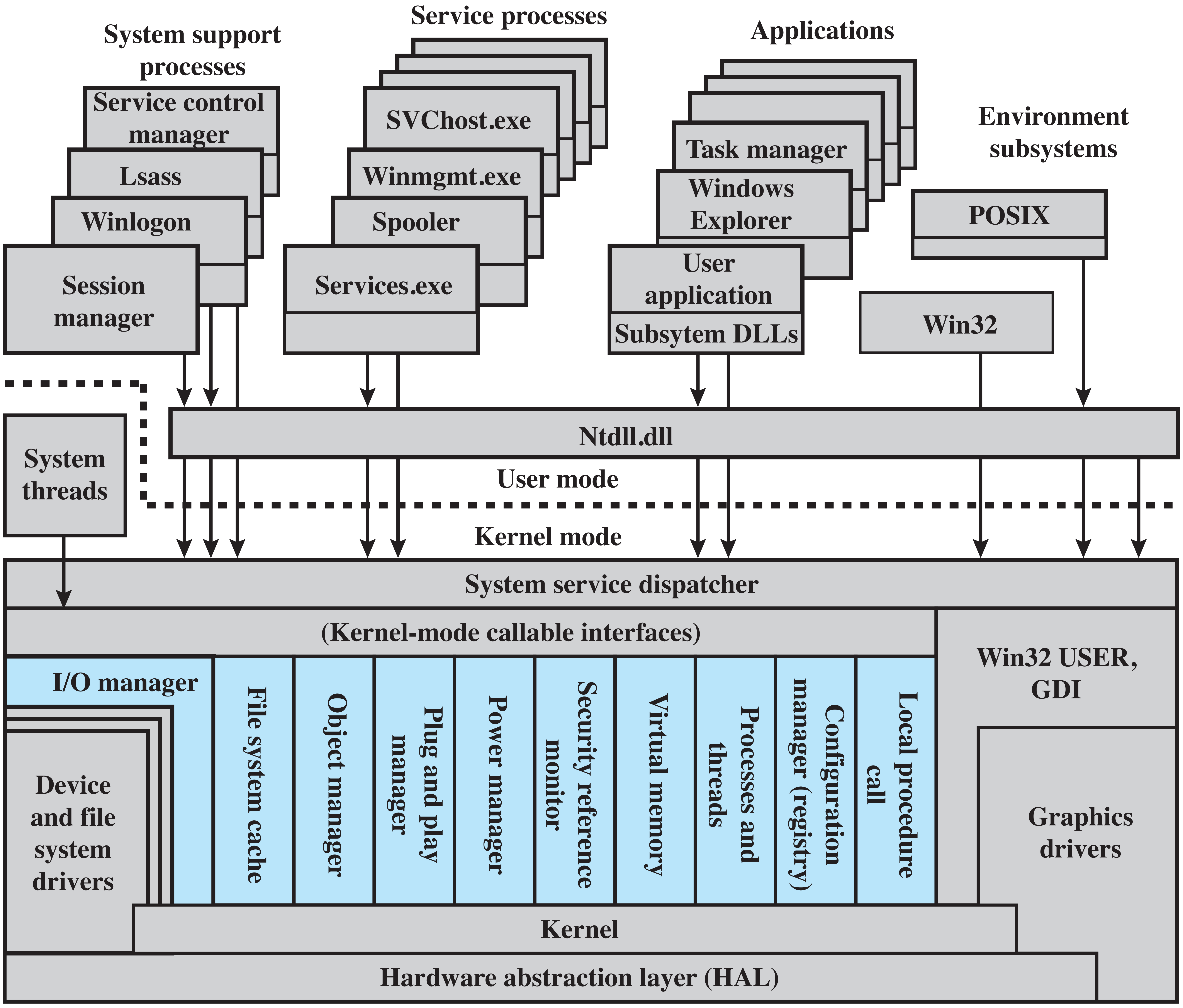


**Figure 2.11 Key Elements of an Operating System for Multiprogramming**



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

Colored area indicates Executive

Figure 2.14 Windows Architecture

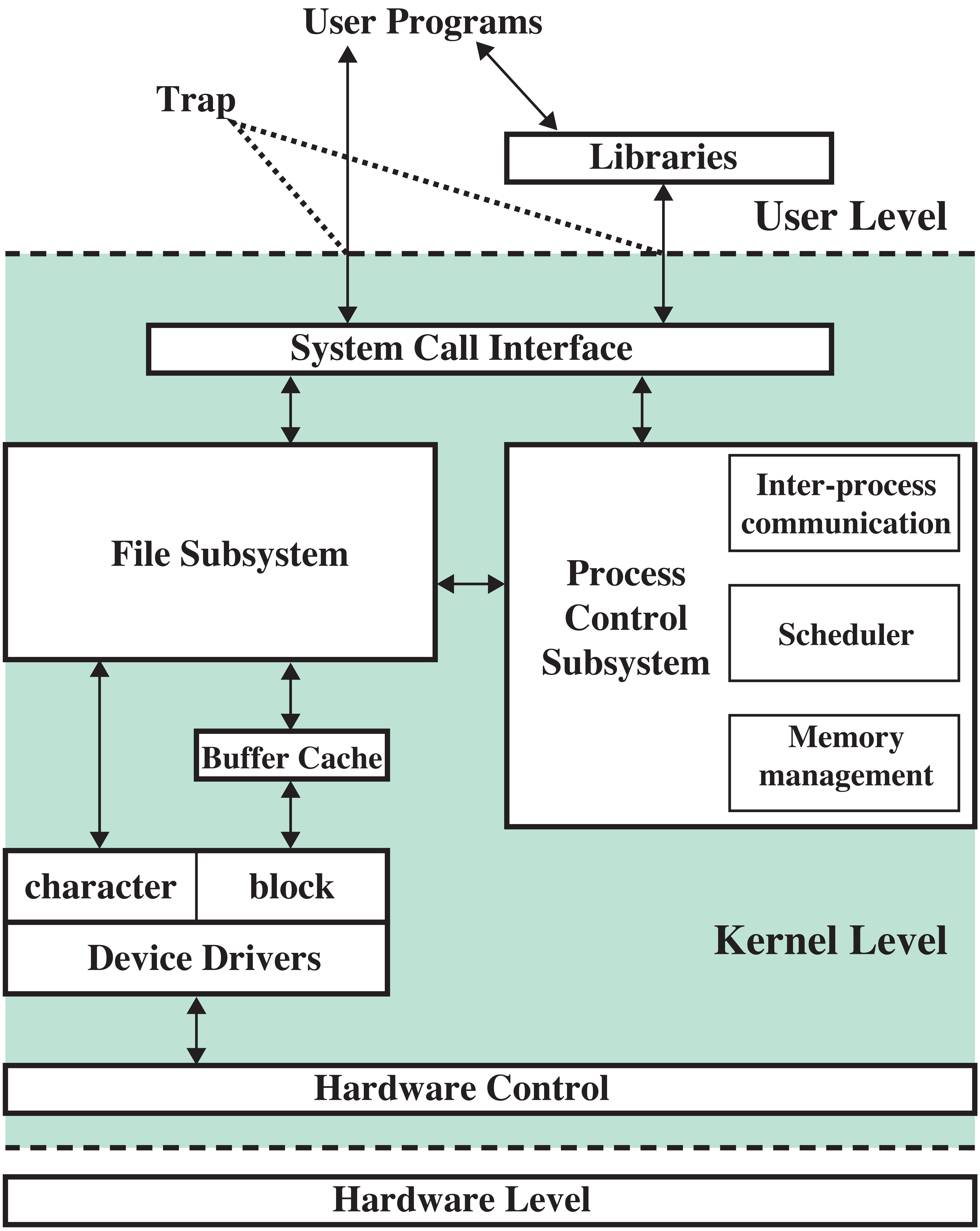


Figure 2.15 Traditional UNIX Kernel

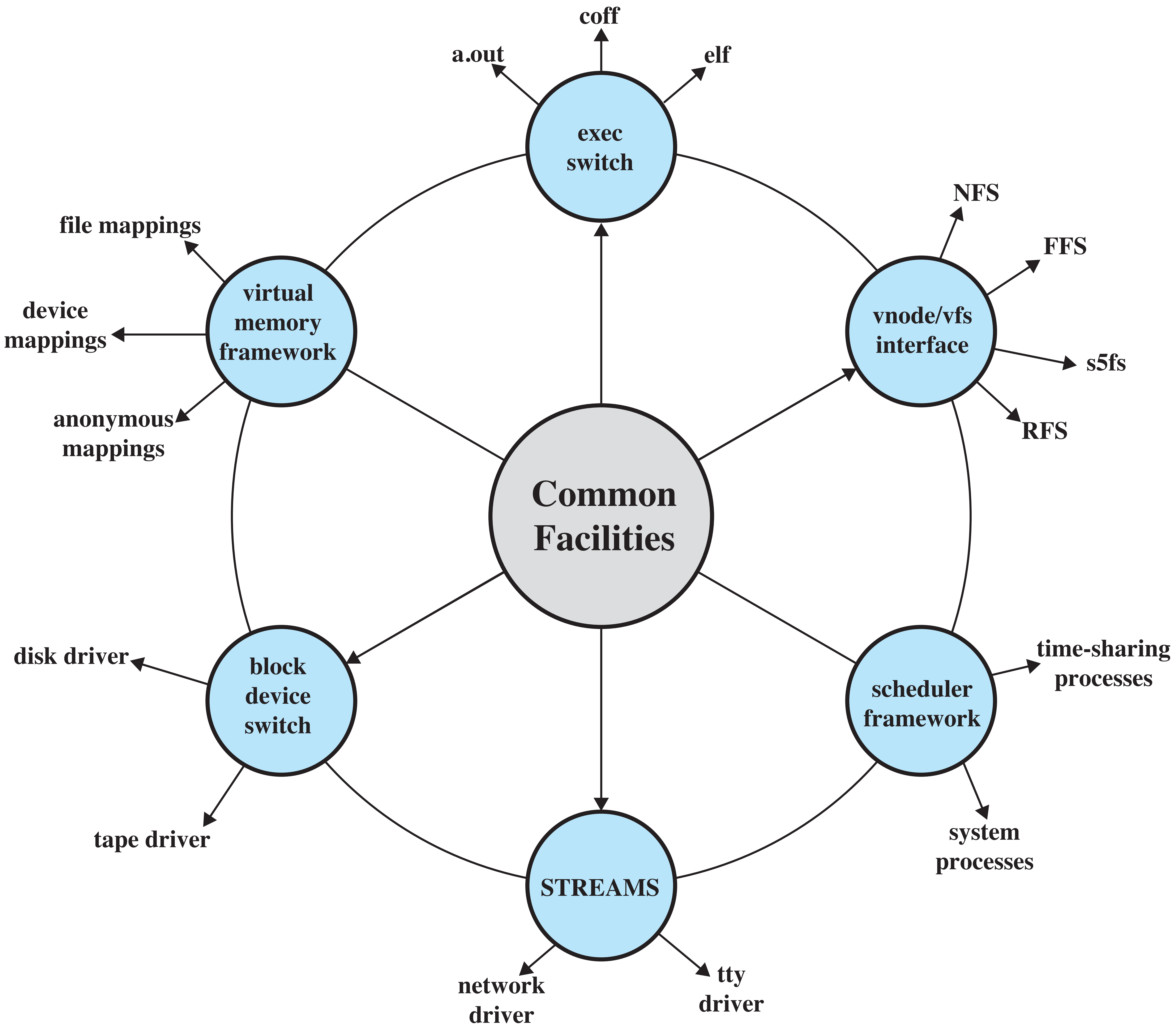


Figure 2.16 Modern UNIX Kernel

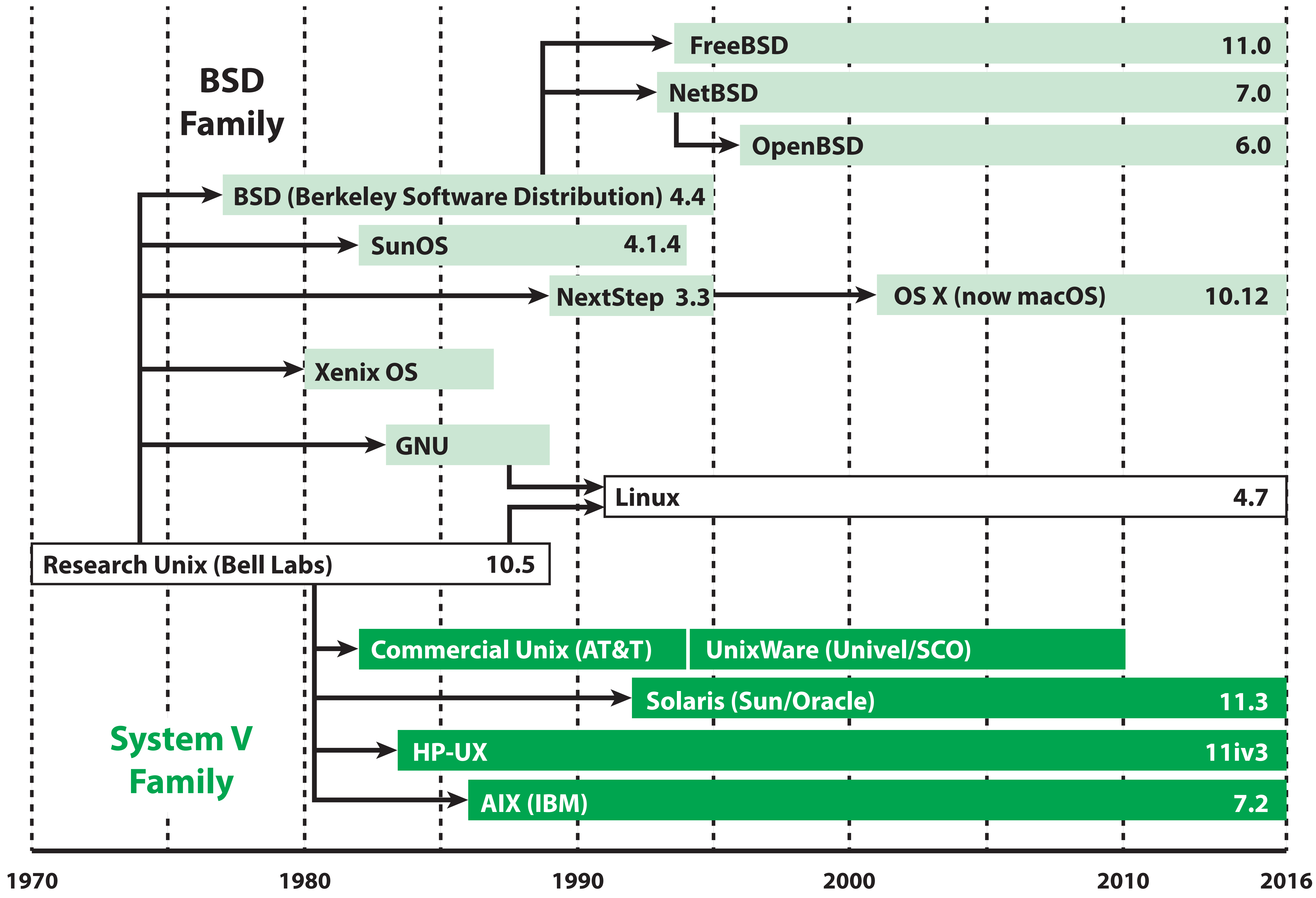


Figure 2.17 Unix Family Tree

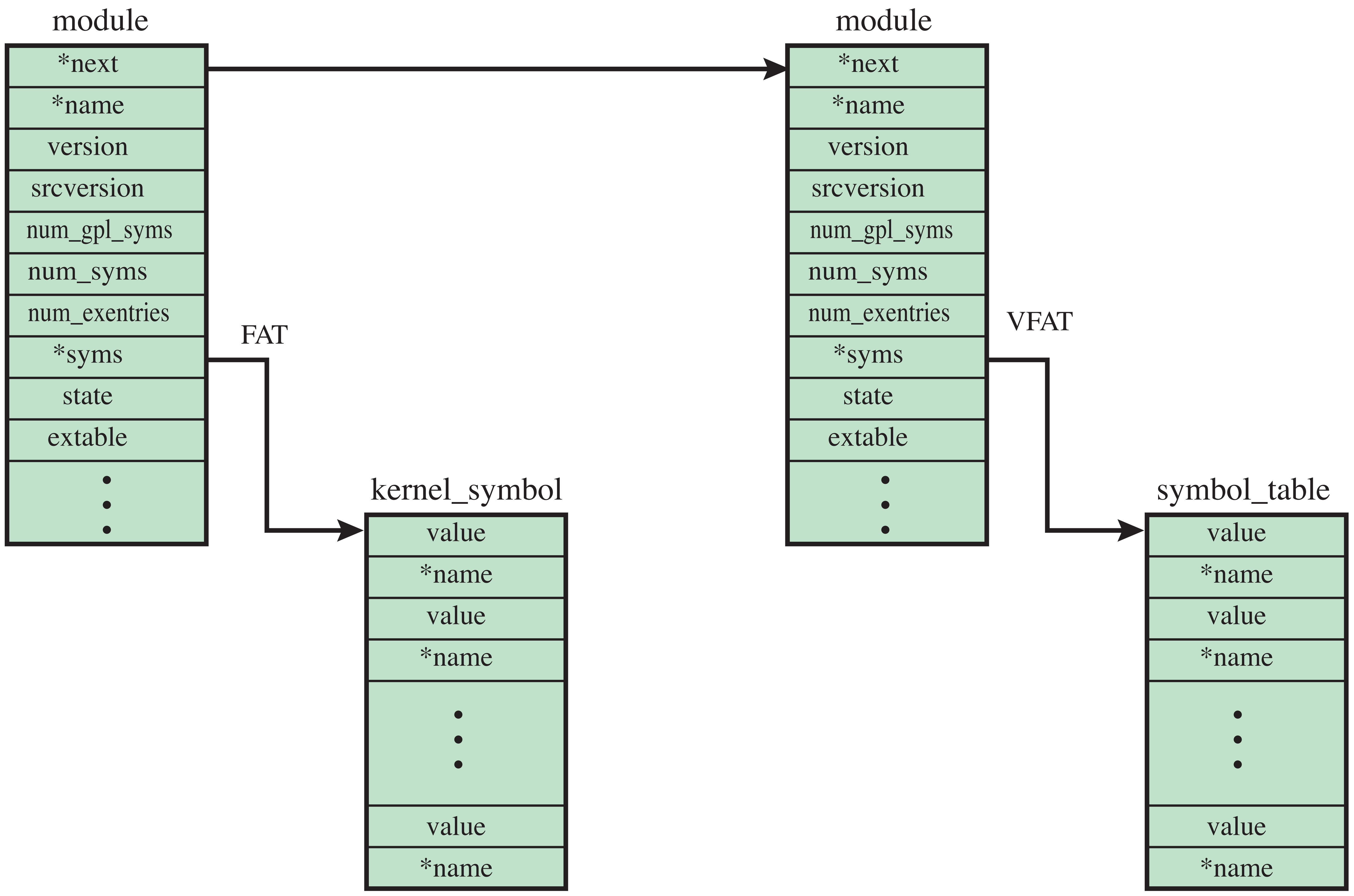


**Figure 2.18  Example List of Linux Kernel Modules**

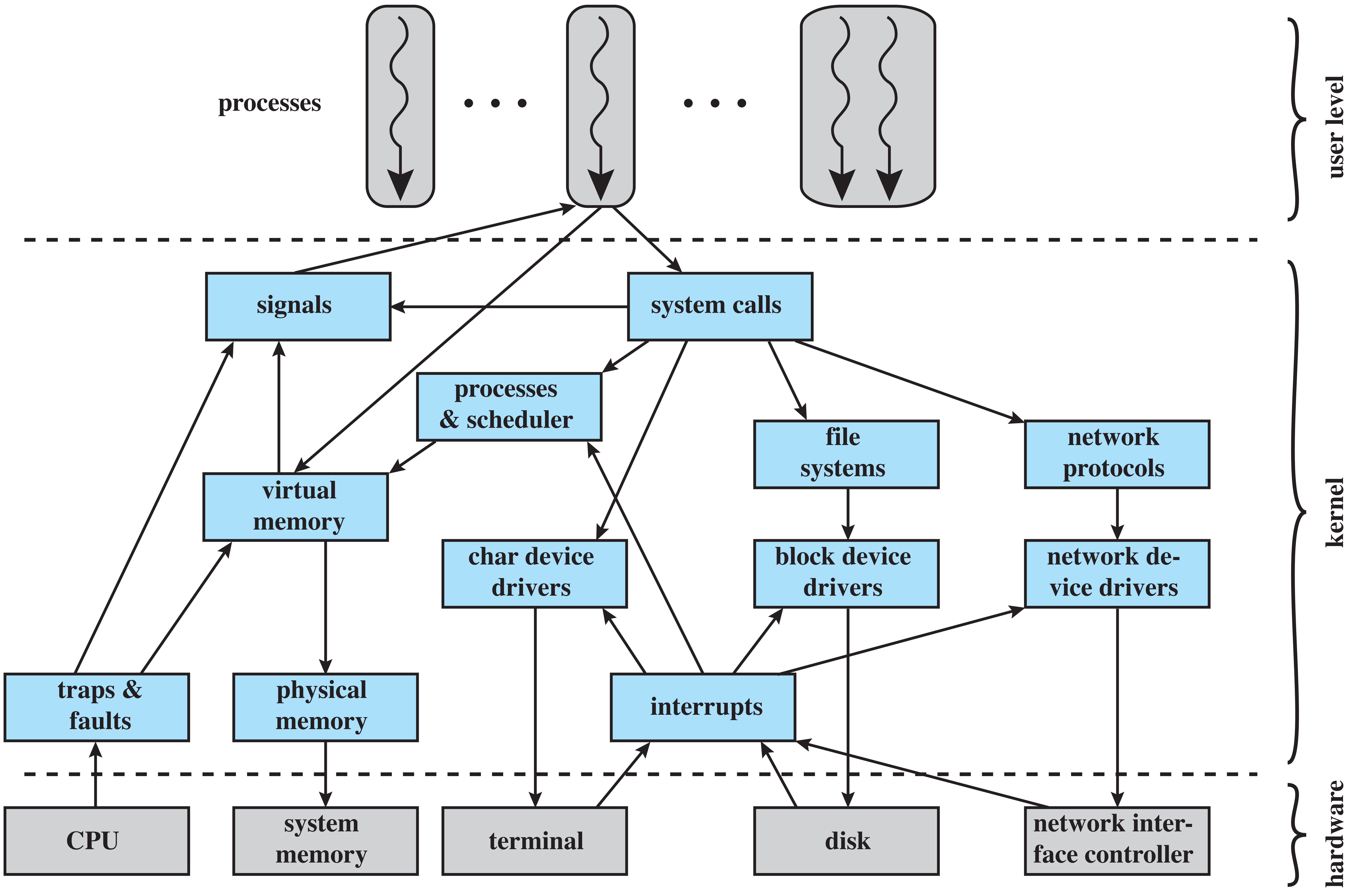


Figure 2.19 Linux Kernel Components

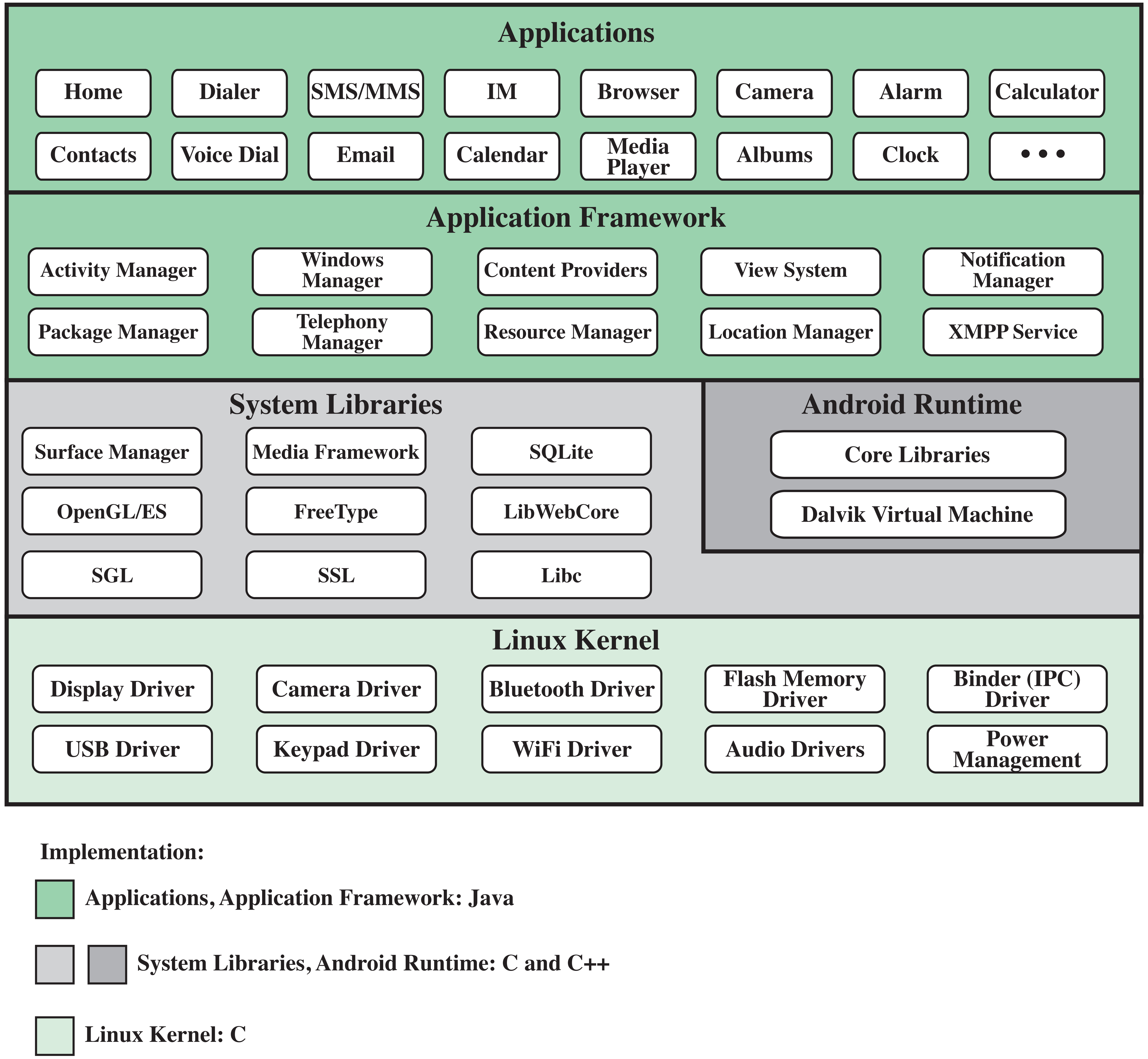


**Figure 2.20  Android Software Architecture**

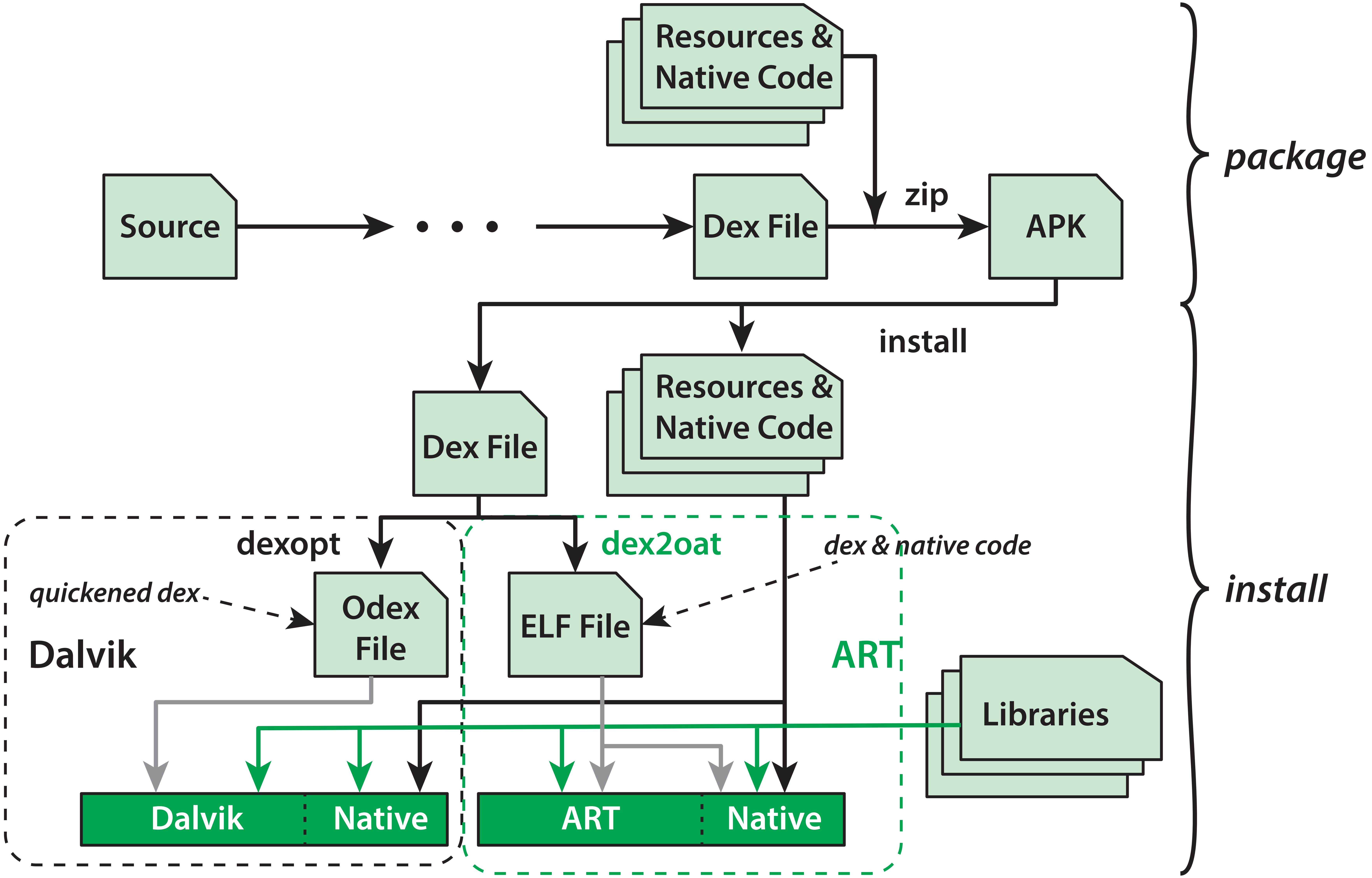


**Figure 2.21 The Life Cycle of an APK**

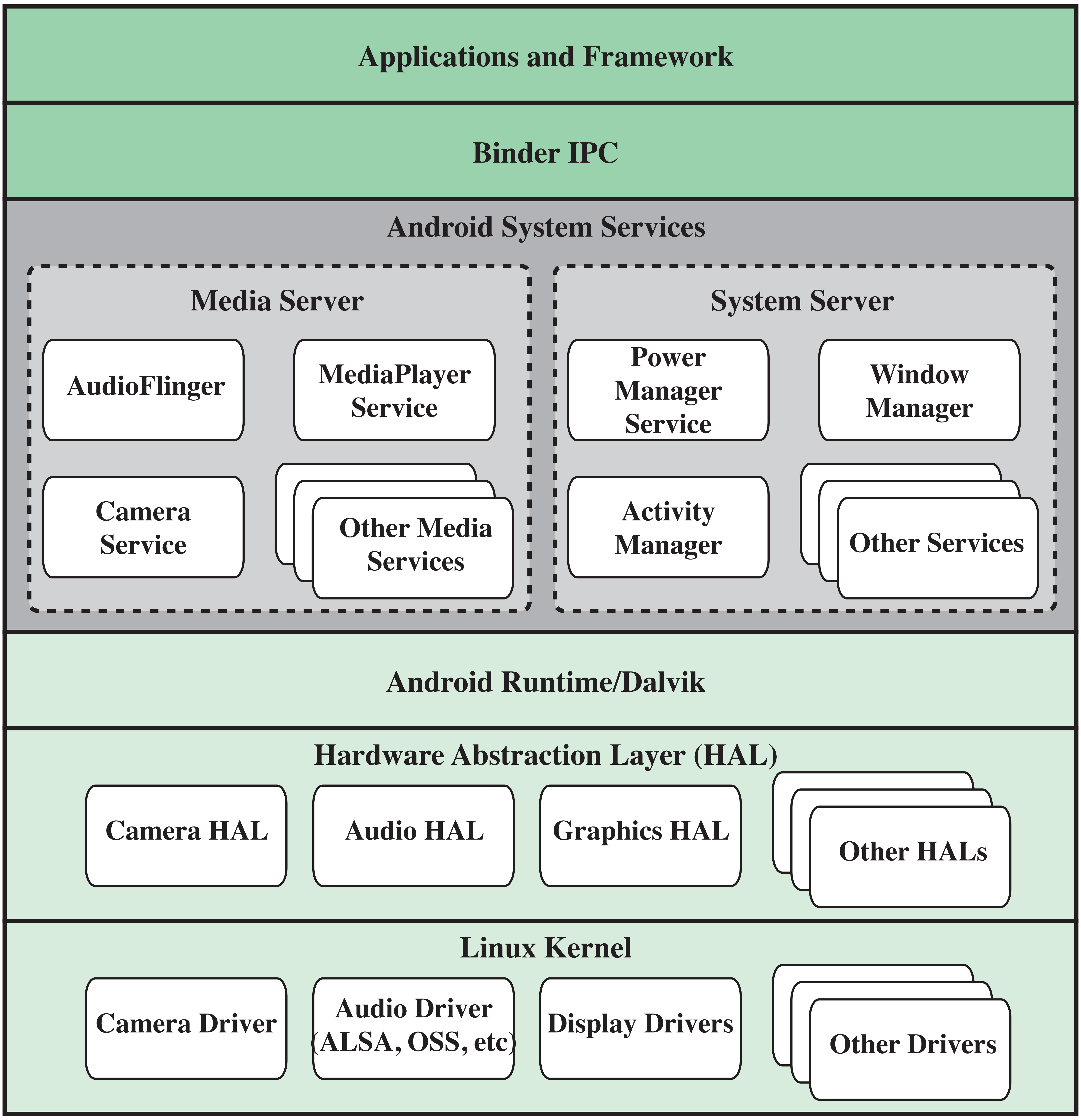


**Figure 2.22 Android System Architecture**