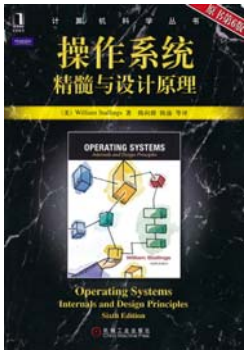
西南交通大学
Southwest Jiaotong University

操作系统

Operating Systems
Internals and Design Principles

周杲 Zhou Gao
email: gzhou@swjtu.edu.ch

Textbook

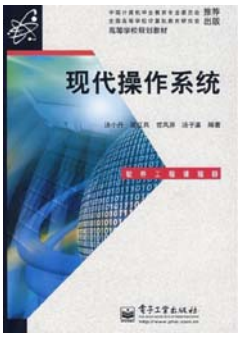


- Title: Operating Systems—Internals and Design Principles (Seventh Edition)
- Author: William Stallings
- Publisher: Prentice-Hall

Reference

- Operating Systems-Design and implementation (Second Edition)
- Andrew S. Tanenbaum
- Prentice Hall & Tsinghua Univ. Press

Reference.



- 汤小丹，梁红兵，哲凤屏，汤子瀛，现代操作系统，电子工业出版社，2008年4月
- 庞丽萍，郑然，操作系统原理与Linux系统实验，机械工业出版社，2011年5月

Evaluation

- Final exam 60%
- Mid-term exam 10%
- Assignment 15%
- In-class participation 15%

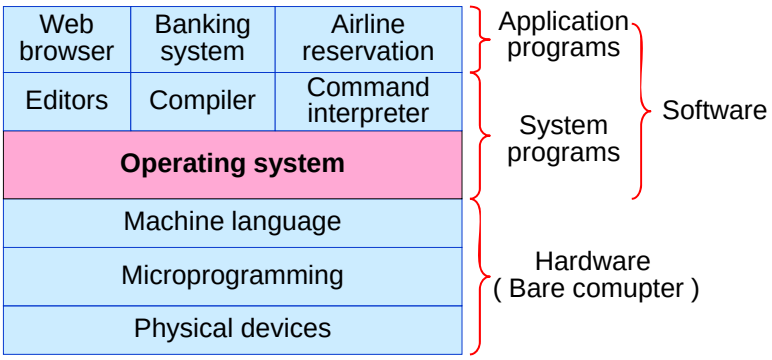
目录 Contents

Chapter 1: Overviews

1. Operating System Objectives and Functions
2. The Evolution of Operating Systems
3. Typical Modern Operating Systems

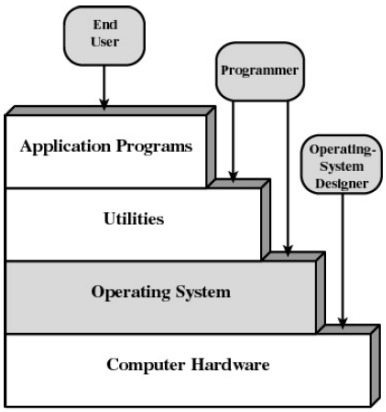
1.1 Operating System Objectives and Functions

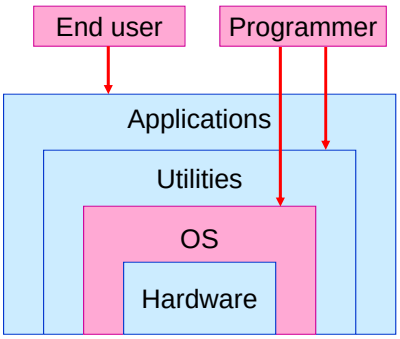
Composition of a Computer



A Computer System

The User's Point of View

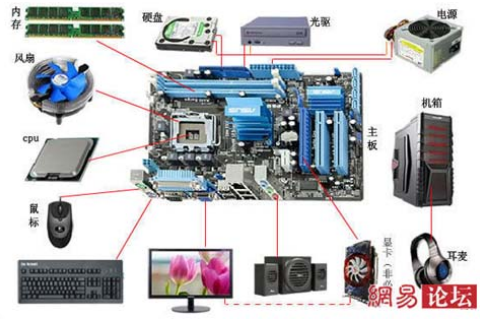




The Operating System as a Virtual Machine

1.1 Operating System Objectives and Functions

1 Convenience



A Computer Physical devices

Convenience.

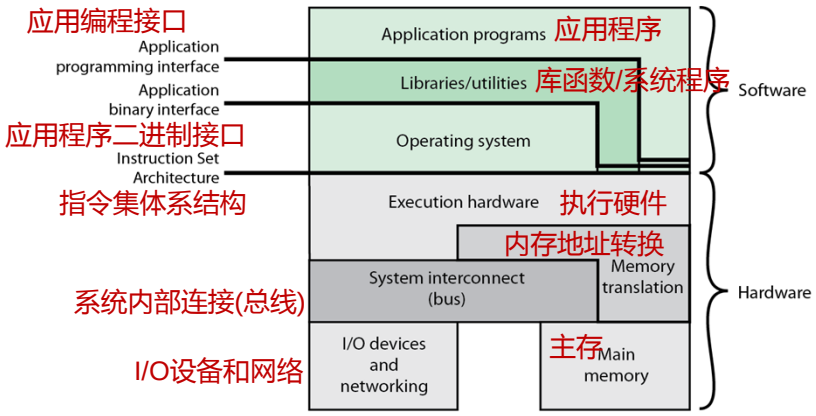


Figure 2.1 Computer Hardware and Software Infrastructure

The Operating System as a User/Computer Interface

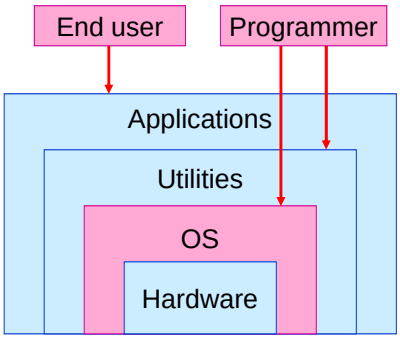
Convenience..

The Operating System functions

- Program development
- Program execution
- Access I/O devices
- Controlled access to files
- System access
- Error detection and response
- Accounting

1.1 Operating System Objectives and Functions

2 Efficiency



The Operating System as a Resource Manager

Efficiency.

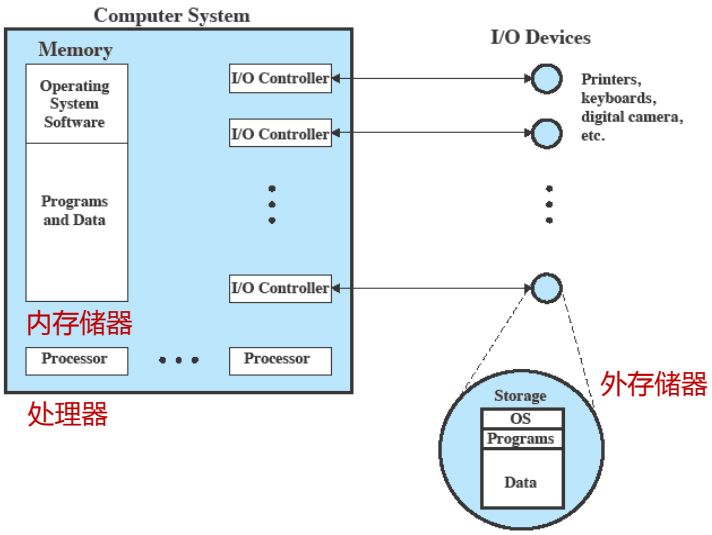


Figure 2.2 The Operating System as Resource Manager

Efficiency..

Categories of I/O Devices

- Human readable: suitable for communicating with the computer user. e.g., printers, terminals, video display, keyboard, mouse
- Machine readable: suitable for communicating with electronic equipment, e.g., disk drives, USB keys, sensors, controllers
- Communication: suitable for communicating with remote devices, e.g., modems, digital line drivers

Efficiency...

- 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

- Functions in the same way as ordinary computer software, that is, it is a program, or suite of programs, executed by the processor
- Frequently relinquishes control and must depend on the processor to allow it to regain control

1.1 Operating System Objectives and Functions

The Role of an OS

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

1 Convenience

- Makes the computer more convenient to use

2 Efficiency

- Allows computer system resources to be used in an efficient manner

1.1 Operating System Objectives and Functions

3 Ability to evolve

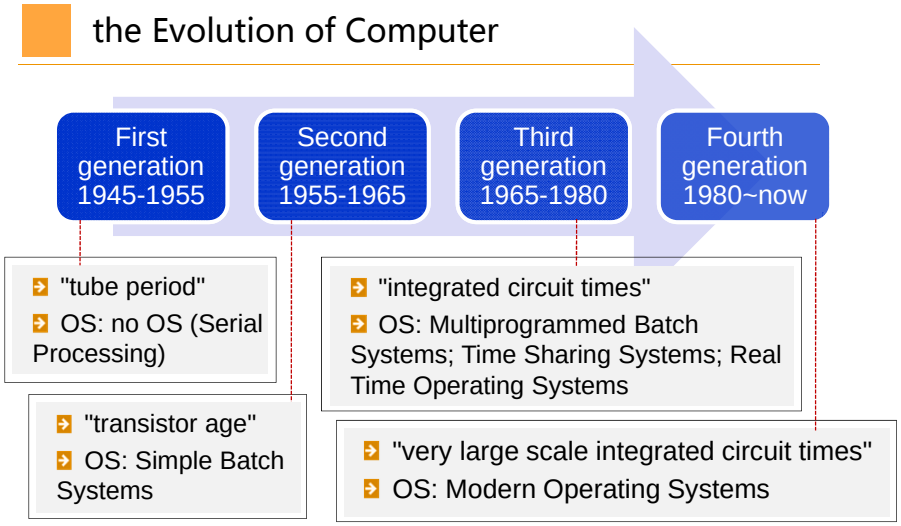
- Permit effective development, testing, and introduction of new system functions without interfering with service

Evolution of an OS

- Hardware upgrades plus new types of hardware
- New services
- Fixes

1.2 the Evolution of Operating Systems

the Evolution of Computer



1.2 the Evolution of Operating Systems

1 Serial Processing

- No operating system, programmers interacted directly with the computer hardware.
- Machines run from a console with display lights, toggle switches, input device, and printer.
- Users have access to the computer in “series”.

(1) Basic Elements of Computer

- Processor(处理器，CPU-中央处理单元)
- Main Memory(内存/主存)
  - Volatile(易失的)，referred to as real memory or primary memory
- I/O modules(I/O模块)
  - secondary memory devices(辅助存储设备)
  - communications equipment
  - Terminals(终端)
- System bus(系统总线)
  - communication among processors, memory, and I/O modules

Basic Elements of Computer.

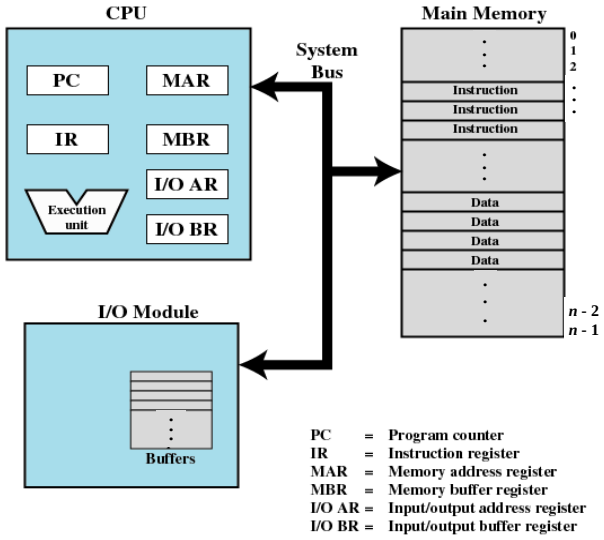


Figure 1.1 Computer Components: Top-Level View

(2) Instruction Execution(指令的执行)

- Two Steps
  - Processor reads instructions from memory
  - Processor executes each instruction

Instruction cycle(指令周期)

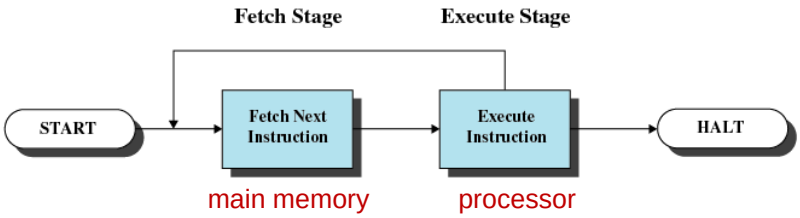
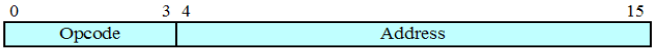


Figure 1.2 Basic Instruction Cycle

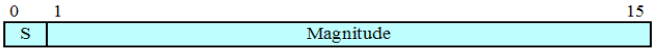
(3) Instruction Fetch and Execution

- Program counter (**PC**, 程序计数器) holds address of the instruction to be fetched next
  - The processor fetches the instruction (取指令) from memory
  - Program counter is incremented after each fetch
- Fetched instruction is placed in the instruction register (**IR**, 指令寄存器)
- Categories of actions:
  - Processor-memory (处理器和内存之间): Transfer data between processor and memory
  - Processor-I/O (处理器和I/O设备之间): Data transferred to or from a peripheral device(外部设备)
  - Data processing: Arithmetic(算术) or logic operation on data
  - Control: Alter sequence (顺序) of execution

A Example of 16-bit words



(a) Instruction format



(b) Integer format

Program Counter (PC) = Address of instruction
Instruction Register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from Memory
0010 = Store AC to Memory
0101 = Add to AC from Memory

(d) Partial list of opcodes

Figure 1.3 Characteristics of a Hypothetical Machine

Example of Program Execution

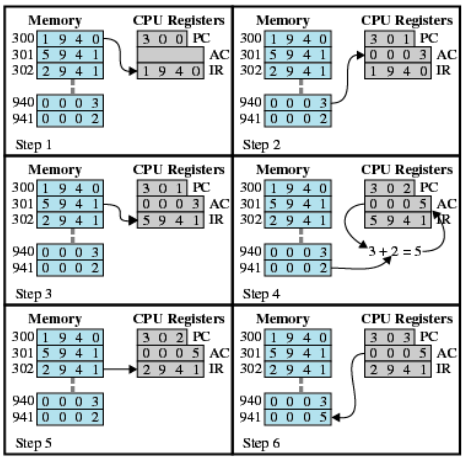


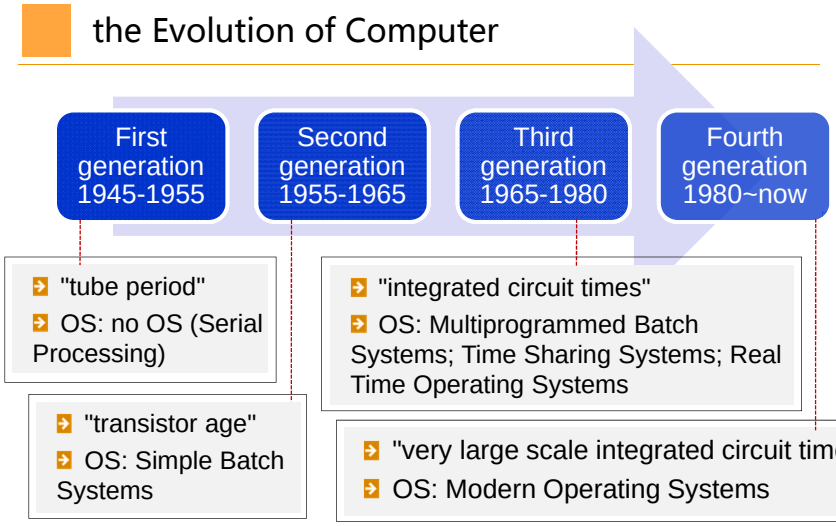
Figure 1.4 Example of Program Execution
(contents of memory and registers in hexadecimal)

Serial Processing: problems

- Scheduling: most installations used a hardcopy sign-up sheet to reserve computer time
  - time allocations could run short or long, resulting in wasted computer time.
- Setup time: a considerable amount of time was spent just on setting up the program to run
  - Setup included loading the compiler, source program, saving compiled program, and loading and linking

1.2 the Evolution of Operating Systems

the Evolution of Computer



1.2 the Evolution of Operating Systems

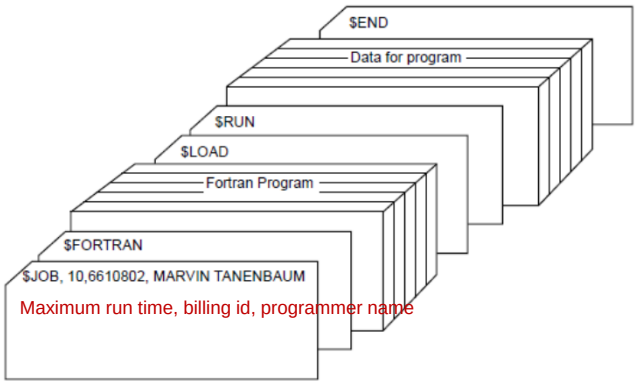
2 Simple Batch Systems

- The central idea behind the simple batch processing scheme was the use of a piece of software known as the monitor, by which a batch of jobs is automatically processed.
- Jobs(作业): collections of user programs and their required data and commands.
- Requirements:
  - Software Features: Monitor and Job Control Language (JCL)
  - Hardware Features

(1) Software Features

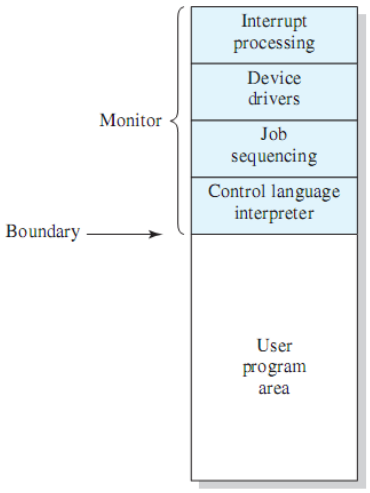
- Monitor
  - user no longer has direct access to processor, the monitor controls the sequence of events.
  - Batch jobs together: job is submitted to computer operator who batches them together and places them on an input device
  - program branches back to the monitor when finished
- Job Control Language (JCL)
  - Special type of programming language used to provide instructions to the monitor
  - Job control instructions are denoted by the beginning "$"

A Typical FMS Job



- Typical operating systems are FMS (FORTRAN Monitor System) and IBSYS (IBM Operating System for Machine 7094).

Monitor Point of View



- Monitor is a 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**

Processor Point of View

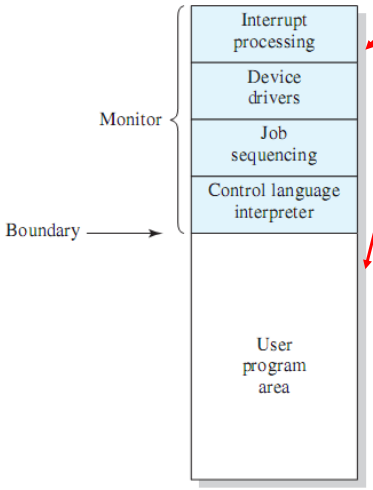


Figure 2.3 Memory Layout for a Resident Monitor

- Processor executes instruction from the memory containing the monitor
- Executes the instructions in the user program until it encounters an ending or error condition
- "***control is passed to a job***" means processor is fetching and executing instructions in a user program
- "***control is returned to the monitor***" means that the processor is fetching and executing instructions from the monitor program

Modes of Operation

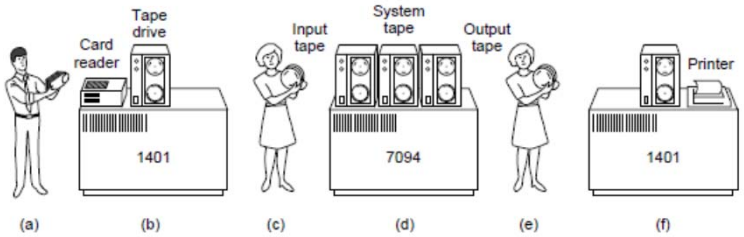
- User Mode
  - user program executes in user mode
  - certain areas of memory are protected from user access
  - certain instructions may not be executed
- Kernel(核心/内核) Mode
  - monitor executes in kernel mode
  - privileged instructions may be executed
  - protected areas of memory may be accessed

(2) Hardware Features

- Memory protection for monitor
  - while the user program is executing, it must not alter the memory area containing the monitor
- Timer
  - prevents a job from monopolizing the system
- Privileged instructions
  - can only be executed by the monitor
- Off-line Input/Output
- Interrupts
  - gives OS more flexibility in controlling user programs

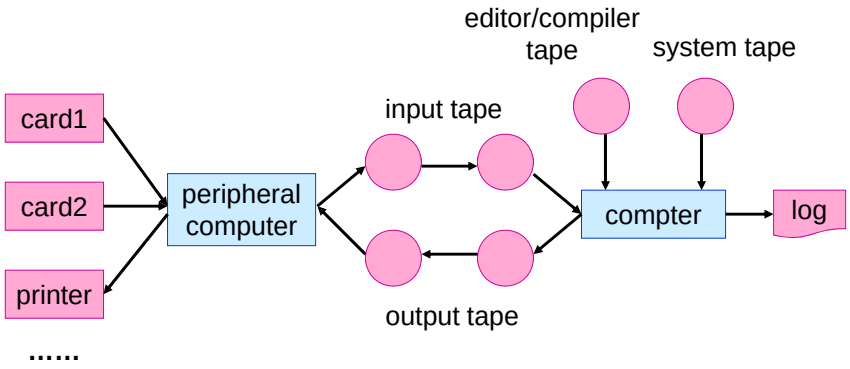
(3) Off-line Input/Output (脱机输入/输出)

- Off-line Input/Output: the input/output of programs and data is done under the control of peripheral computers, or away from the mainframe.



(a) The programmer takes several cards(several jobs) to machine 1401
(b) 1401 reads the batch job to tape
(c) Operator sends input tape to 7094
(d) 7094 is calculated
(e) The operator sends the output tape to 1401
(f) 1401 prints the results

Off-line Input/Output.



Schematic of offline input and output

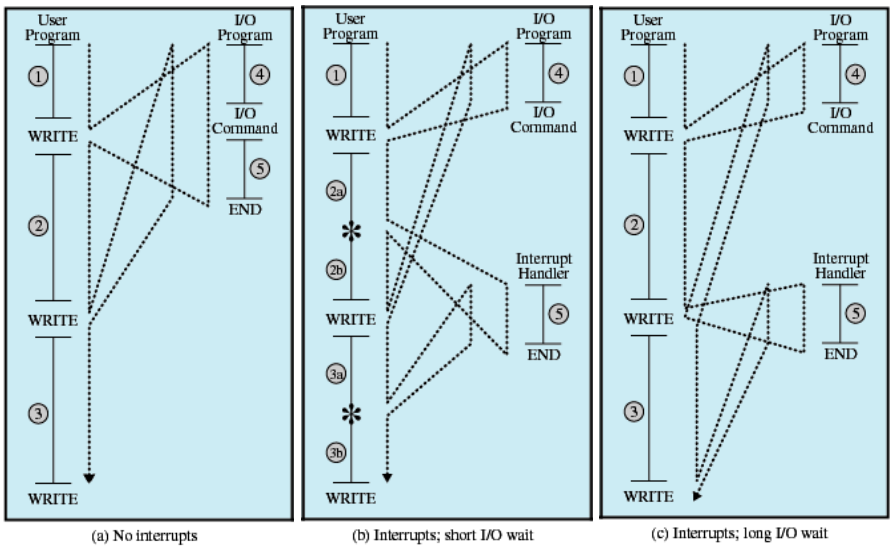
(4) Interrupts (中断)

- Interrupts are provided primarily as a way to improve processor utilization
  - Most I/O devices are slower than the processor
  - Processor must pause to wait for device
  - Interrupt the normal sequencing of the processor

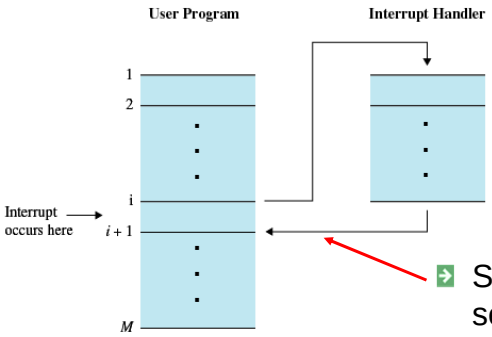
Table 1.1 Classes of Interrupts

Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
I/O	Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.
Hardware failure	Generated by a failure, such as power failure or memory parity error.

Program Flow of Control



Program Flow of Control.



- Suspends (挂起) the normal sequence of execution

Figure 1.6 Transfer of Control via Interrupts

- Interrupt Handler(中断服务程序): also named Interrupt Service Routine, a program to service a particular I/O device, generally part of the operating system

Interrupt Cycle

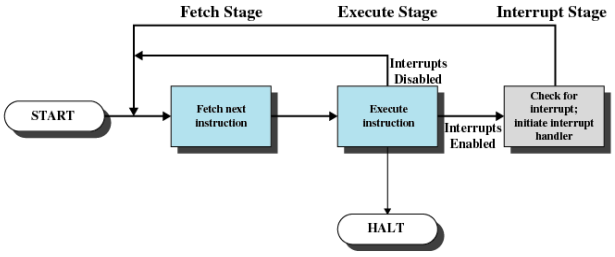


Figure 1.7 Instruction Cycle with Interrupts

- Processor checks for interrupts after each instruction
- If no interrupts are pending, fetch the next instruction for the current program
- If an interrupt is pending, suspend execution of the current program, and execute the interrupt-handler routine

Timing Diagram

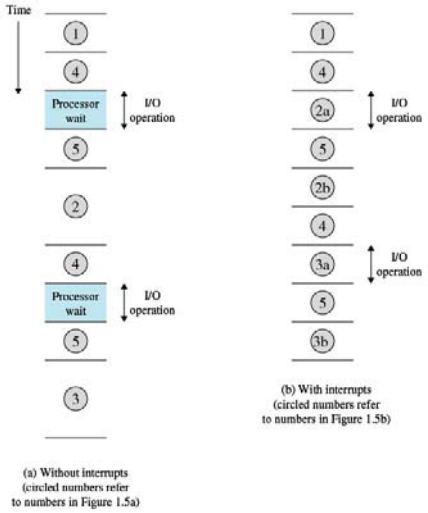


Figure 1.8 Program Timing: Short I/O Wait

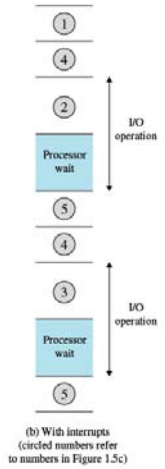


Figure 1.9 Program Timing: Long I/O Wait

Simple Interrupt Processing

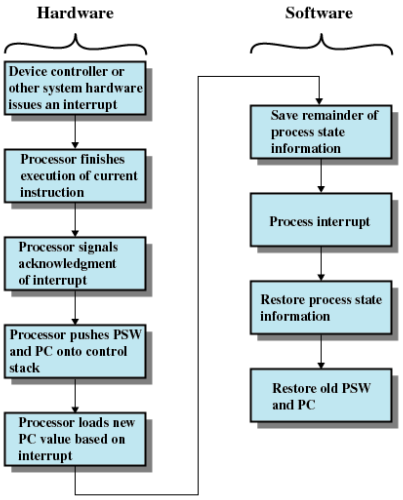
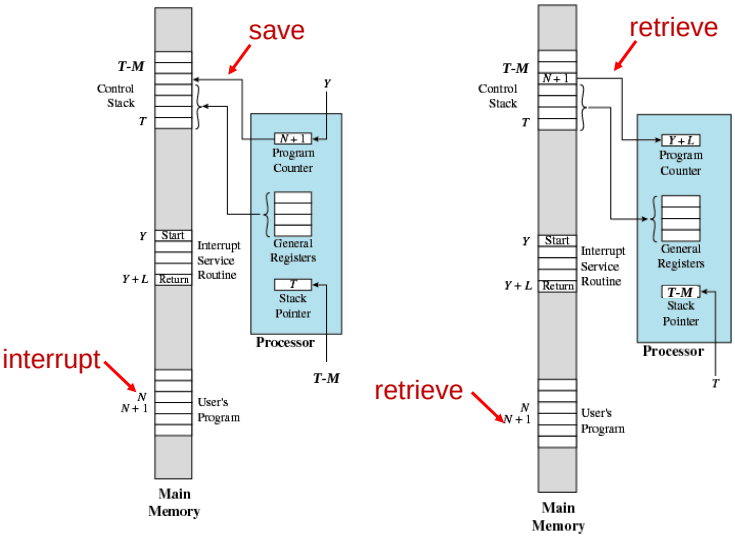


Figure 1.10 Simple Interrupt Processing

Changes in Memory and Registers

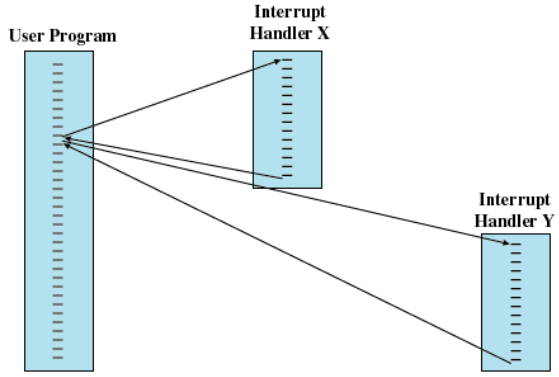


(a) Interrupt occurs after instruction at location *N*

(b) Return from interrupt

(5) Multiple Interrupts

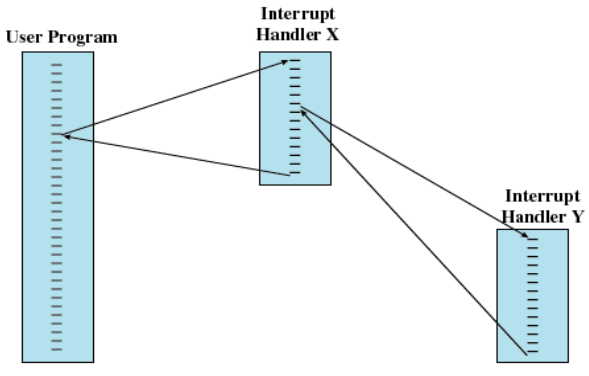
- Disable interrupts while an interrupt is being processed



(a) Sequential interrupt processing

Multiple Interrupts.

- Define priorities (优先级) for interrupts



(b) Nested interrupt processing

Multiple Interrupts..

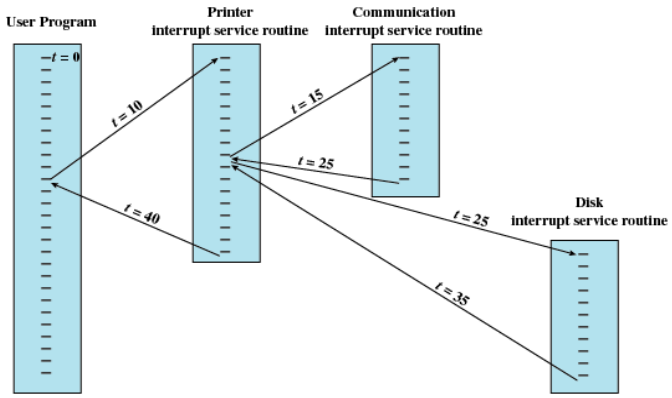


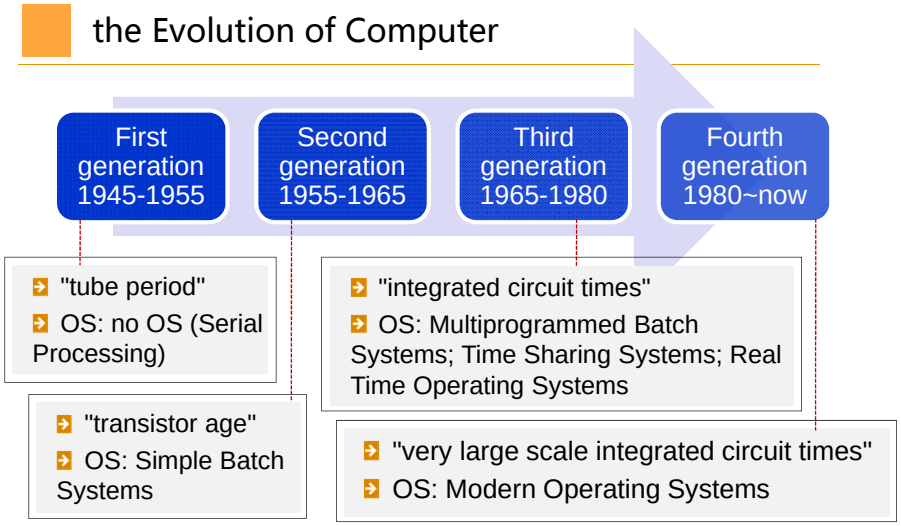
Figure 1.13 Example Time Sequence of Multiple Interrupts

Simple Batch System Overhead

- Processor time alternates between execution of user programs and execution of the monitor
- Sacrifices:
  - some main memory is now given over to the monitor
  - some processor time is consumed by the monitor
- Despite overhead(开销), the simple batch system improves utilization of the computer

1.2 the Evolution of Operating Systems

the Evolution of Computer



1.2 the Evolution of Operating Systems

3 Multiprogrammed Batch Systems

- Even with automatic job sequencing, processor is often idle
  - 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

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

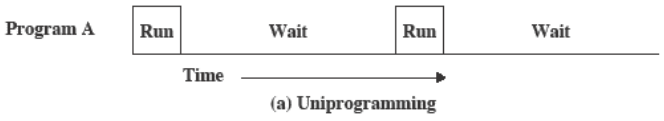
Figure 2.4 System Utilization Example

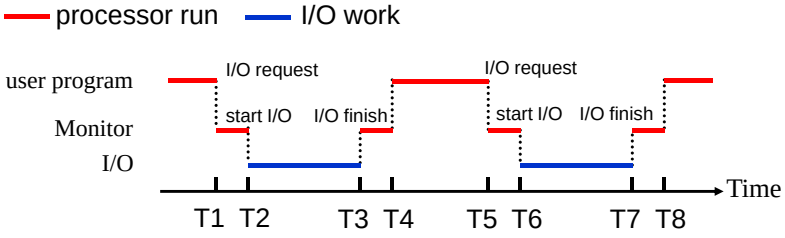
(1) Spooling (联机的即时外部设备操作)

- **Spooling(Simultaneous Peripheral Operation on Line)**: the technique uses program to simulate the peripheral machine in the offline input/output, so that the data obtained by the I/O devices can be directly into the host. It enables multiple users to share an I/O device, improves the utilization rate of the devices and speeds up the execution process of the program.

(2) Multiprogramming (多道/多任务)

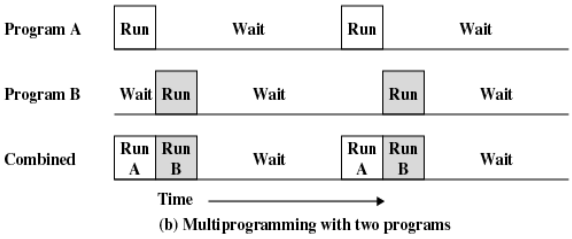
- **Multiprogramming**: A technique for maximizing CPU utilization by placing several jobs in memory at the same time and allowing them to be executed alternately, sharing various hardware and software resources in the system.



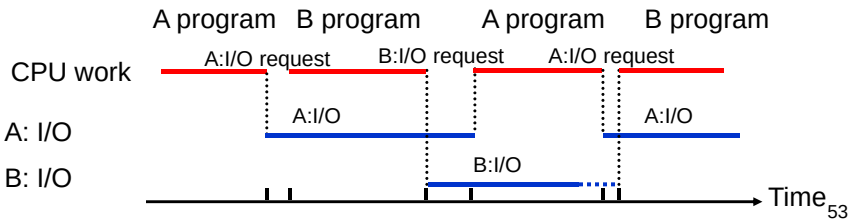


Multiprogramming.

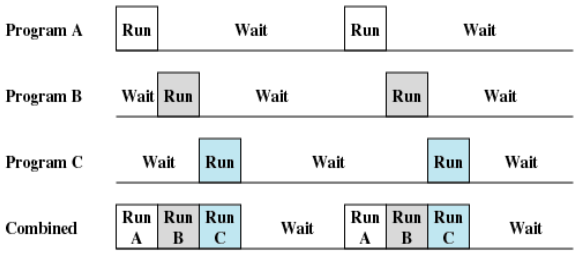
- When one job needs to wait for I/O, the processor can switch to the other job



(b) Multiprogramming with two programs



Multiprogramming..



(c) Multiprogramming with three programs

- On the macro, several programs, which are executed successively, all are running and have not finished yet.
- On the micro, each program is executed alternately, that is, the CPU switches back and forth among the processes.

(3) Multiprogrammed Batch Systems

- Processor has more than one program to execute
- memory is expanded to hold three, four, or more programs and switch among all of them.
- There must be enough memory to hold the OS (resident monitor) and other user programs.
- The sequences the programs are executed depend on their relative priority and whether they are waiting for I/O
- After an interrupt handler completes, control may not return to the program that was executing at the time of the interrupt

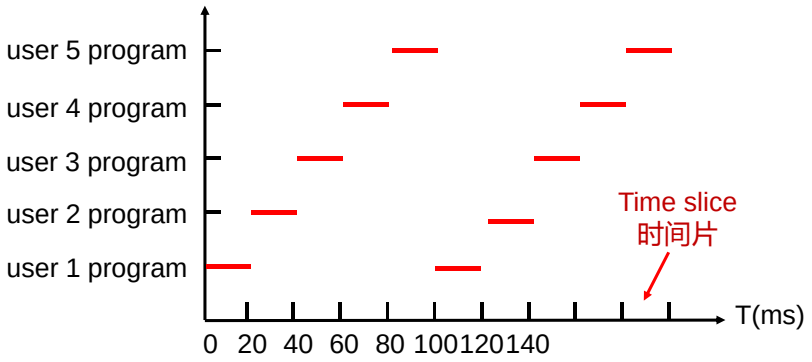
1.2 the Evolution of Operating Systems

4 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
- Generally be used to handle multiple interactive jobs.
- The OS interleaving the execution of each user program in a short burst or quantum of computation.
- Processor time is shared among multiple users.

Time sharing technology (分时技术)



- Typical time-sharing system: CTSS and MULTICS

CTSS Operation

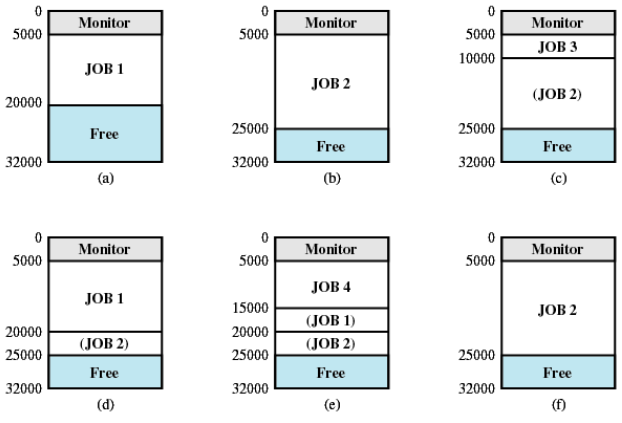
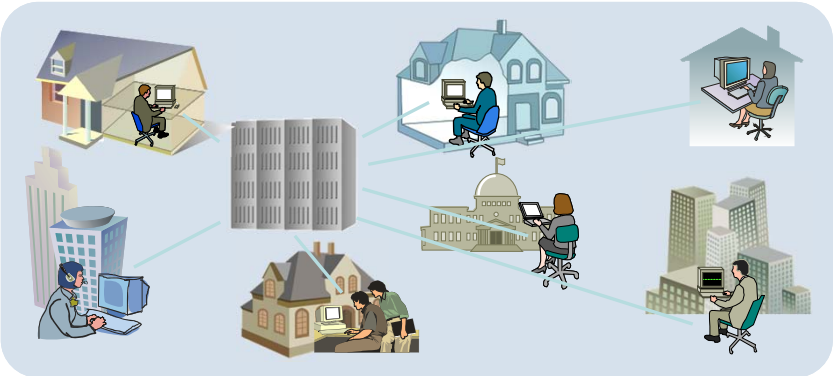


Figure 2.7 CTSS Operation

- CTSS developed at MIT(麻省理工学院)

MULTICS Operation

- MULTICS (MULTiplexed Information and Computing Service) by MIT, Bell Labs, General Electric(通用电气)



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

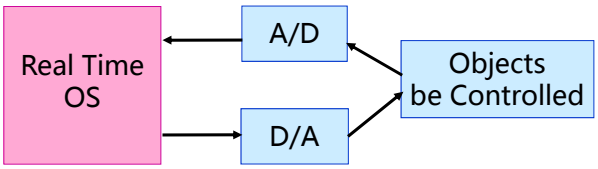
- Another major development during the 3rd generation was the phenomenal growth of minicomputers, staring with the Digital Equipment Company(DEC) PDP-1($12000, 5% of the price of 7094). Ken Thompson subsequently set out to write a stripped-down one-user version of MULTICS on a PDP-7(1969). This work later developed into the **UNIX**.

1.2 the Evolution of Operating Systems

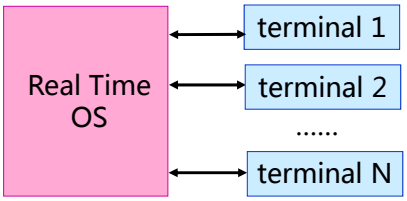
5 Real Time Operating System

- Real-time operating system refers to the system can timely respond to the request of external events, complete the processing of the event within the specified time (deadline), and control the coordinated operation of all real-time tasks.
- Application
  - Real-time control(Military control, industrial control, medical control)
  - Real-time information processing(Medical control reservation system, online information retrieval)

Real Time Operating System.



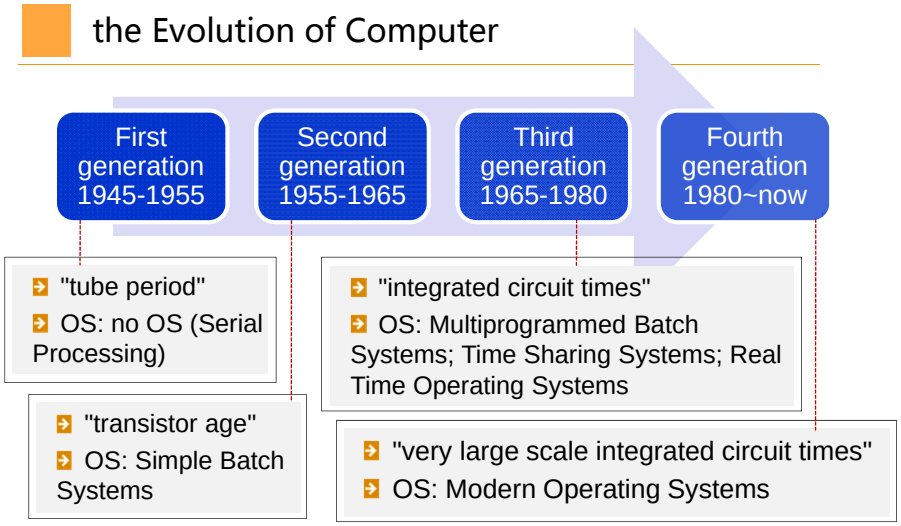
Real-time control



Real-time information processing

1.2 the Evolution of Operating Systems

the Evolution of Computer



1.2 the Evolution of Operating Systems

6 Modern Operating Systems

8bit

- CP/M(Control Program for Microcomputer,1974), Apple II(1977), UNIX(1969)
- 8080 or Z80 microcomputer / minicomputer PDP-7

16bit

- DOS(Disk Operating System, 1980s), MS-DOS, Windows 1.0(1985), Apple DOS, UNIX V1(1971),V6(1975), V7(1979)
- 8086/8088 microcomputer / PDP-11/24...

GUI

- GUI(Graphical User Interface, 1973): windows, icons, menus, mouse
- Apple Macintosh System 1(1984), Windows3.0 (on top of MS-DOS,1990),
- Motorola's 68000 / Personal computer

Modern Operating Systems.

32bit

- Apple Macintosh, Mac OS X(2001, on top of UNIX)
- IBM Power PC, Intel Processor(2005)

- Windows 95(on top of DOS), Windows 98, Windows NT(1993)
- Personal computer, Intel Processor

- UNIX 32V, UNIX(X window,1984)
- Workstation/Sever(VAX, RISC chips, Pentium-based Computer)

- Linux(1991), Linux1.0(1994)

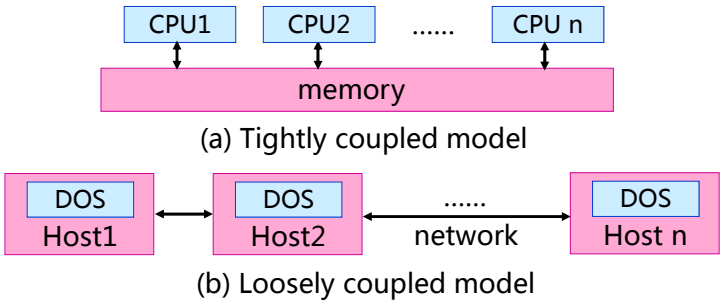
other

- Network OS and Distributed OS(mid-1980s)

(1) Network OS and Distributed OS

- Network operating systems are not fundamentally different from single-processor operating systems. They obviously need a network interface controller and some low-level software to drive it, as well as programs to achieve remote login and remote file access, but these additions do not change the essential structure of the operating system.
- Distributed operating system, in constrast, is one that appears to its users as a traditional uniprocessor system, even though it is actually composed of multiple processors. The users should not be aware of where their programs are being run or where their files are located; that should all be handled automatically and efficiently by the operating system.

Network OS and Distributed OS.



- True distributed OS require more than just adding a little code to an uniprocessor OS, because distributed and centralized systems differ in critical ways. Distributed systems, for example, often allow applications to run on several processors at the same time, thus requiring more complex processor scheduling algorithms in order to optimize the amount of parallelism.

(2) Major Achievements

- Operating Systems are among the most complex pieces of software ever developed.
- Major advances in development include:
  - Processes
  - Memory management
  - Information protection and security
  - Scheduling and resource management
  - System structure
- Three major lines of computer system development
  - multiprogramming batch
  - time sharing
  - real-time transaction system

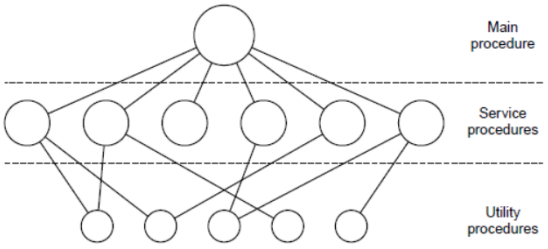
Processes

- A process is basically a program in execution. It is a somewhat more general term than job.
- A process contains three components:
  - an executable program
  - the associated data needed by the program (variables, work space, buffers, etc.)
  - the execution context (or “process state”) of the program job.

(3) System structures

Monolithic Systems (单片机系统)

- The operating system is written as a collection of procedures(过程/函数), each of which can call any of the other ones whenever it needs to.
  - A main program that invokes the requested service procedure
  - A set of service procedures that carry out the system calls
  - A set of utility procedures that help the service procedures



Layered Systems

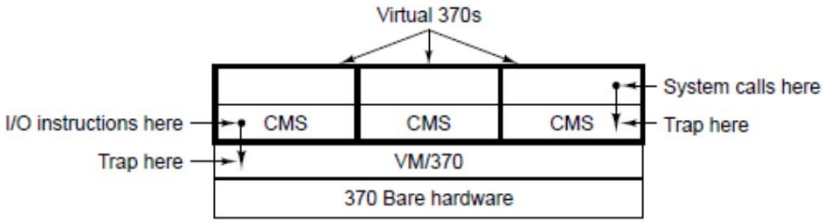
- A generalization of the approach of Monolithic Systems is to organize the OS as a hierachy of layers, each one constructed upon the one below it.
- The THE system by E.W.Dijkstra (1968) is the first system in this way.The THE system was a simple batch system for a Dutch computer, the Electrologica X8.

Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

Virtual Machines

- CP/CMS and later renamed VM/370(1979) was based on a very astute observation:
  - multiprograming;
  - an extended machine with a more convenient interface than the bare hardware.
- The heart of the system, known as the virtual machine monitor(VM), runs on the bare hardware and does the multiprogramming, providing not one, but several virtual machines to the next layer up. These virtual machines are exact copies of the bare hardware. Because each virtual machine is identical to the true hardware, each one can run any OS that will run directly on the bare hardware.

Virtual Machines.



- For timesharing users, interactive system called CMS(Conversation Monitor System) should be run, to execute a system call in virtual 370s, and issue the normal hardware I/O instructions to VM/370,which then performs them as part of its simulation of the real hardware.

Virtual Machines..

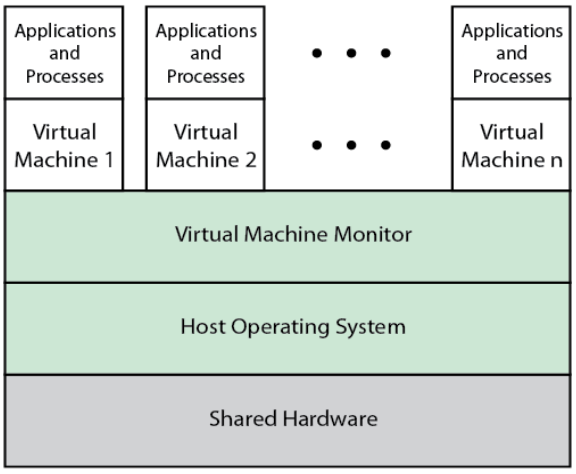
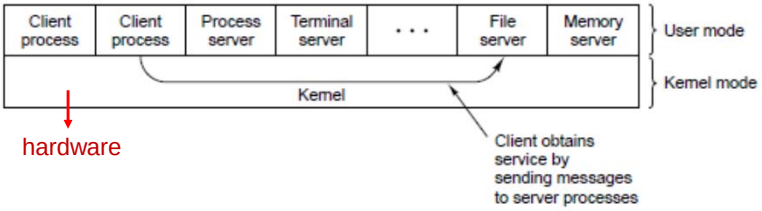


Figure 2.13 Virtual Memory Concept

Client-Server Model

- A trend in modern operating systems is to take the virtual machine idea of moving code up into higher layers even further and remove as much as possible from the OS, leaving a minimal kernel. The usual approach is to implement most of the OS functions in user processes. To request a service, such as reading a block of a file, a user process, now known as the client process, sends the request to a sever process, which then does the work and sends back the answer. All the kernel does is handle the communication between clients and servers.



(4) Features of Modern Operating Systems

- Microkernel architecture(微内核): Assigns only a few essential functions to the kernel
  - Address spaces
  - Interprocess communication (IPC)
  - Basic scheduling
- Multithreading(多线程): Process is divided into threads that can run concurrently
  - Thread: Dispatchable unit of work, can execute sequentially and is interruptable
  - Process: is a collection of one or more threads

Features of Modern Operating Systems.

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

Features of Modern Operating Systems..

- Multiprogramming(多道) and multiprocessing (多处理)

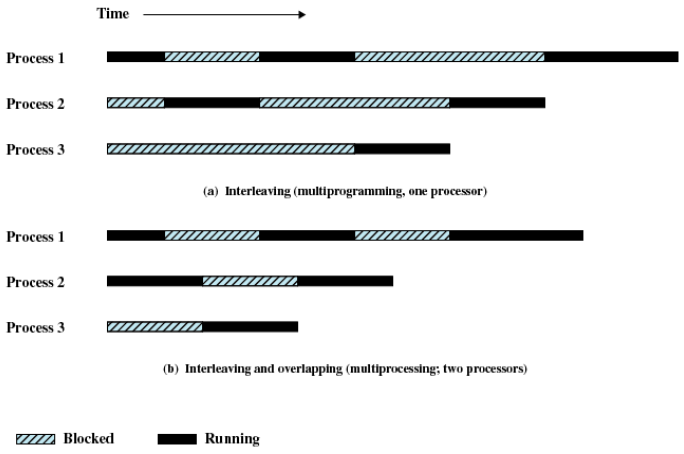


Figure 2.12 Multiprogramming and Multiprocessing

1.3 Typical Modern Operating Systems

1 Traditional UNIX

- Hardware is surrounded by the operating system software
- Operating system is called the system kernel
- Comes with a number of user services and interfaces
  - Shell
  - Components of the C compiler

(1) Traditional UNIX Architecture

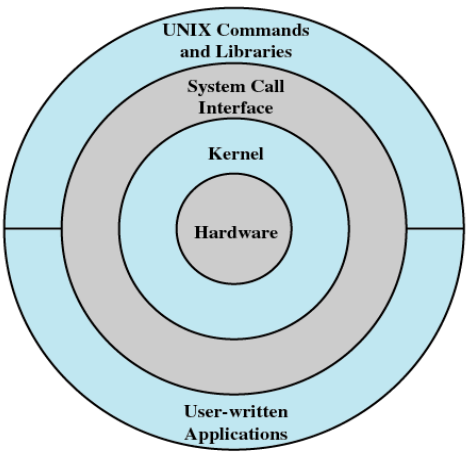


Figure 2.14 General UNIX Architecture

Traditional UNIX Kernel

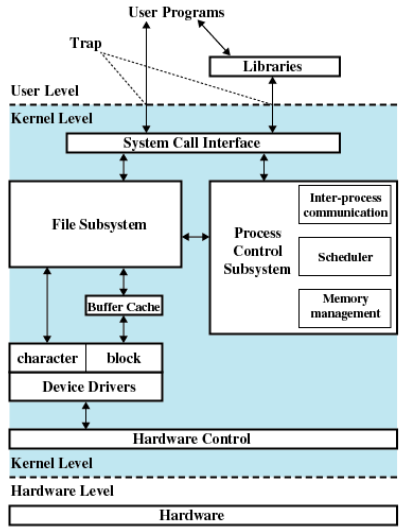
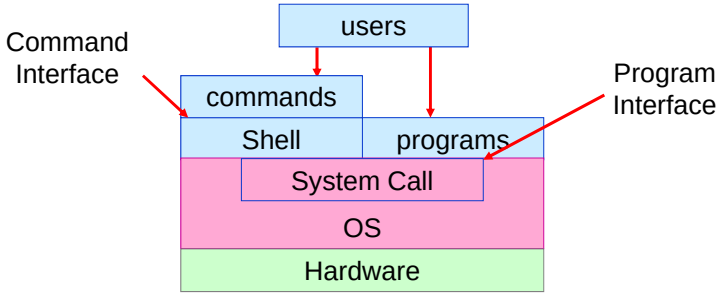


Figure 2.15 Traditional UNIX Kernel [BACH86]

(2) User Interface

- **Command Interface**: users use operational commands to organize and control the execution of jobs or to manage a computer system.
- **Program Interface**: also known as **system call**. Programmers use system calls in their own applications to request services from the operating system, such as requesting and releasing resources.



Command Interface

- process
  - input commands on the terminal by keyboard.
  - The terminal handler receives the command and displays it on the screen
  - Command interpreters analyze and execute commands
- Implementation: Use the Command Interpreter(命令解释器)
  - MS-DOS：COMMAND.COM
  - UNIX & MINIX：Shell
- **Shell**, not a part of OS, is the interface between the end user and the operating system. It is responsible for interpreting commands from the terminal and input/output data to/from terminal.

Program Interface

- A **system call** is a set of instructions provided by the operating system to the user, also called the "extended instruction set". It is the interface between the User program and the operating system (Kernel).
- System calls can be used not only by all applications, but also by other parts of the OS itself, especially command handlers.

```
int do_fork(message *m_ptr); // create a process
```

- User interface is an important indicator to measure the quality of an operating system.
- Graphical User Interface(GUI)

(3) Modern UNIX Kernel

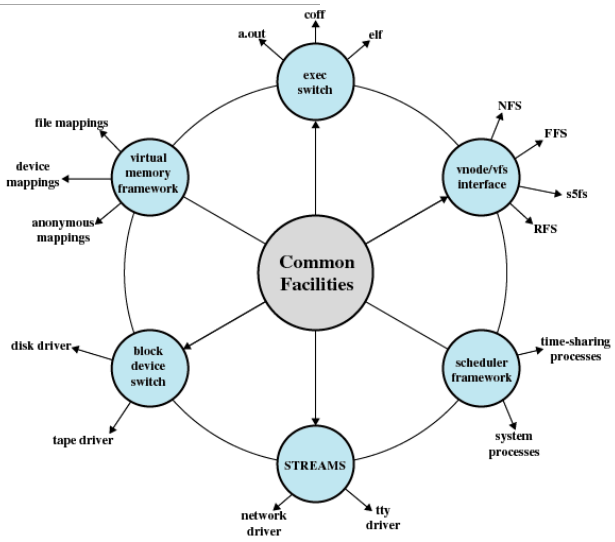


Figure 2.16 Modern UNIX Kernel [VAHA96]

1.3 Typical Modern Operating Systems

2 Microsoft Windows

- Windows is a sophisticated multitasking OS, designed to manage the complexity of modern computing environment, provide a rich platform for application developers, and support a rich set of experiences for users.
  - Modular structure for flexibility
  - Executes on a variety of hardware platforms
  - Supports application written for other operating system
- Windows program is event-driven, that is, the main program waits for the occurrence of events, such as the click of the mouse, and then according to the event content, call the corresponding program for processing.

(1) Windows Architecture

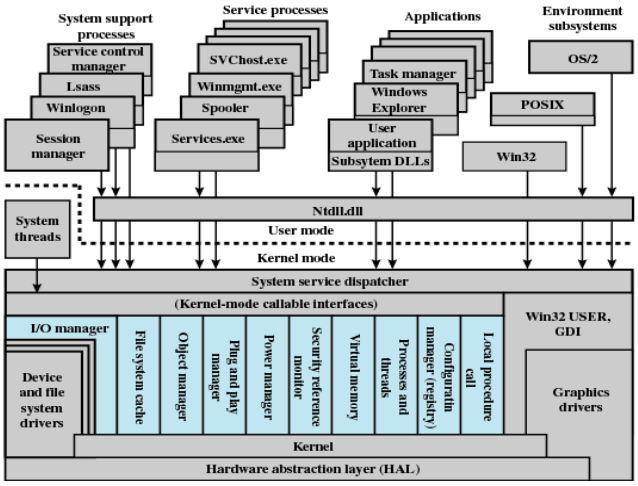


Figure 2.15 Windows and Windows Vista Architecture[RUSS11]

(2) User Interface

- Command Interface
  - Command interpreter
  - GUI: The OS generates a process for the user to run explorer.exe, which runs a command interpreter with a window interface, that is, a special window, the desktop. Similarly, command interpreter will generates new processes to pop new windows for every user's actions, which means that the user's action will generate an event, which will lead to the event-driven control program work.
- Program Interface: Application Program Interface(Win32 API), is the definition of a function used to provide operating system services.

Terminology

- operating system;
- serial processing;
- user mode; kernel mode; privileged instruction;
- off-line input/output; Spooling;
- interrupt; job; batch processing; batch system
- multitasking (multiprogramming); multiprogrammed batch system
- time sharing; time-sharing system
- real-time systems
- network OS and distributed OS
- monolithic model; layered model; microkernel model
- command interface; program interface