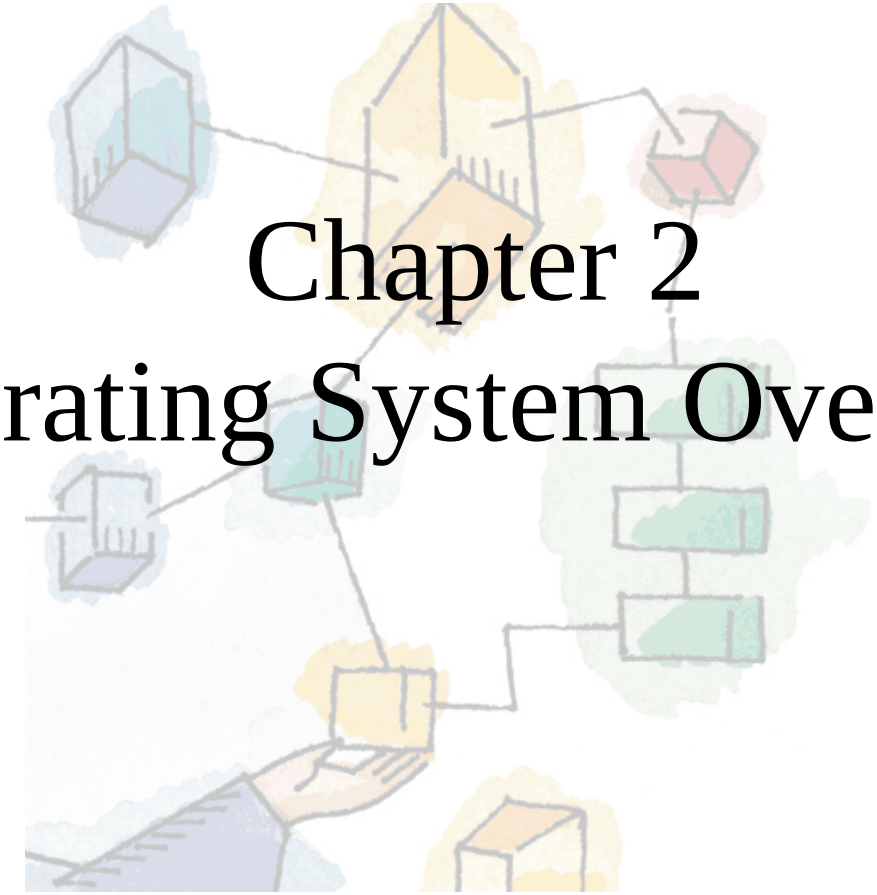
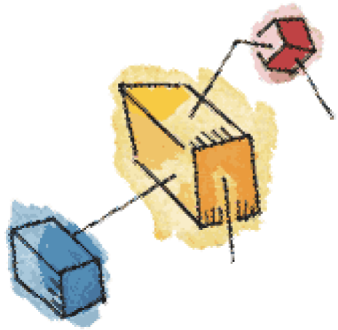
Operating Systems:
Internals and Design Principles, 8/E
William Stallings



Chapter 2
Operating System Overview

Patricia Roy
Manatee Community College, Venice, FL



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

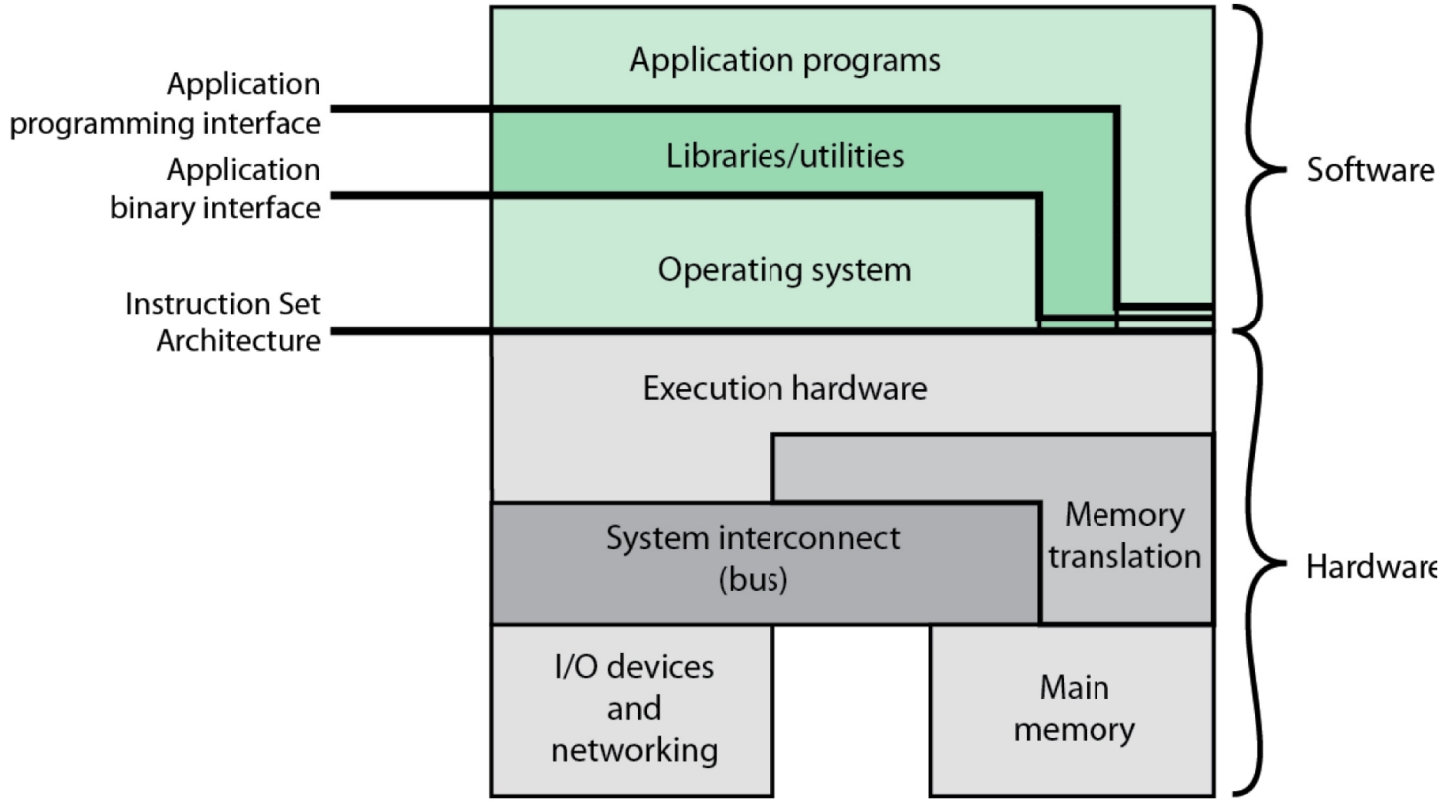
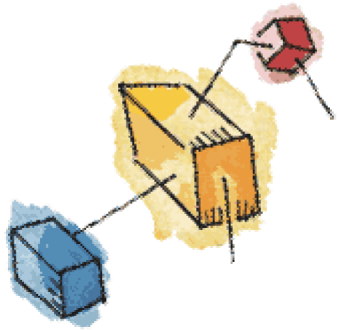
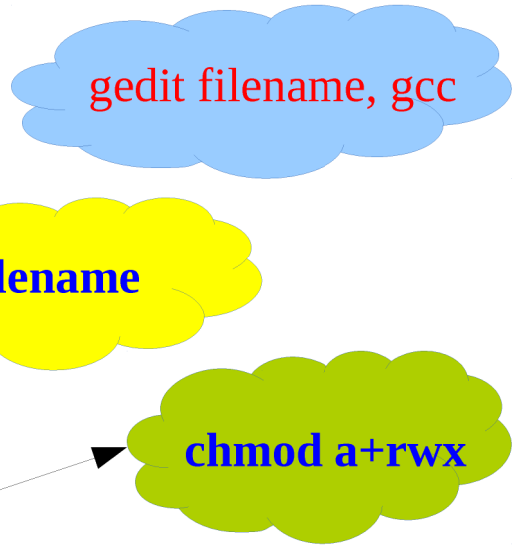


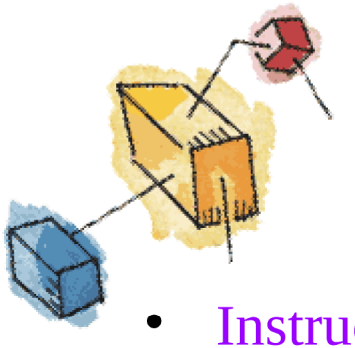
Figure 2.1 Computer Hardware and Software Infrastructure



Services Provided by the OS

- User/Computer Interface
  - Program development
    - Editors and debuggers
  - Program execution
  - Access I/O devices
  - Controlled access to files
  - System access
  - Error detection and response
  - Accounting





Key Interfaces of Computer

- Instruction Set Architecture (ISA)
  - Defines the repertoire of machine language instructions
  - Boundary between hardware and software
  - May be accessed directly
- Application Binary Interface (ABI)
  - Defines a standard for binary portability across programs
  - Defines the system call interface to the OS & hardware resources
- Application Programming Interface (API)
  - Gives a program access to the hardware resources & services
  - Any system call is perform through libraries

Services Provided by the OS

- Resources Manager
  - Functions same way as ordinary computer software
  - It is a program that is executed
  - Operating system relinquishes control of the processor
  - Kernel (nucleus)
    - Portion of operating system that is in main memory
    - Contains most frequently used functions

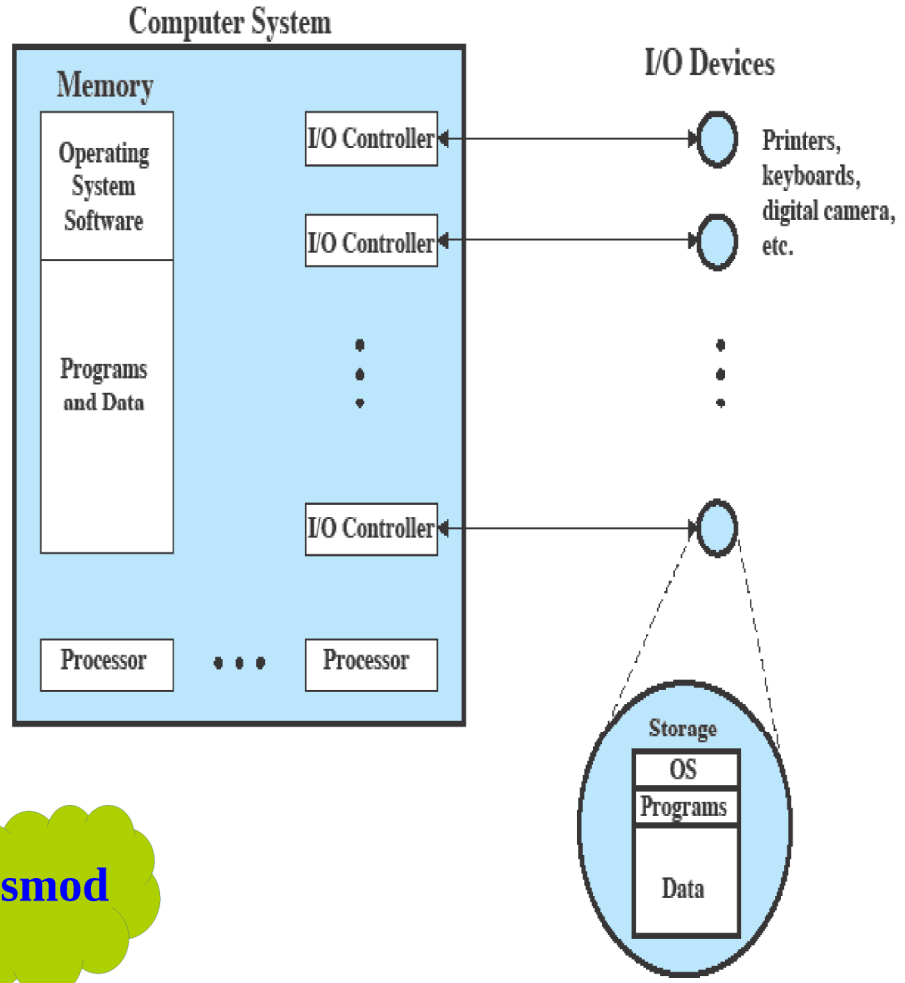
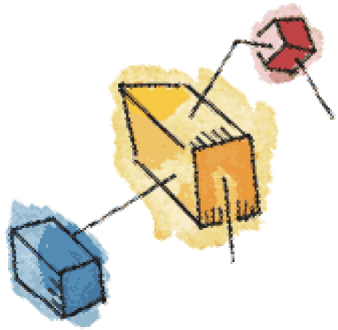


Figure 2.2 The Operating System as Resource Manager

lsmod

- shows which loadable kernel modules are currently loaded



Evolution of Operating Systems

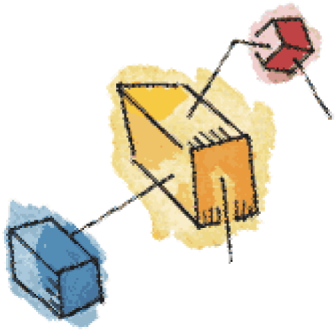
A major OS will evolve over time for a number of reasons:

Hardware Upgrades

New Hardware

New Services

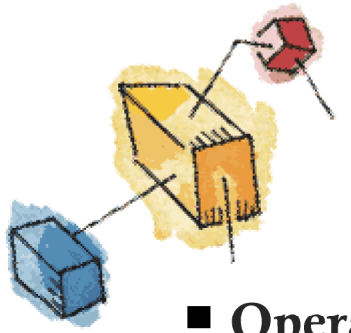
Fixes



Evolution of Operating Systems

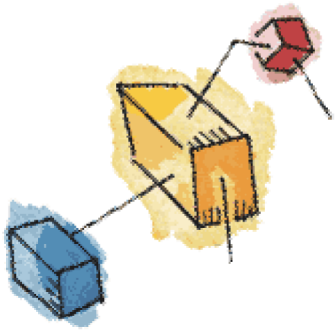
A) Serial processing

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



Summary

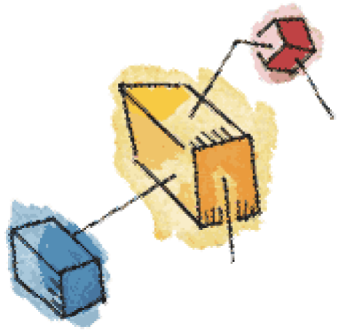
- **Operating system objectives and functions:**
  - convenience, efficiency, ability to evolve
  - user/computer interface
  - resource manager
  - Evolution:
    - serial processing, simple batch systems, multiprogrammed batch systems, time sharing systems
  - UNIX/Linux systems
  - Android systems and architecture
  - Common Linux command used for some operation in Linux



Evolution of Operating Systems

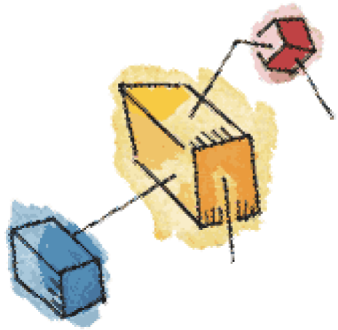
B) Simple batch system

- Monitor
  - Software that controls the sequence of events
  - Batch jobs together
  - Program returns control to monitor when finished
  - Job Control Language (JCL)
    - Provides instruction to the monitor
      - » What compiler to use
      - » What data to use



Batch System: Memory Protection

- Monitor executes in system mode
  - Kernel mode
  - Privileged instructions are executed
  - Protected areas of memory may be accessed



Evolution of Operating Systems

C) Multiprogrammed Batch System: (System Utilization Example)

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

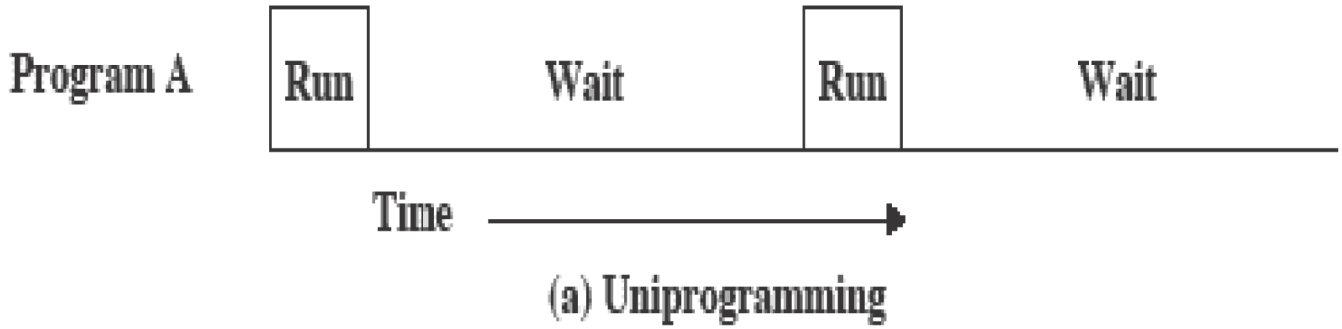
$$\text{Percent CPU Utilization} = \frac{1}{31} = 0.032 = 3.2\%$$

Related Command:
ps
top

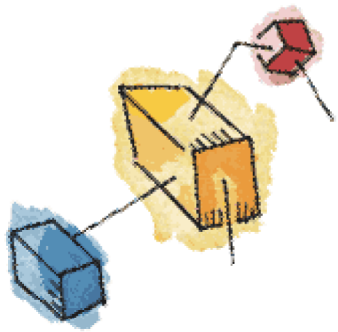
Figure 2.4 System Utilization Example

Uniprogramming

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

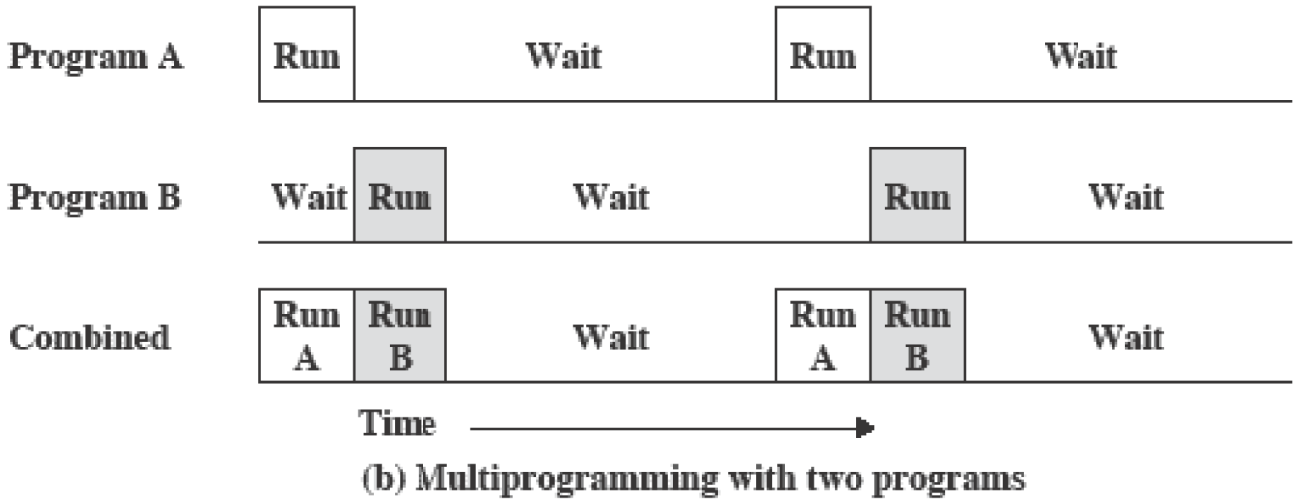


(a) Uniprogramming

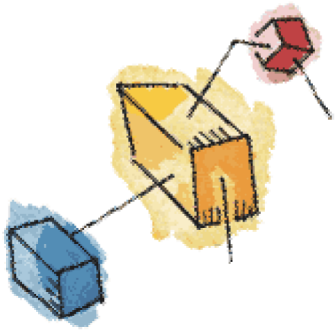


Multiprogramming

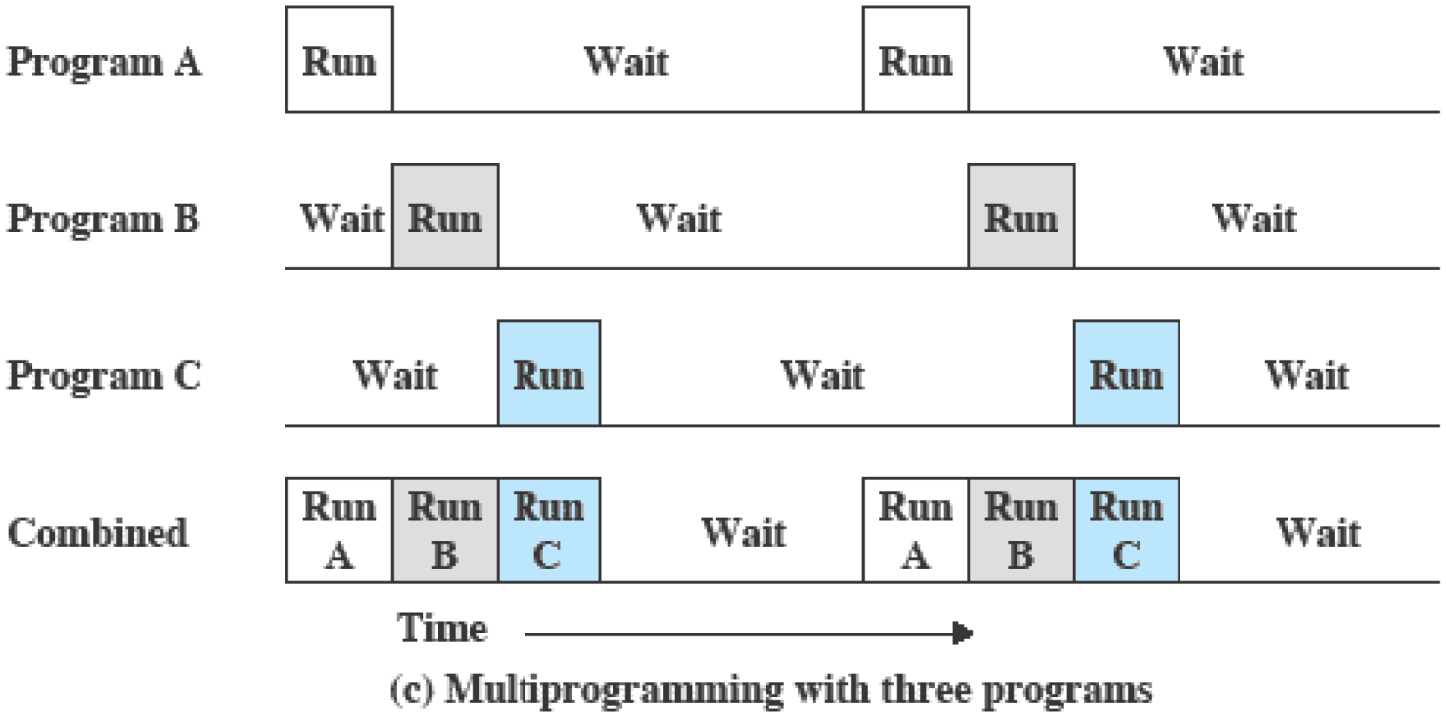
- When one job needs to wait for I/O, the processor can switch to the other job



(b) Multiprogramming with two programs



Multiprogramming



(c) Multiprogramming with three programs

Utilization Histograms

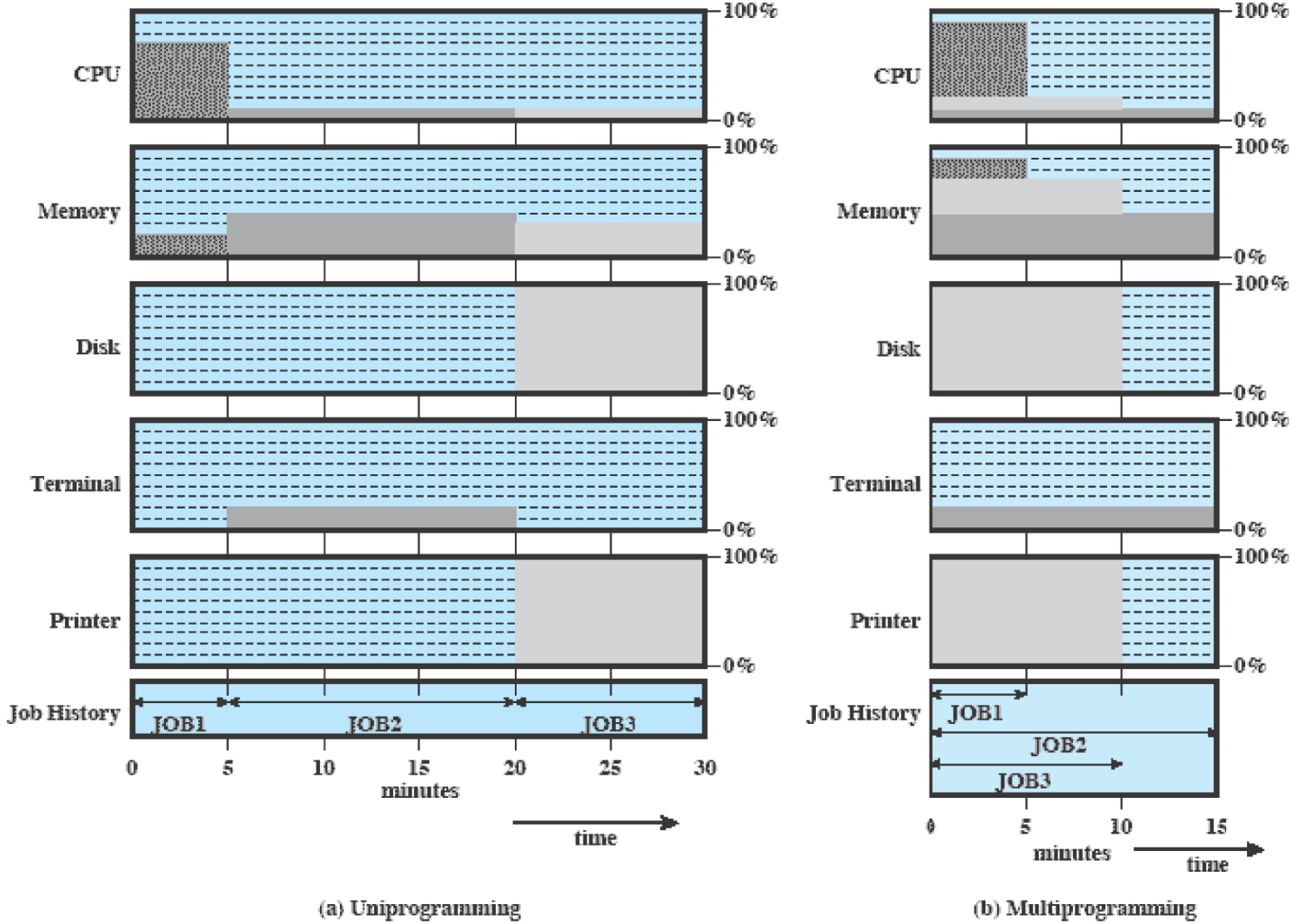
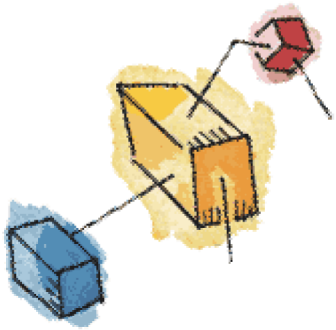


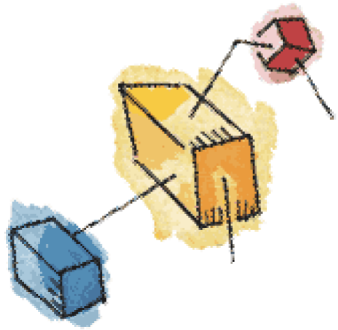
Figure 2.6 Utilization Histograms



Example

Table 2.1 Sample Program Execution Attributes

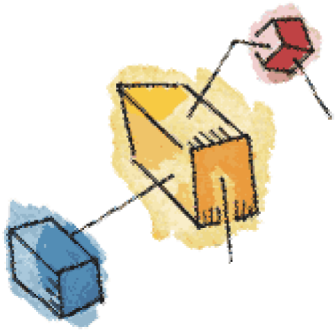
	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min	15 min	10 min
Memory required	50 M	100 M	75 M
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes



Evolution of Operating Systems

D) 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 versus Time Sharing

Table 2.3 Batch Multiprogramming versus 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

Compatible Time-Sharing System (CTSS) Operation

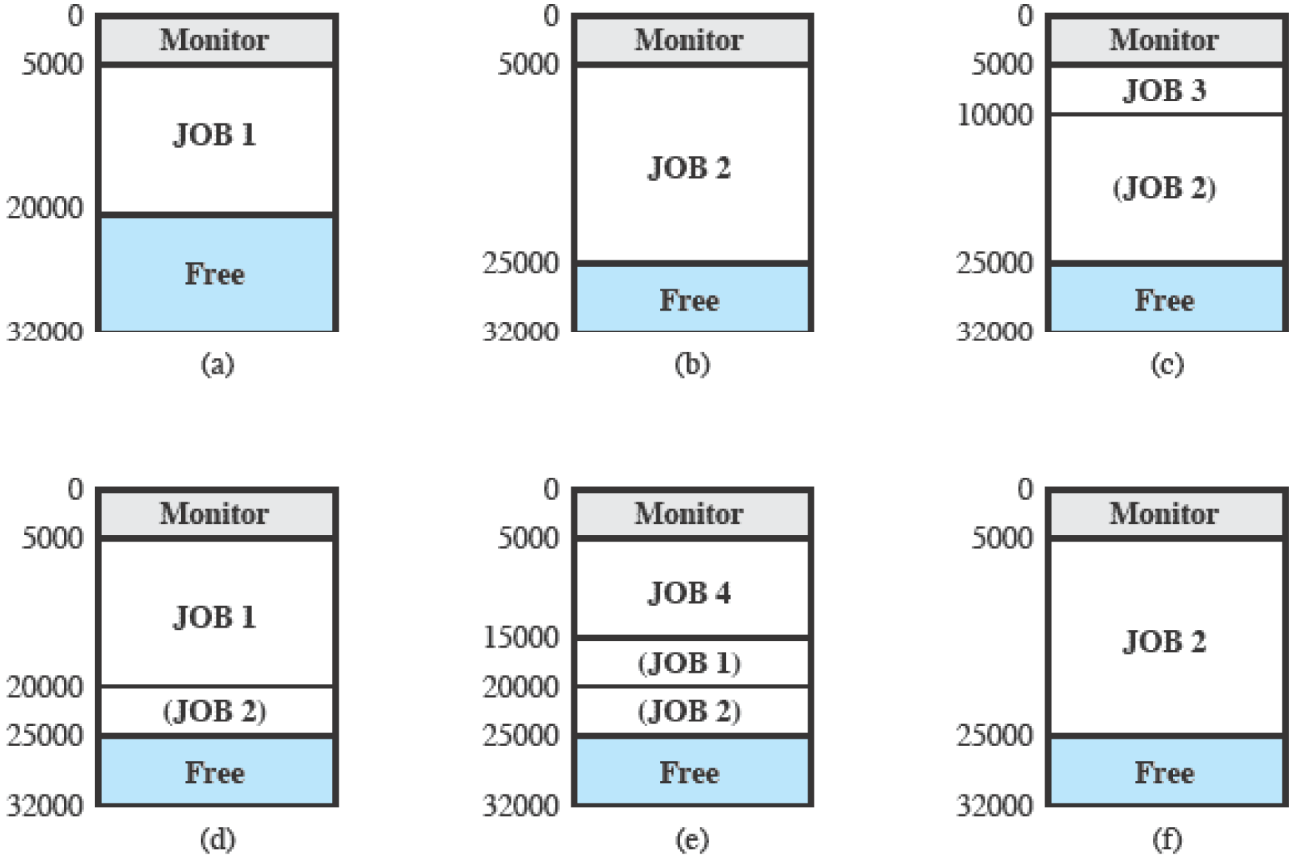
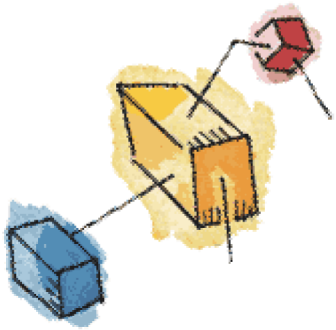
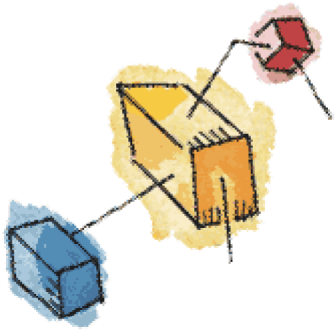


Figure 2.7 CTSS Operation



Major Achievements

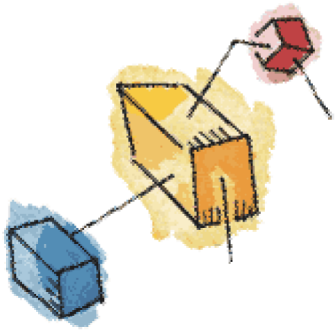
- Processes
- Memory management
- Information protection and security
- Scheduling and resource management
- System structure



The Process

Related Command:
ps -f
top
htop

- 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
  - An associated set of system resources



Difficulties with Designing System Software

- Improper synchronization
- Failed mutual exclusion
- Nondeterminate program operation
- Deadlocks

Solution

- Revised the current process:
  - Consists of three components
  - An executable program
  - Associated data needed by the program
  - Execution context of the program
    - All information the operating system needs to manage the process

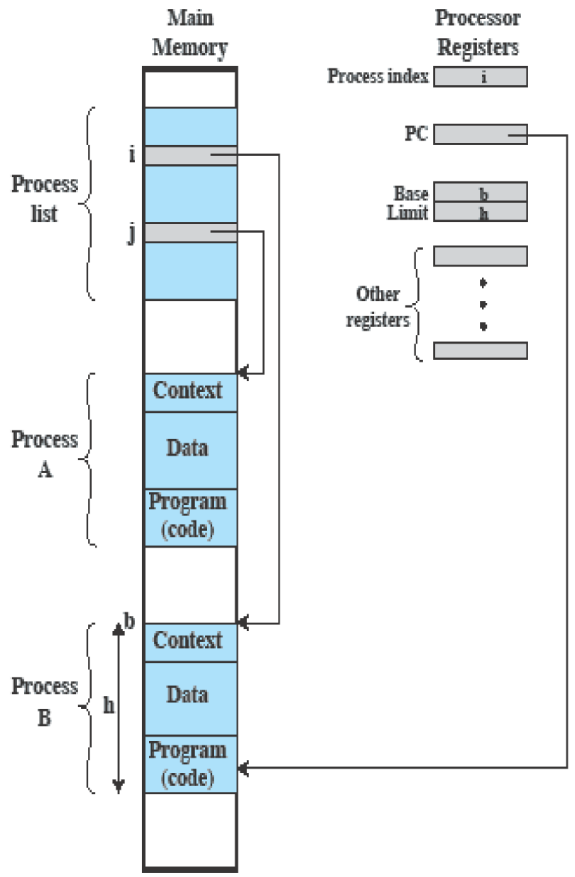
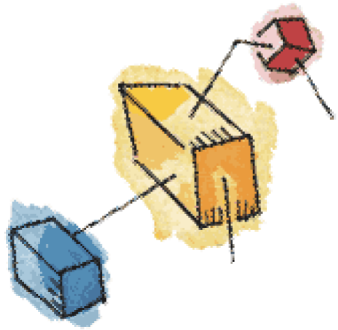


Figure 2.8 Typical Process Implementation

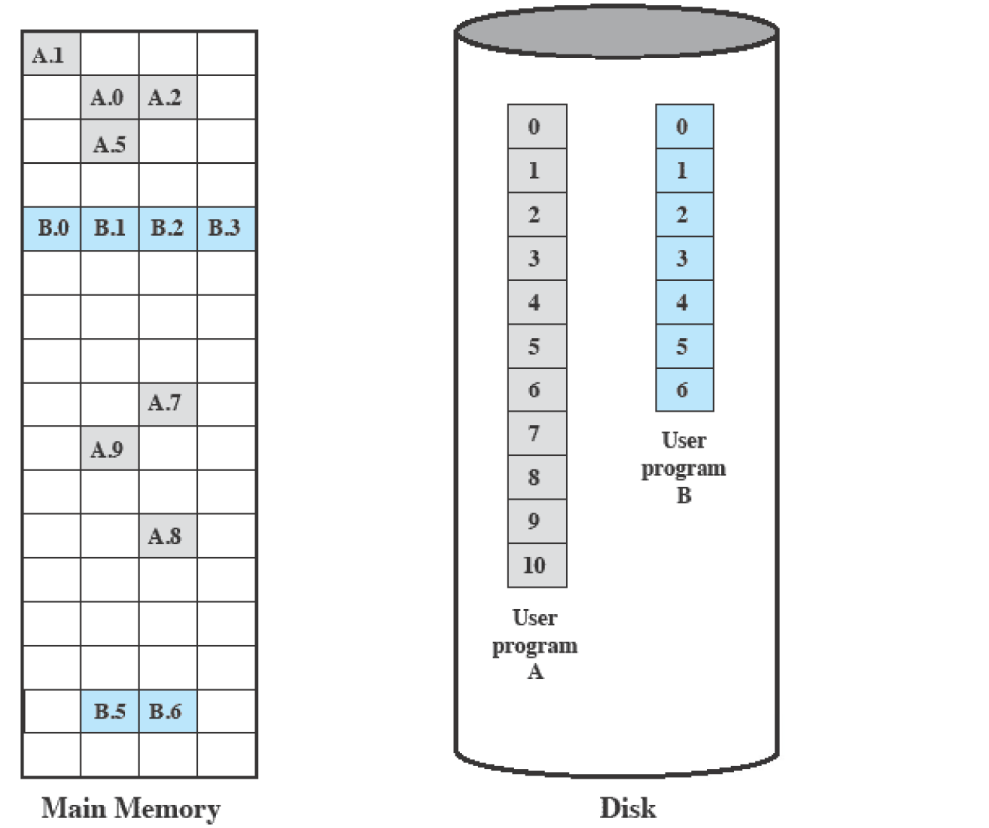


Memory Management

- Process isolation
- Automatic allocation and management
- Support of modular programming
- Protection and access control
- Long-term storage

Virtual Memory

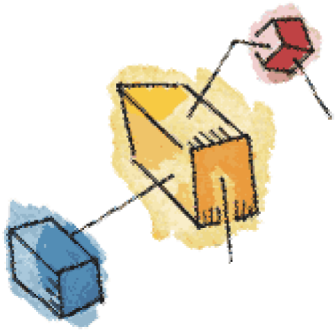
- Implements long-term store
- Information stored in named objects called files
- Allows programmers to address memory from a logical point of view



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.

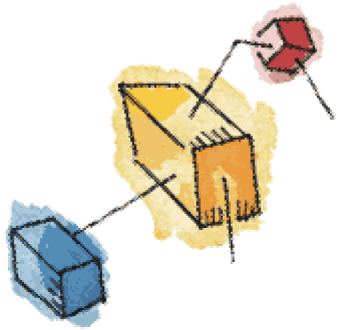
Secondary memory (disk) can hold many fixed-length pages. A user program consists of some number of pages. Pages for all programs plus the operating system are on disk, as are files.

Figure 2.9 Virtual Memory Concepts



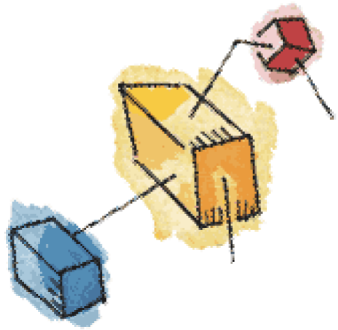
Paging System

- 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
- Real address or physical address in main memory



Information Protection and Security

- Availability
  - Concerned with protecting the system against interruption
- Confidentiality
  - Assuring that users cannot read data for which access is unauthorized
- Data integrity
  - Protection of data from unauthorized modification
- Authenticity



Scheduling and Resource Management

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

Key Elements of an Operating System

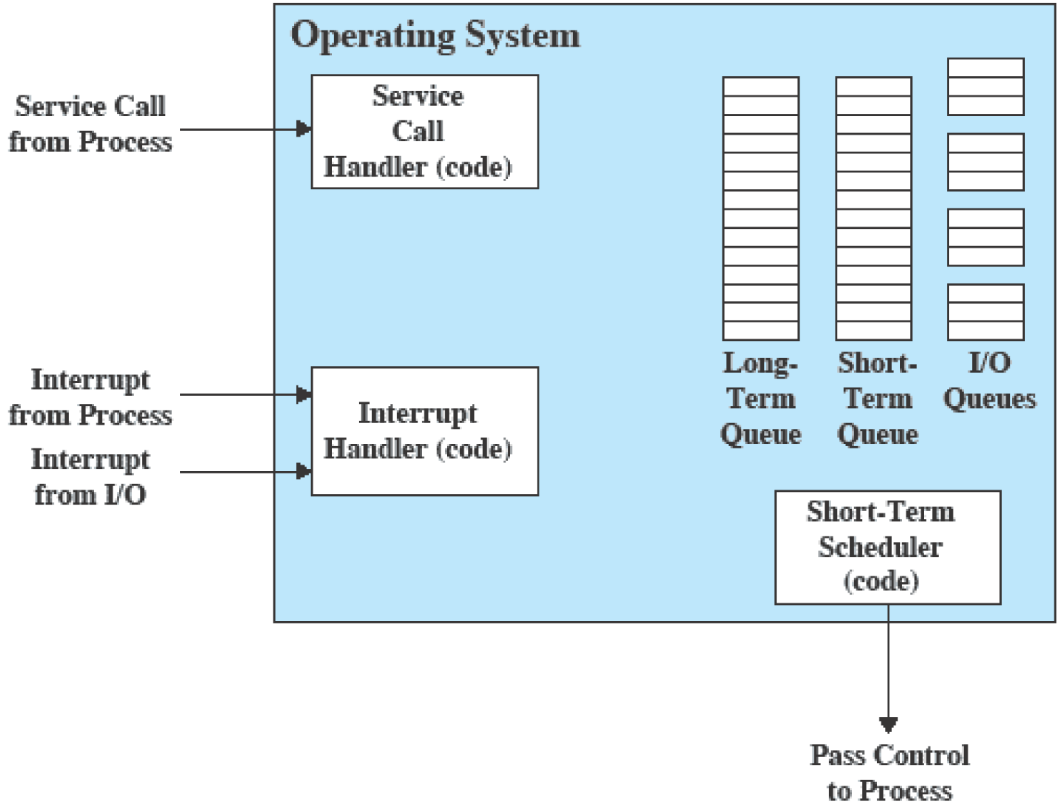
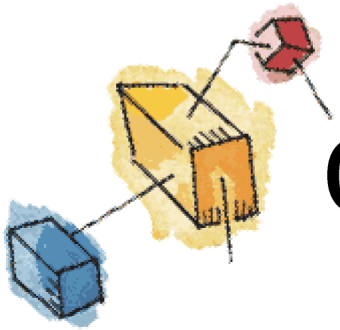
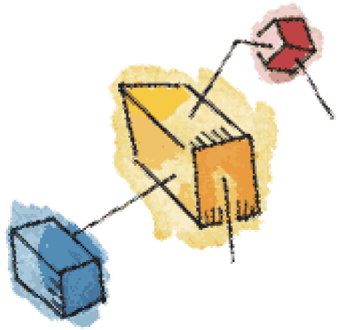


Figure 2.11 Key Elements of an Operating System for Multiprogramming



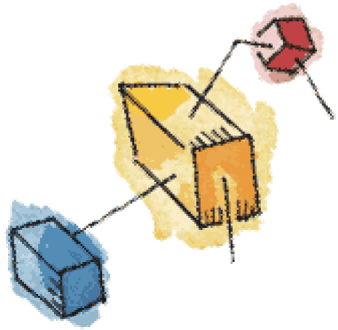
Modern Operating Systems Design

- Microkernel architecture
- Multithreading
- Symmetric Multiprocessing
- Distributed Operating System
- Object-Oriented Design



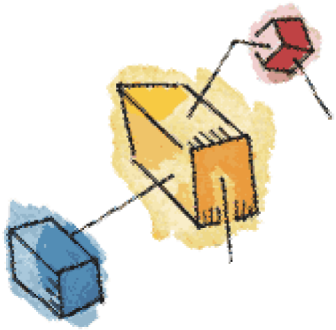
Modern Operating Systems

- Microkernel architecture
  - Assigns only a few essential functions to the kernel
    - Address spaces
    - Interprocess communication (IPC)
    - Basic scheduling



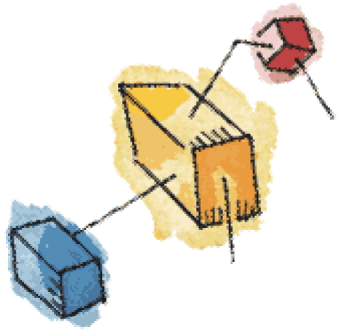
Modern Operating Systems

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



Modern Operating Systems

- Symmetric multiprocessing (SMP)
  - There are multiple processors
  - These processors share same main memory and I/O facilities
  - All processors can perform the same functions



Multiprogramming and Multiprocessing

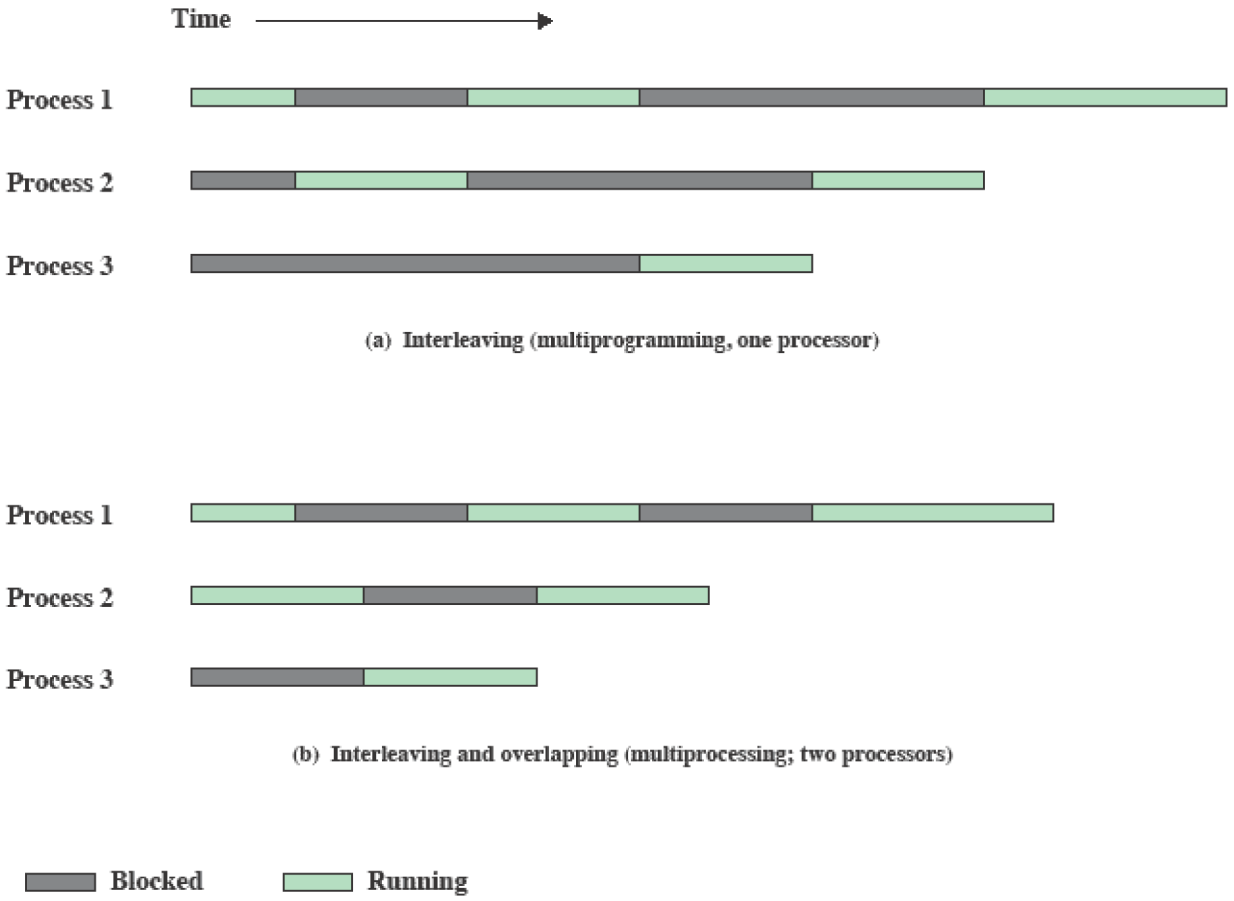
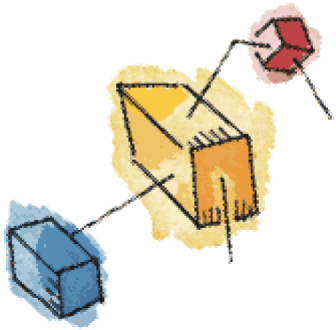


Figure 2.12 Multiprogramming and Multiprocessing



OS Design

Distributed Operating System

Provide the illustration of:
1. A single main memory space
2. Single Secondary main memory space
3. Unified access facilities

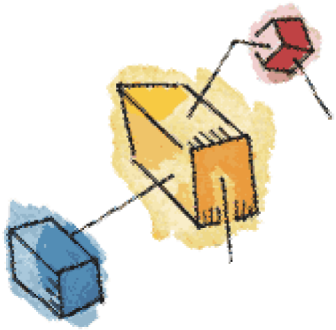
State-of-the-Art for distributed OS
lags that of uniprocessor
And SMP OS

Object-Oriented Design

Used for adding a modular extension to a small kernel

Enable users to customize an OS without disrupting system integrity

Ease the development of distributed tools And full-blown distributed OS



Virtual Machines and Virtualization

- Virtual Machine → software-based emulation of a computer
- Virtualization:
  - enables a single PC or server to simultaneously run multiple operating systems or multiple sessions of a single OS
  - a machine can host numerous applications, including those that run on different operating systems, on a single platform
  - host operating system can support a number of virtual machines (VM)
    - each has the characteristics of a particular OS and, in some versions of virtualization, the characteristics of a particular hardware platform

Virtual Machines Concept

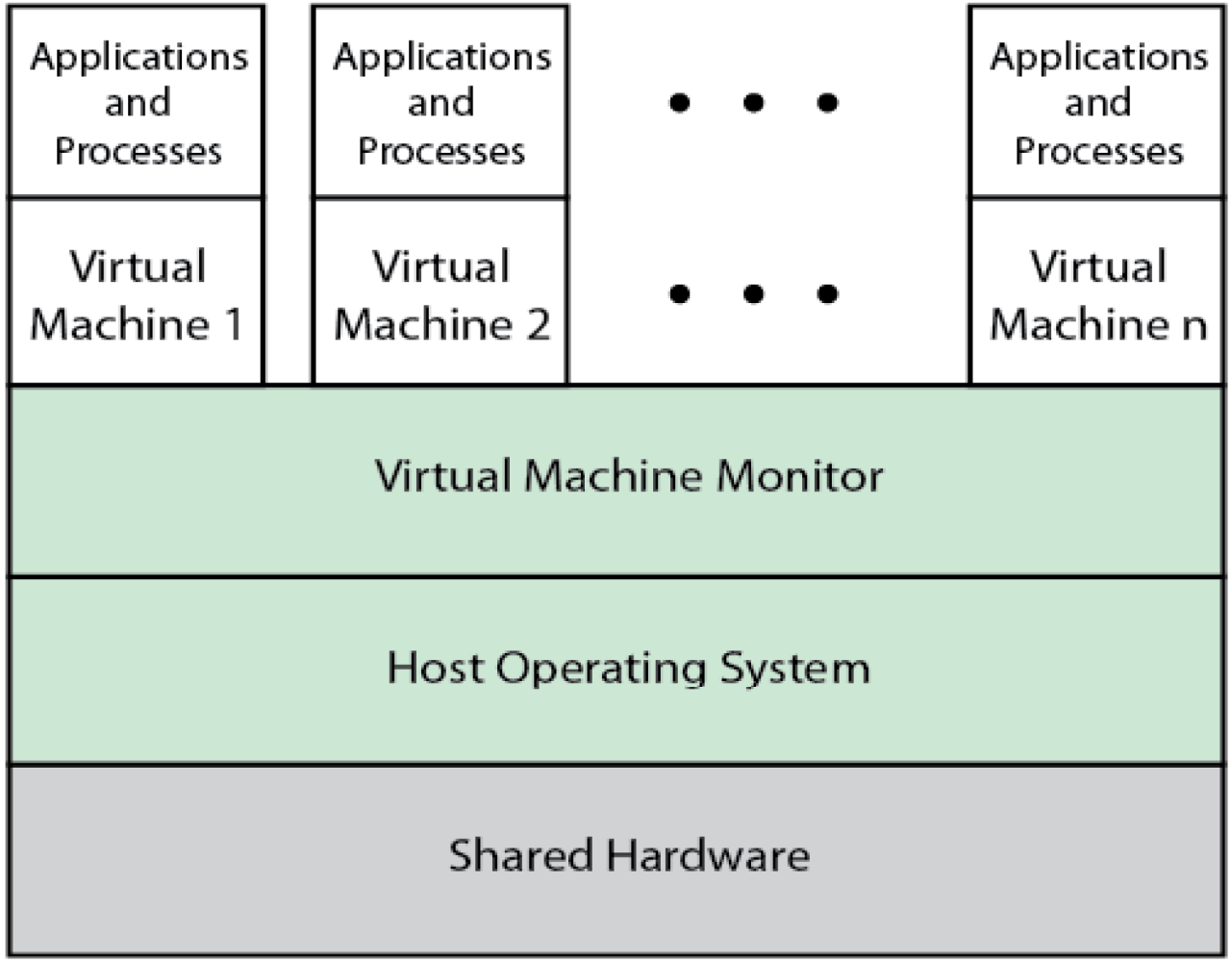
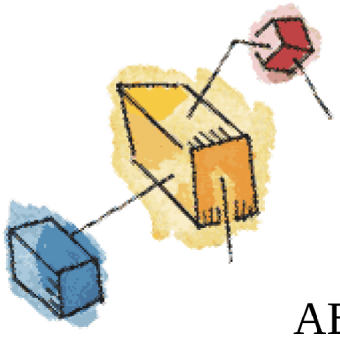


Figure 2.13 Virtual Memory Concept



VM Architecture

ABI

Process point-of-view:

A machine consists of :
1. VM space assigned to a process
2. Processor Register
3. User-Level Machine Instruction
4. OS system Call

API

Application point-of-view:

Machine characteristic are specified by:

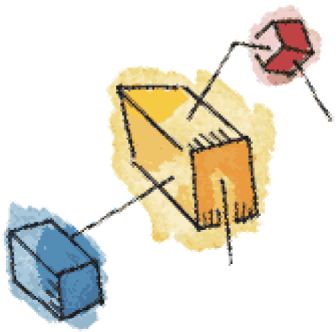
1. High-level language capabilities
2. OS
3. System library call

ISA

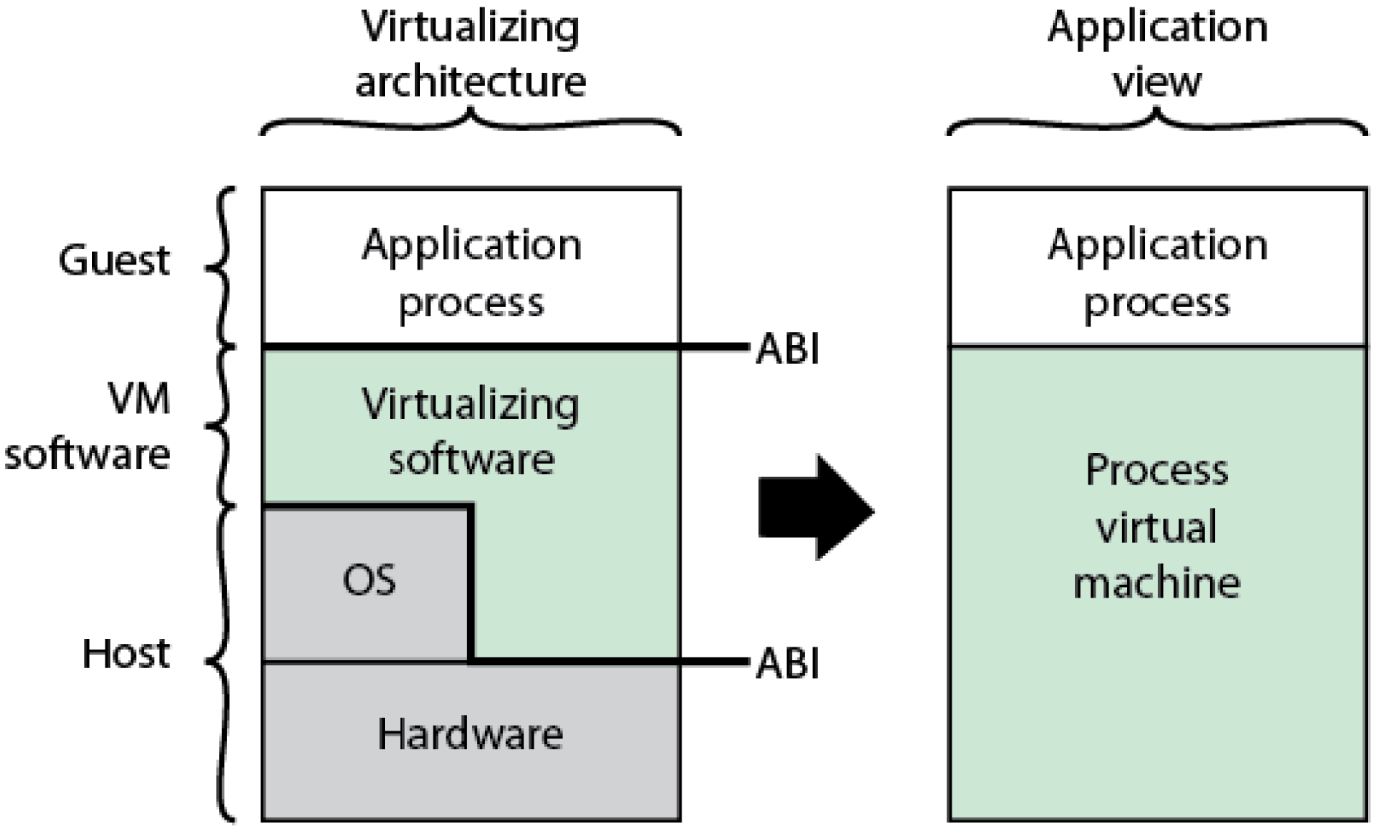
From OS perspective:
Machine hardware defines the system that support the OS

The system allocate the real memory & I/O resources to the Process

ISA provide an interface Between system & machine



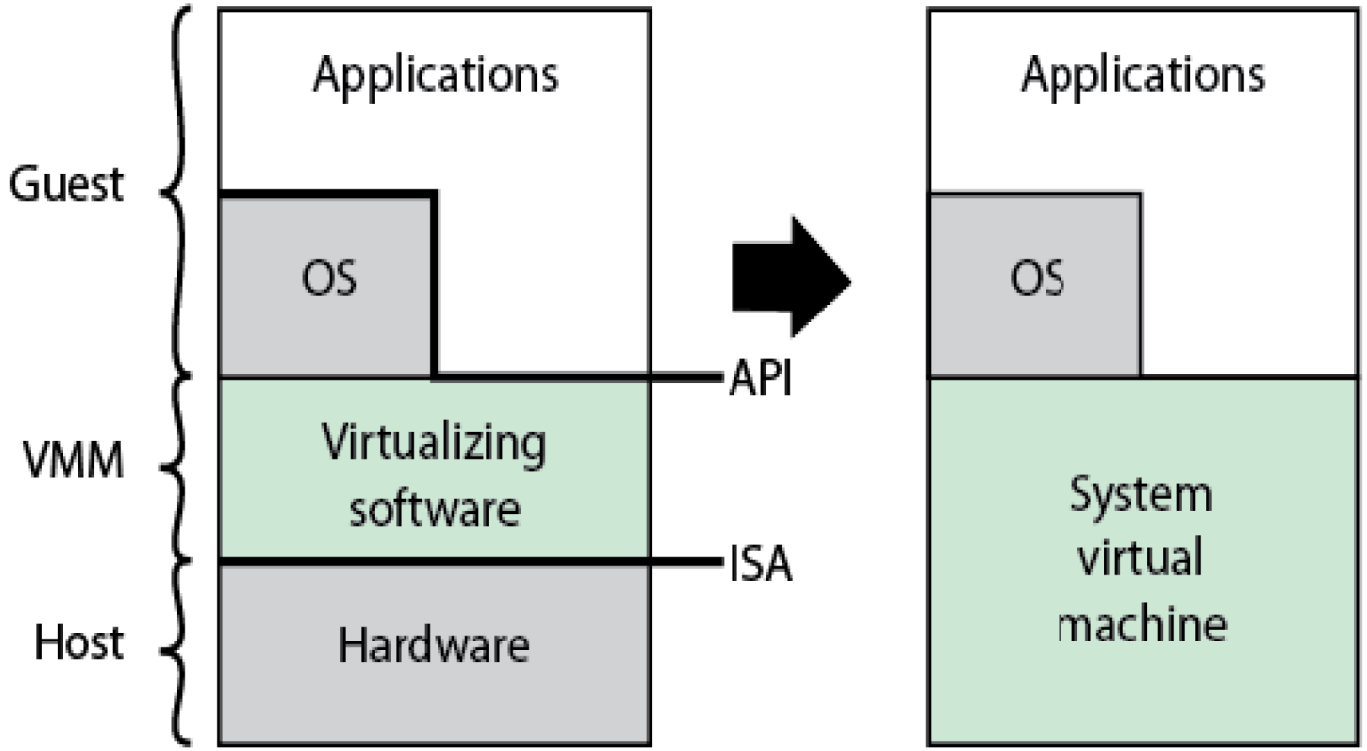
Process & System VM



(a) Process VM

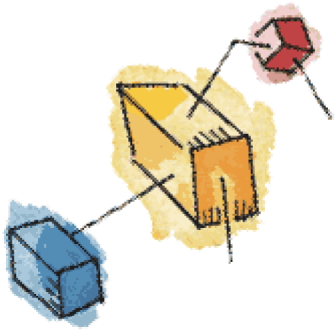
Figure 2.14 Process and System Virtual Machines

Process & System VM



(b) System VM

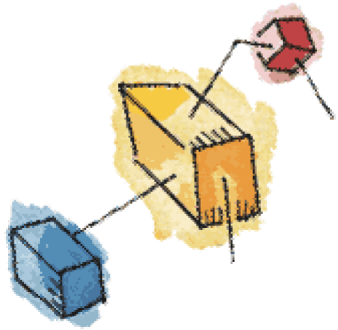
Figure 2.14 Process and System Virtual Machines



Symmetric Multiprocessor OS Considerations

A multiprocessor OS must provide all the functionality of a multiprogramming system plus additional features to accommodate multiple processors

- Key Design Issue:
  - Simultaneous concurrent processes or thread
  - Scheduling
  - Synchronization
  - Memory Management
  - Reliability and fault tolerance

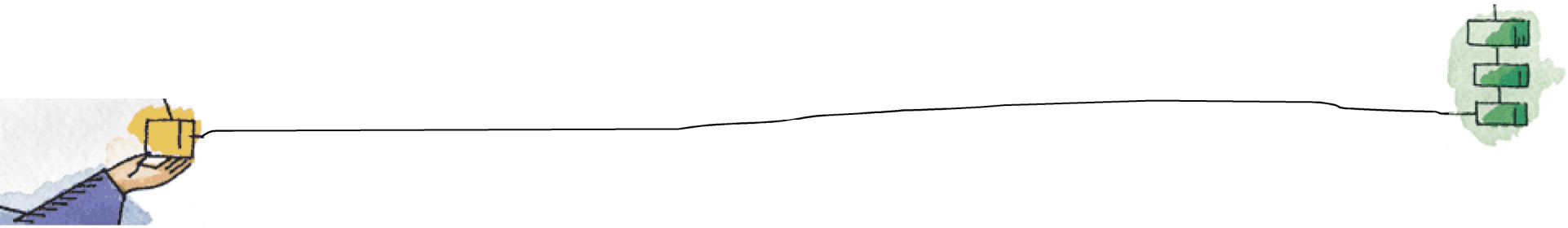


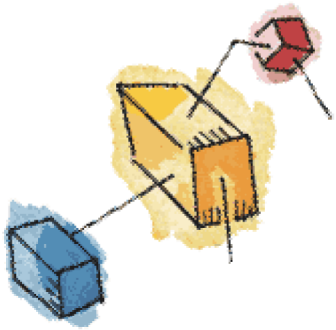
Multicore OS Consideration

The design challenge for a many-core multicore system is to efficiently harness the multicore processing power and intelligently manage the substantial on-chip resources efficiently

Potential for parallelism exists at three levels:

- Hardware parallelism within each core processor
- May or may not be exploited by application program
- There is a potential for multi-programming and multi-thread execution within each processor
- There is a potential for a single application to execute in concurrent processes or thread across multiple cores





Modern Operating Systems

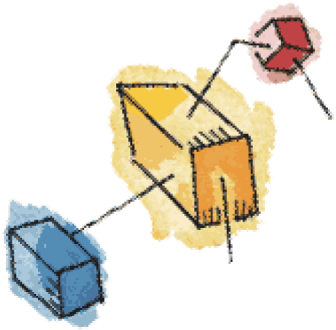
Parallelism within Application:

- Multiple task, execute in parallel, multiple processes & thread
- challenge: developer must decide how to split-up the application
- The most effective initiative: UNIX-based MacOS XOS
- E.g: MacOS X 10.6 includes a multicore support known as Grand Central Dispatch (GCD). The GCD make the developer's task easier after the developer has identified which has to be split-off into a separate tasks.

VM Approach:

Allows one or more cores to be dedicated to a particular process and then leave the processor alone to devote its efforts to that process

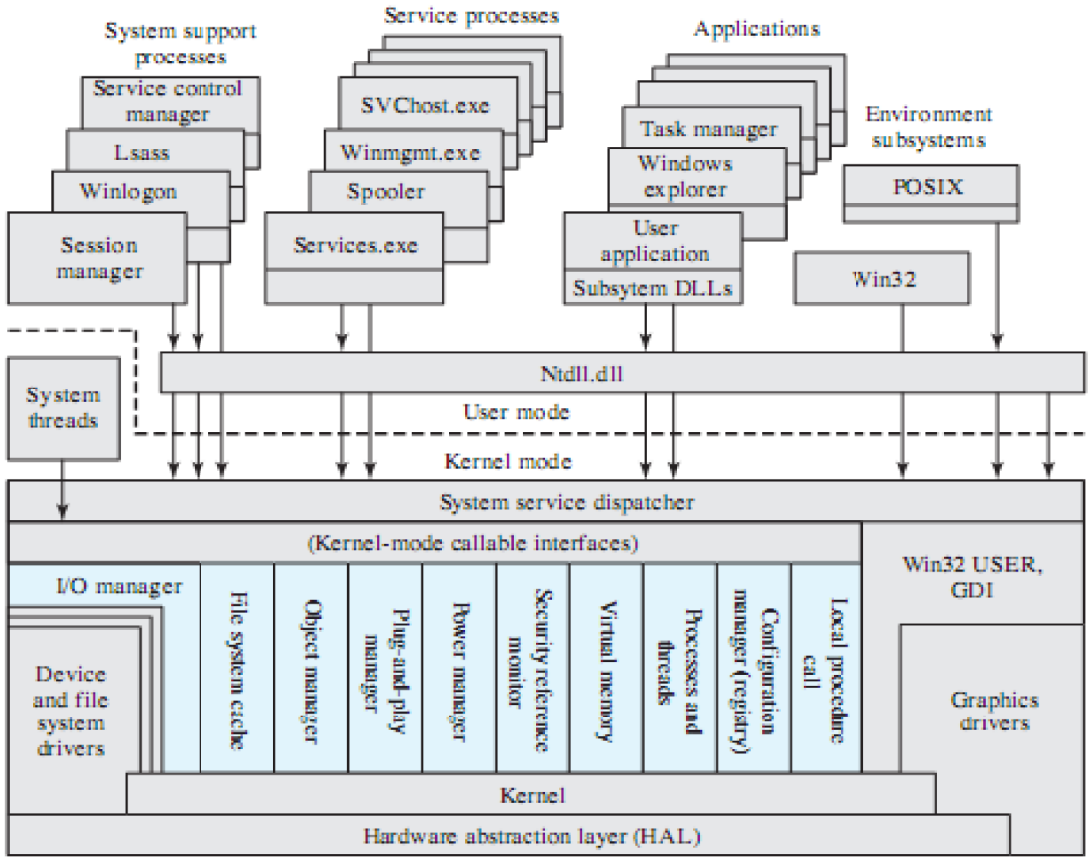
Multicore OS could then act as a hypervisor that makes a high-level decision to allocate cores to applications but does little in the way of resource allocation beyond that



Modern Operating Systems

- Distributed operating systems
  - Provides the illusion of a single main memory space and single secondary memory space
- Object-oriented design
  - Used for adding modular extensions to a small kernel
  - Enables programmers to customize an operating system without disrupting system integrity

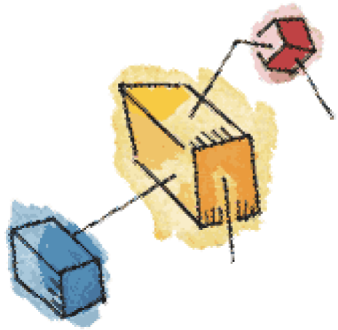
Modern OS Architecture



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

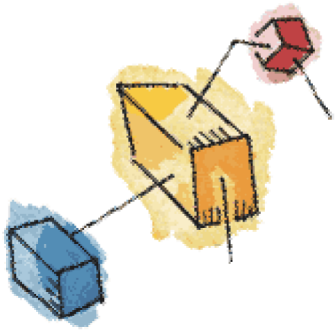
Colored area indicates Executive

Figure 2.13 **Windows and Windows Vista Architecture [RUSS05]**



Traditional UNIX System

- Were developed at Bell Labs and became operational on a PDP-7 in 1970
- Incorporated many ideas from Multics
- PDP-11was a milestone because it first showed that UNIX would be an OS for all computers
- Next milestone was rewriting UNIX in the programming language C
  - demonstrated the advantages of using a high-level language for system code
  - Was described in a technical journal for the first time in 1974
  - First widely available version outside Bell Labs was Version 6 in 1976
  - Version 7, released in 1978 is the ancestor of most modern UNIX systems
  - Most important of the non-AT&T systems was UNIX BSD (Berkeley Software Distribution)



Unix System

- Programmers platform to develop software
- Later becomes OS
- Features:
  - Portable
  - Multi-tasking
  - Multi-users
  - Time-sharing configurations

Types of Unix

Sun Solaris

GNU/Linux

MacOS

UNIX Architecture

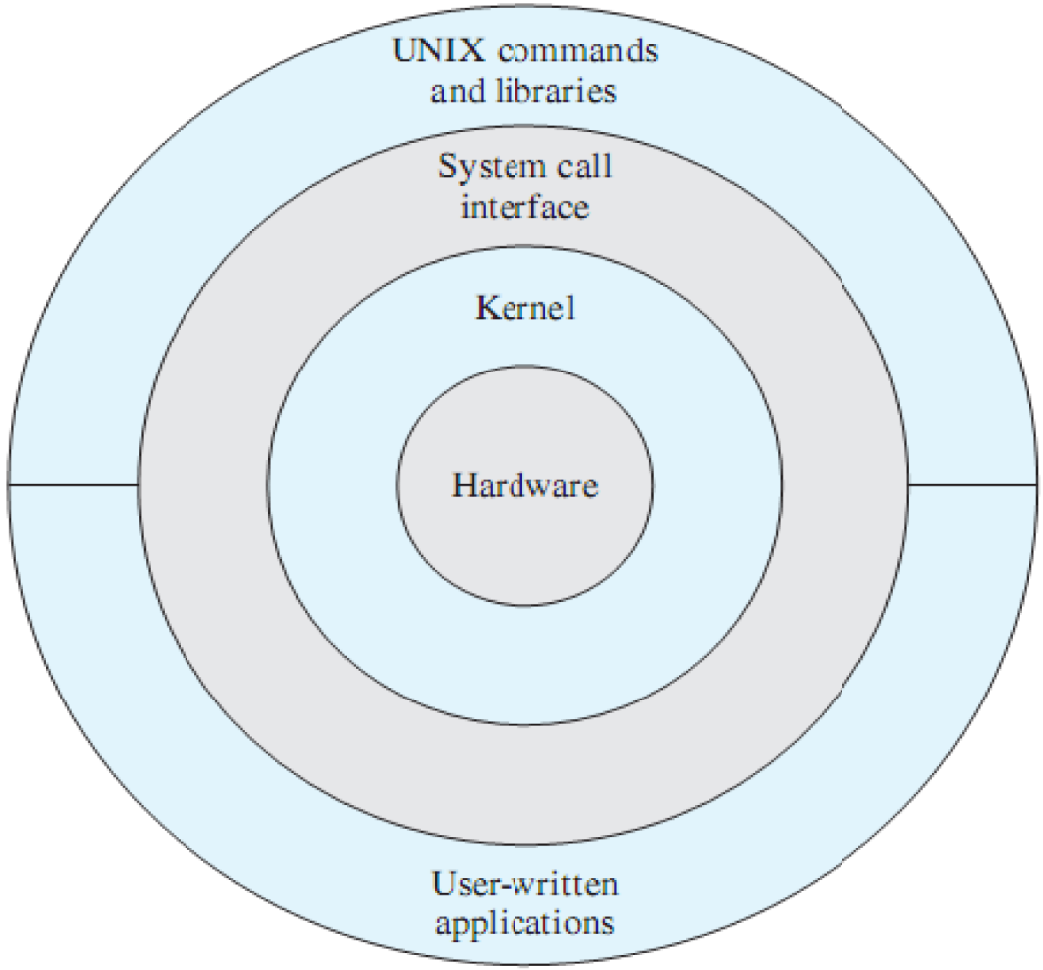
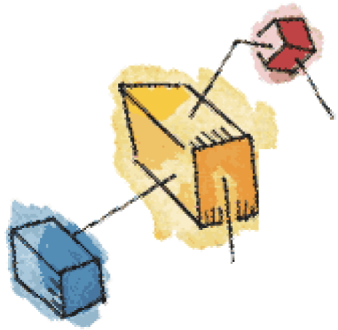


Figure 2.14 General UNIX Architecture



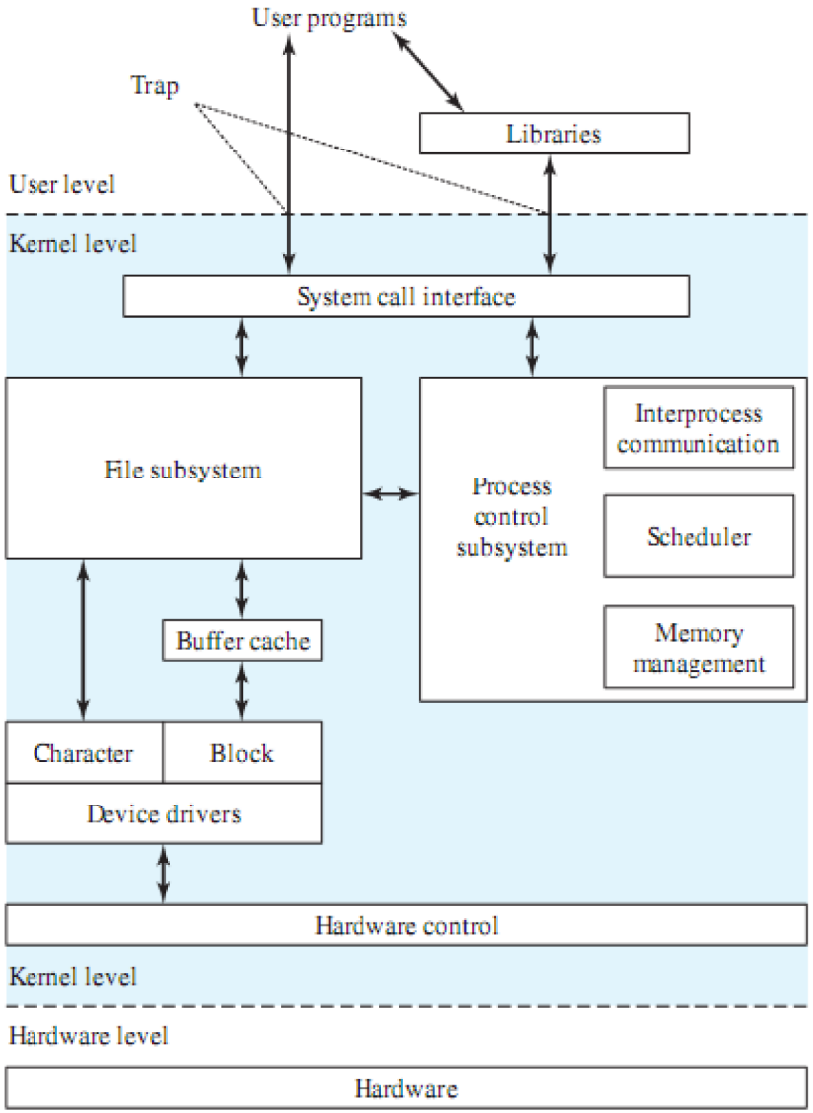


Figure 2.15 **Traditional UNIX Kernel**

Traditional UNIX Kernel

- Hub of the program
- Allocates time and memory to program
- Handles file store & communication

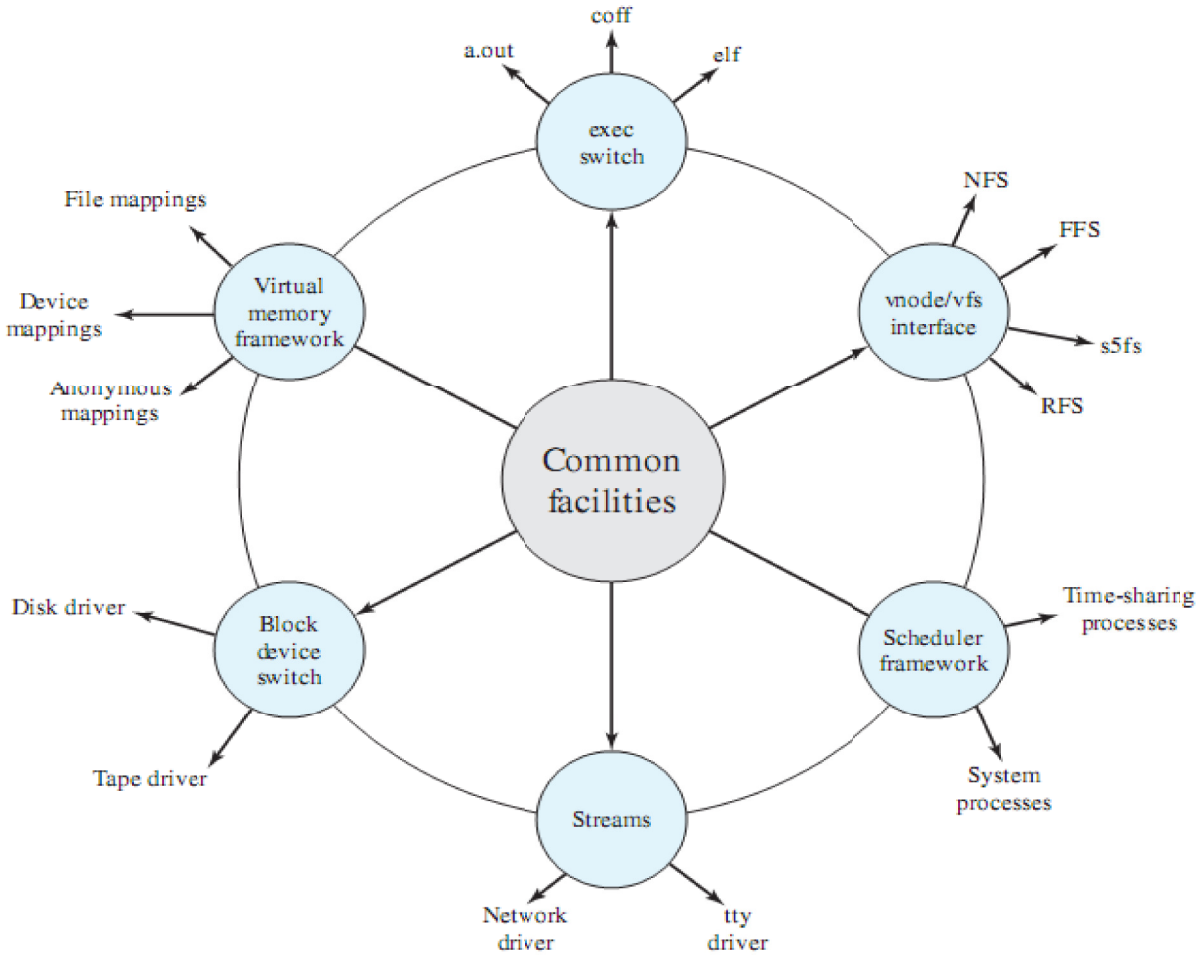
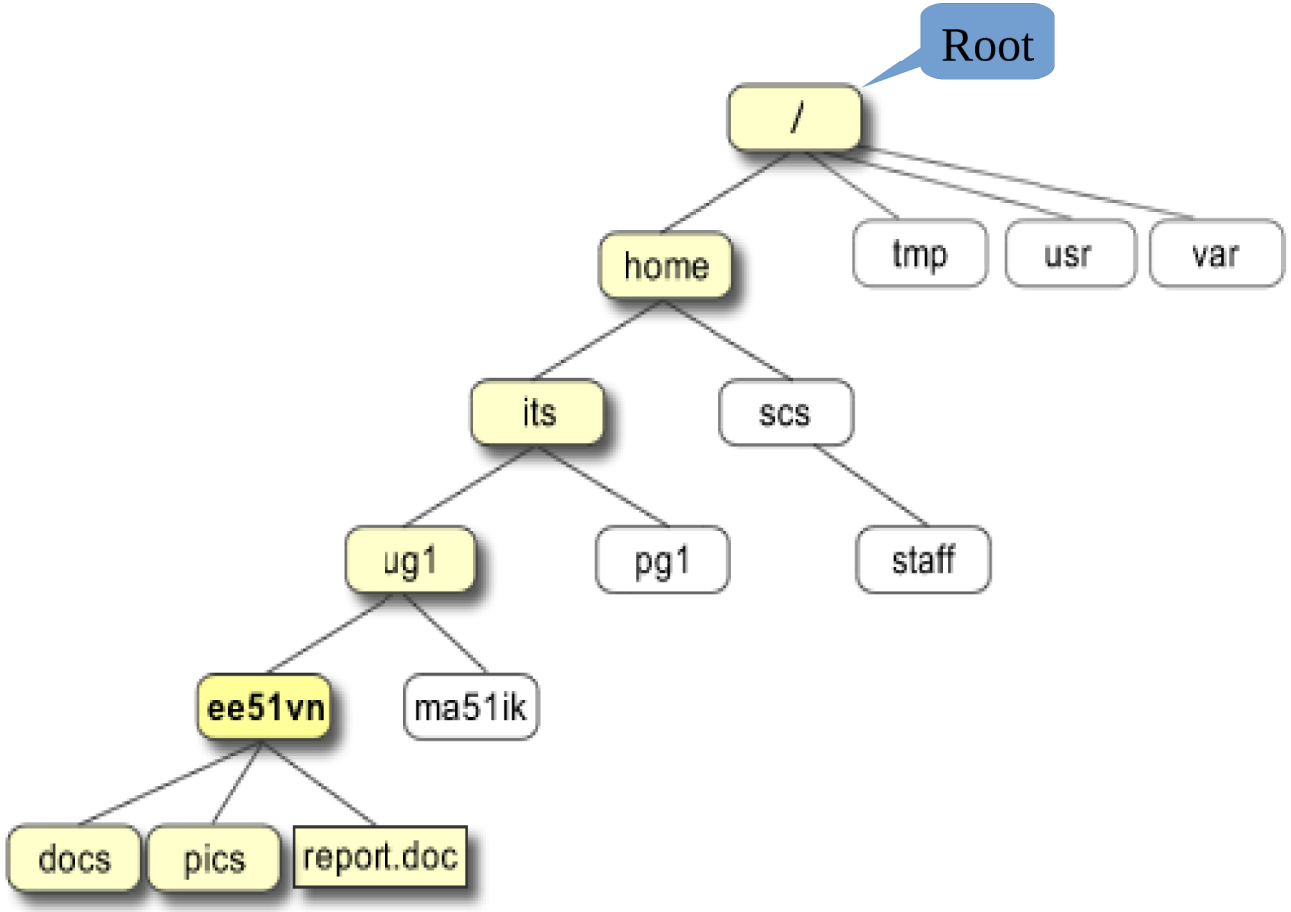


Figure 2.16 **Modern UNIX Kernel**

Modern UNIX Kernel

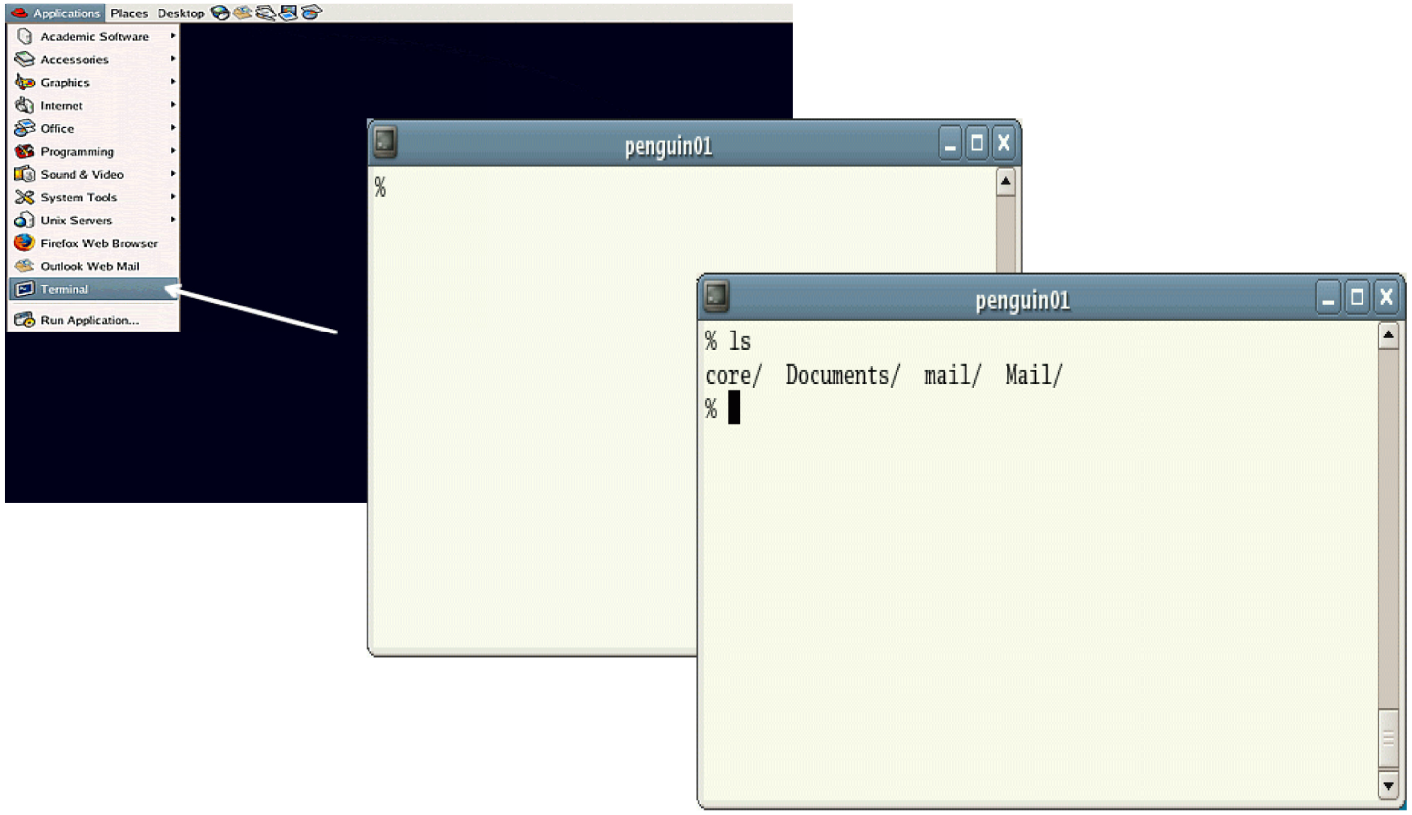
SVR4
BSD
Solaris 10

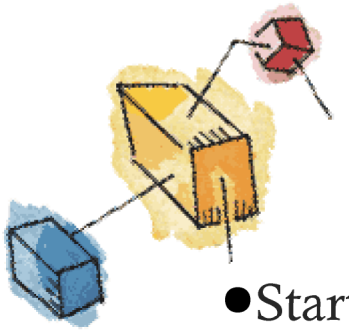
Unix: Directory Structure



/home/its/ug1/ee51vn/report.doc

Starting a UNIX Terminal





Linux

- Started out as a UNIX variant for the IBM PC
- Linus Torvalds, a Finnish student of computer science, wrote the initial version
- Linux was first posted on the Internet in 1991
- Today it is a full-featured UNIX system that runs on several platforms
- Is free and the source code is available
- Key to success has been the availability of free software packages
- Highly modular and easily configured

List of Linux Kernel Modules

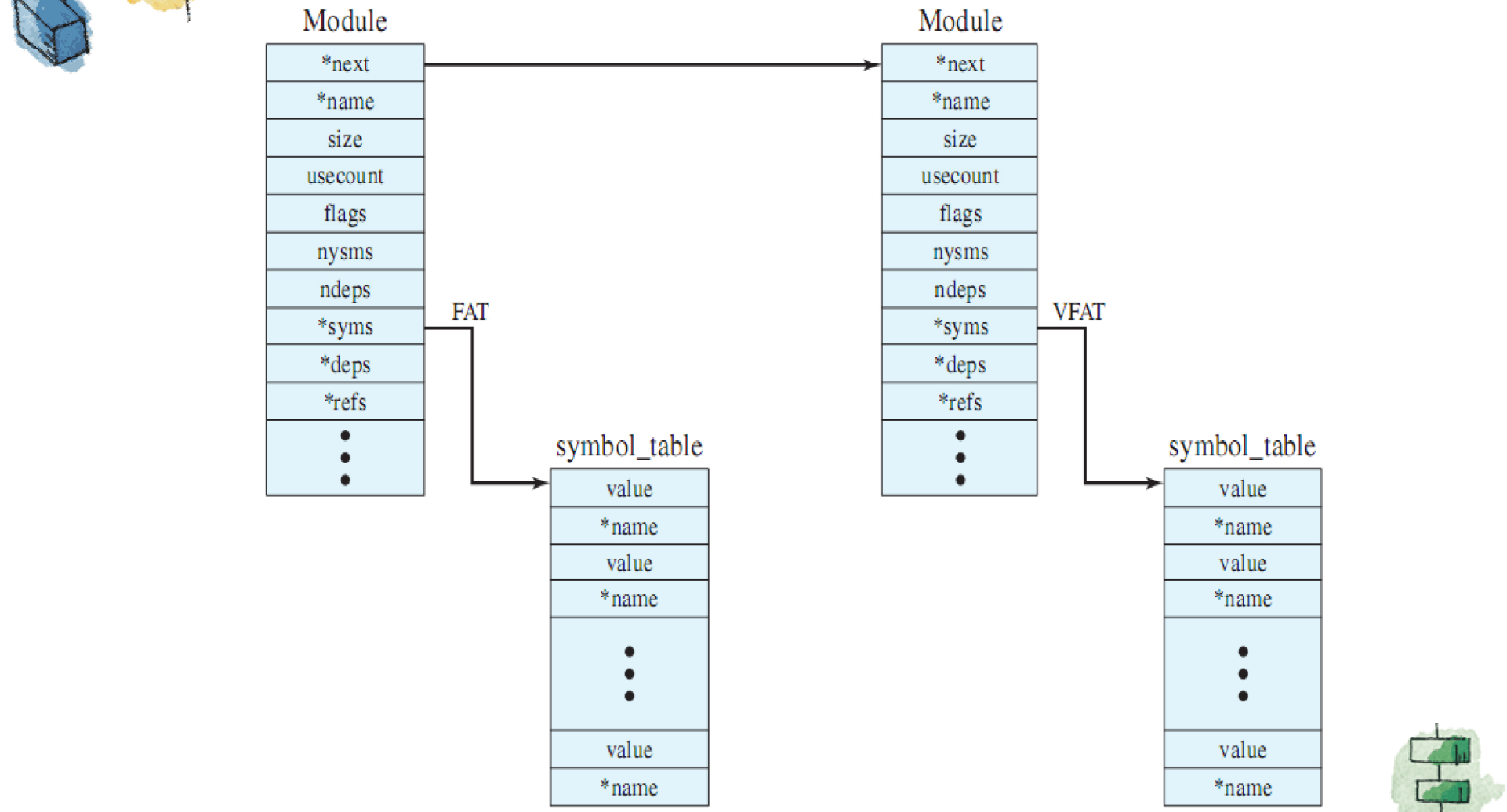


Figure 2.17 **Example List of Linux Kernel Modules**

Linux Kernel Components

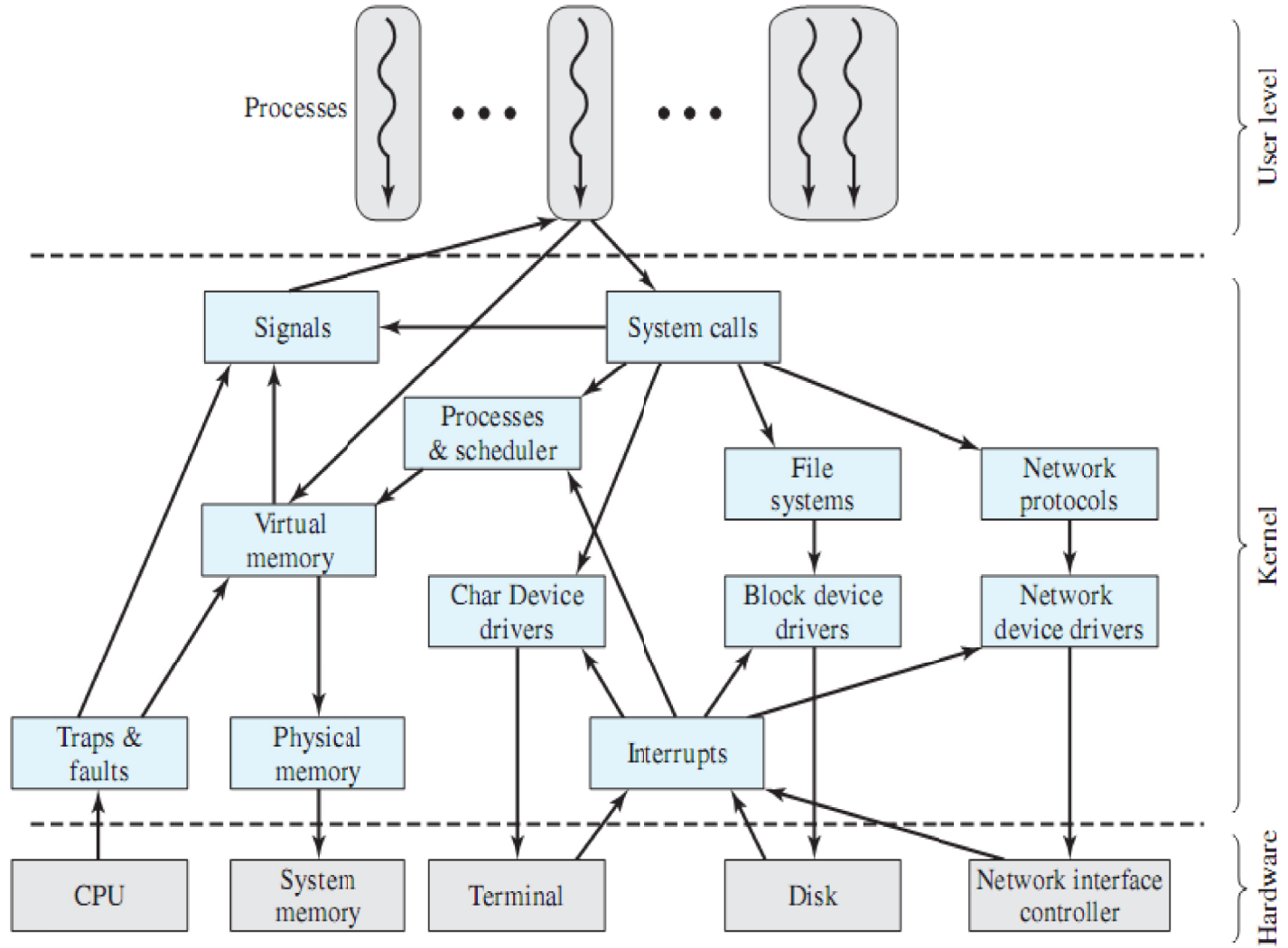
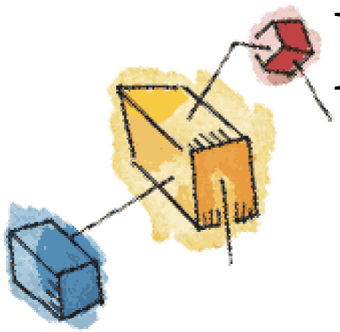
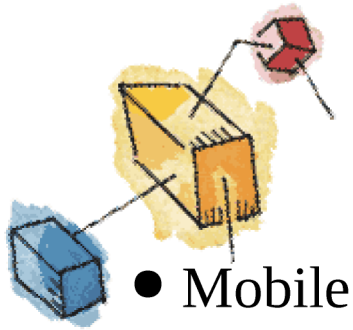


Figure 2.18 **Linux Kernel Components**



Linux VServer Virtual Machine Architecture

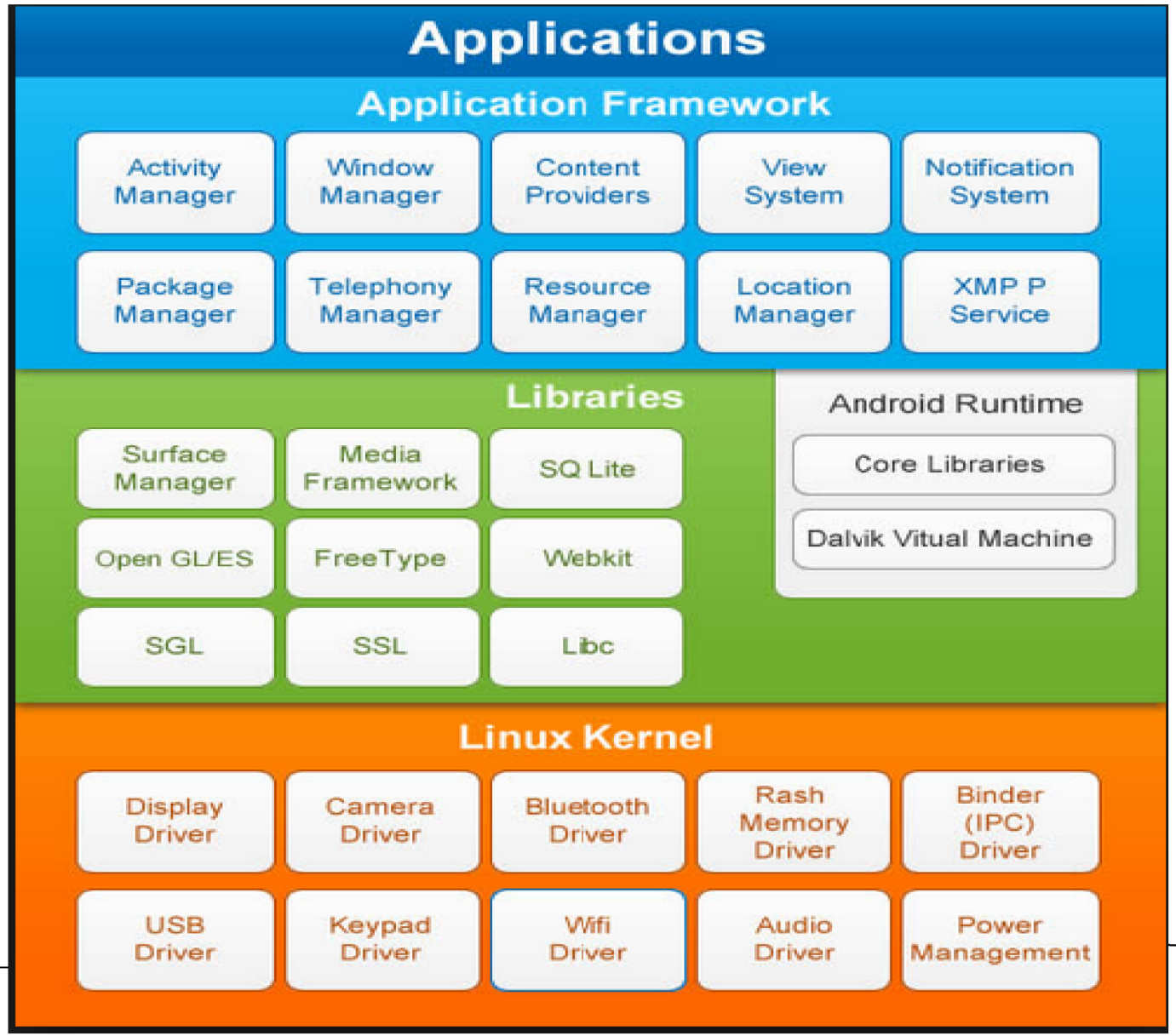
- Open-source, fast, lightweight approach to implementing virtual machines on a Linux server
- Only a single copy of the Linux kernel is involved
- Supports a number of separate virtual servers
- Each virtual server is isolated from the others
- Involves four elements:
  - chroot
  - chcontext
  - chbind
  - capabilities



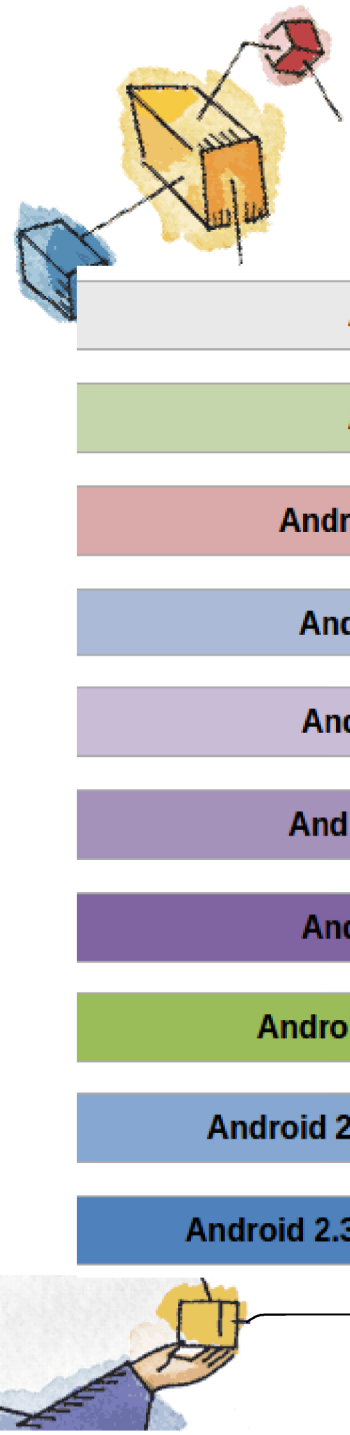
Android

- Mobile OS based on Linux
- Touch Screen mobile devices
- Developed in 2003 in Carlifornia
- Bought by Google in 2005 – continue to be developed by Google till now
- Open source
- Phenomena of “Smart Phone Wars”
- 2013 – Have over 1 million applications in Google Store, 50 billions apps were downloaded

Android Architecture



Android Versions



- Android 1.0 (API level 1)
- Android 1.1 (API level 2)
- Android 1.5 Cupcake (API level 3)
- Android 1.6 Donut (API level 4)
- Android 2.0 Eclair (API level 5)
- Android 2.0.1 Eclair (API level 6)
- Android 2.1 Eclair (API level 7)
- Android 2.2–2.2.3 Froyo (API level 8)
- Android 2.3–2.3.2 Gingerbread (API level 9)
- Android 2.3.3–2.3.7 Gingerbread (API level 10)
- Android 3.0 Honeycomb (API level 11)
- Android 3.1 Honeycomb (API level 12)
- Android 3.2 Honeycomb (API level 13)
- Android 4.0–4.0.2 Ice Cream Sandwich (API level 14)
- Android 4.0.3–4.0.4 Ice Cream Sandwich (API level 15)
- Android 4.1 Jelly Bean (API level 16)
- Android 4.2 Jelly Bean (API level 17)
- Android 4.3 Jelly Bean (API level 18)
- Android 4.4 KitKat (API level 19)
- Android 4.4 KitKat with wearable extensions (API level 20)
- Android 5.0–5.0.2 Lollipop (API level 21)