北京邮电大学软件学院

School Of software Engineering Of BUPT

Operating Systems

Lecture 2 Operating-System Structures

JinpengChen

Email: jpchen@bupt.edu.cn

Outline

- System components
- Operating System Services
- User Operating System Interface
- System Calls
- Operating System Design and Implementation
- Operating System Structure

System components

- **Process Management, 进程管理**
- **Main Memory Management, 内存管理**
- **I/O System Management, I/O管理**
- **File Management, 文件管理**
- **Secondary-Storage Management, 辅存/外存管理**
- **Command-Interpreter System, 命令解释系统**
- **Protection System, 保护**
- **Networking, 网络**

process(or processor) management

- 多道环境下，处理器的运行及分配都以进程（process）为单位，因此处理器管理可归结为进程管理（process management）。
  - (1) process control
    - ✓ create/destroy a process;
    - ✓ process state(进程状态) transferring
    - ✓ 一般由process control primitives(进程控制原语)完成
  - (2) process synchronization(同步)
    - ✓ 为使多个进程有条不紊地运行，应建立synchronization mechanism(同步机制)。
    - ✓ including

      process mutual execution/synchronization(进程互斥/同步); dead-lock avoidance, prevention, detection and resolution(死锁避免、预防、检测和消除)

process(or processor) management

- (3)Process communication
  - 源于进程合作，如：输入进程、计算进程、打印进程相互间有信息传递
  - 类型：
    - ✓ directly(直接通信)：
    - ✓ P_A 发msg，P_B 收msg

      $P_A \rightarrow$（msg） P_B
    - ✓ indirectly(间接通信)：

      P_A 发msg到中间实体(如mailbox)，P_B 从中间实体收msg

      P_A(msg)→ MailBox (msg)→ P_B

process(or processor) management

- (4) Job scheduling and process scheduling
  - Job scheduling:
    - ✓ 为作业分配必要资源，调入内存建立进程，并使之进入就绪队列。
  - Process scheduling:
    - ✓ 从就绪队列中选出进程，分配CPU，使之运行。
    - ✓ Schedule algorithms:
    - ✓ FCFS、优先权等

Main Memory Management(存储管理)

- Main Memory, MM:
  - a large array of words or bytes, each with its own address
  - A repository of quickly accessible data shared by CPU and I/O devices
  - A volatile storage device
  - Maybe the most architecture-specified component of OS
- Activities
  - Keeping track of memory usage(which parts of memory are currently being used and by whom)
  - Deciding which processes to load
  - Allocating and deallocating memory space

Main Memory Management(存储管理)

- GOAL：方便用户使用，且提高存贮器利用率

- (1) Memory allocation
  - ✓ 静态分配：
  - ✓ 动态分配：作业在内存中可移动

  需内存分配的数据结构及内存分配和回收功能

- (2) Memory protection
  - ✓例：设置上、下界寄存器，每条指令进行越界检查（一般是硬件实现）

- (3) Memory mapping，内存映射
  - ✓ 地址范围:地址
  - ✓ 逻辑空间:逻辑地址(相对地址)
  - ✓ 物理空间:物理地址(绝对地址)

- (4) Memory expansion，内存扩充
  - ✓ 利用虚存技术，从逻辑上扩充内存容量
  - ✓ 系统应有：请求调入/置换功能以支持虚存技术

I/O system management

- I/O subsystem(I/O子系统)
  - To hide the peculiarities of specific hardware devices from the user
- Maybe the most complicate component and has largest line of code (LOC)
  - Various device
- Consists of
  - Buffering, caching, and spooling
  - A general device-driver interface
  - Drivers(驱动) for specific hardware device

I/O system management

- GOAL: 提高I/O利用率和速度，方便用户
- (1) 缓冲管理
  - Buffer(缓冲区)：用来解决CPU-I/O矛盾，如：CPU快则应多创建缓冲区
- (2) device allocation(设备分配)
  - 包括：设备，设备控制器，I/O通道的分配和回收
- (3) Drivers
  - 控制设备进行实际的操作，包括读、写等以及向CPU发中断。
  - 设备处理/驱动程序应能根据用户I/O请求，自动地构成通道程序。

File management(文件管理)

- A file is a collection of related information defined by its creator.
  - Commonly, programs & data
  - A logical storage unit
- Activities
  - File creation and deletion
  - Directory(目录) creation and deletion
  - Support of primitives for manipulating files and directories
  - Mapping files onto secondary storage
  - File backup(备份) on stable (nonvolatile) storage media

File management(文件管理)

- GOAL: 方便用户，提供安全性
- (1) 文件存贮空间的管理
  - 例：文件系统根据文件长度自动分配连续或离散的扇区，并提供“一句柄”表示该文件。
- (2) Directory management(目录管理)
  - 使用户按名存取，提高速度。
- (3) 文件的读、写管理和存取控制（即保护）

Secondary storage management(辅存管理)

- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time.
- Proper management is of central importance
  - Entire speed of computer operation hinges on disk subsystem and its algorithms
- Activities
  - Free-space management
  - Storage allocation
  - Disk scheduling

Outline

- System components
- Operating System Services
- User Operating System Interface
- System Calls
- Operating System Design and Implementation
- Operating System Structure

Operating System Services

- One set of operating-system services provides functions that are helpful to the user:
  - User interface - Almost all operating systems have a user interface (UI)
    - ✓ Varies between Command-Line (CLI), Graphics User Interface (GUI), Batch
  - Program execution - The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)
  - I/O operations - A running program may require I/O, which may involve a file or an I/O device
  - File-system manipulation - The file system is of particular interest. Obviously, programs need to read and write files and directories, create and delete them, search them, list file information, permission management.

Operating System Services

- One set of operating-system services provides functions that are helpful to the user(cont.):
  - Communications – Processes may exchange information, on the same computer or between computers over a network
    - ✓ Communications may be via shared memory or through message passing (packets moved by the OS)
  - Error detection – OS needs to be constantly aware of possible errors
    - ✓ May occur in the CPU and memory hardware, in I/O devices, in user program
    - ✓ For each type of error, OS should take the appropriate action to ensure correct and consistent computing
    - ✓ Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system

Operating System Services

- Another set of OS functions exists for ensuring the efficient operation of the system itself via resource sharing
  - Resource allocation - When multiple users or multiple jobs running concurrently, resources must be allocated to each of them
    - ✓ Many types of resources - Some (such as CPU cycles, main memory, and file storage) may have special allocation code, others (such as I/O devices) may have general request and release code
  - Accounting - To keep track of which users use how much and what kinds of computer resources
  - Protection and security - The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other
    - ✓ Protection involves ensuring that all access to system resources is controlled
    - ✓ Security of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts
    - ✓ If a system is to be protected and secure, precautions must be instituted throughout it. A chain is only as strong as its weakest link.

Outline

- System components
- Operating System Services
- User Operating System Interface
- System Calls
- Operating System Design and Implementation
- Operating System Structure

User Operating System Interface

- Command Line Interface (CLI) or command interpreter allows direct command entry.
  - Sometimes implemented in kernel, sometimes by systems program
  - Sometimes multiple flavors implemented – shells
  - Primarily fetches a command from user and executes it
    - ✓ Sometimes commands built-in, sometimes just names of programs
    - – If the latter, adding new features doesn't require shell modification

User Operating System Interface

- User-friendly desktop metaphor interface
  - Usually mouse, keyboard, and monitor
  - Icons represent files, programs, actions, etc
  - Various mouse buttons over objects in the interface cause various actions (provide information, options, execute function, open directory (known as a folder)
  - Invented at Xerox PARC
- Many systems now include both CLI and GUI interfaces
  - Microsoft Windows is GUI with CLI “command” shell
  - Apple Mac OS X as “Aqua” GUI interface with UNIX kernel underneath and shells available
  - Solaris is CLI with optional GUI interfaces (Java Desktop, KDE)

Outline

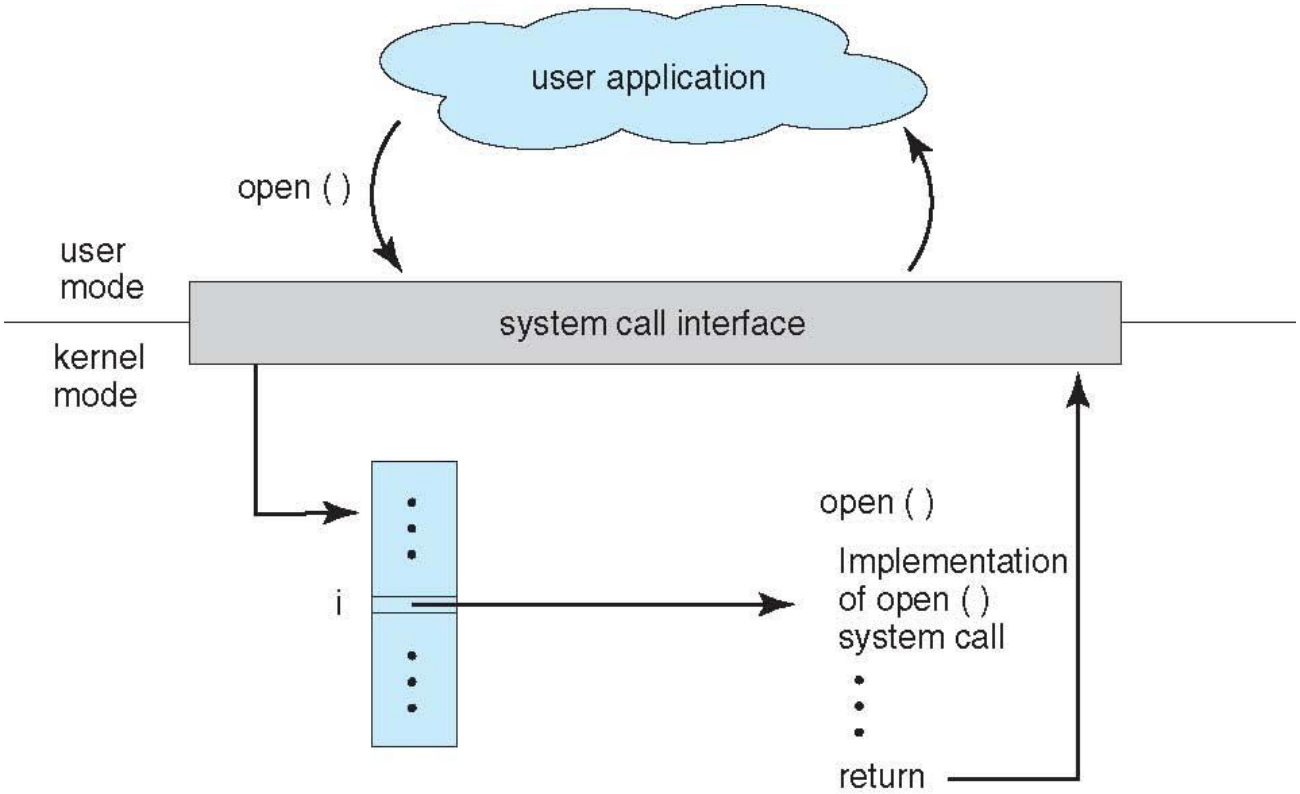
- System components
- Operating System Services
- User Operating System Interface
- System Calls
- Operating System Design and Implementation
- Operating System Structure

System Calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level Application Program Interface (API) rather than direct system call use
  - FOR
    - ✓ Program portability(可移植性)
    - ✓ System calls can often be more detailed and difficult
- Three most common APIs are
  - Win32 API for Windows,
  - POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X),
  - and Java API for the Java virtual machine (JVM)
- Why use APIs rather than system calls?

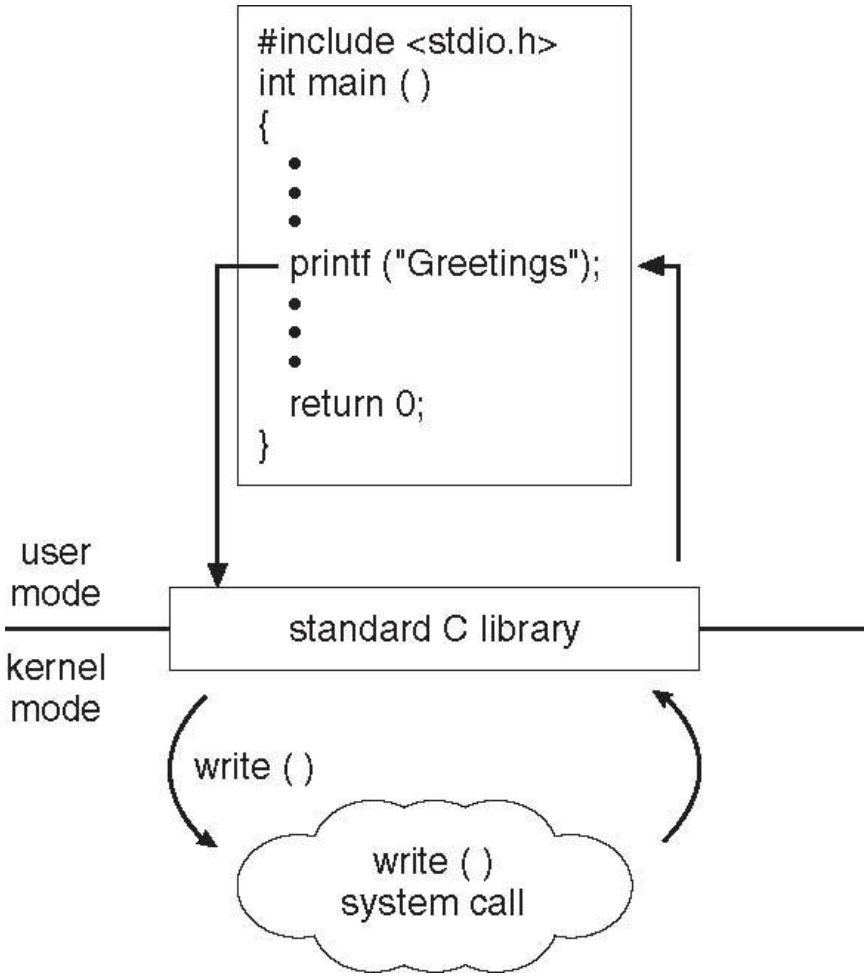
System Call Implementation

- Typically, a number associated with each system call
- Look up a system call table to locate intended function
- Invoke the intended function and returns status and values



Standard C Library Example

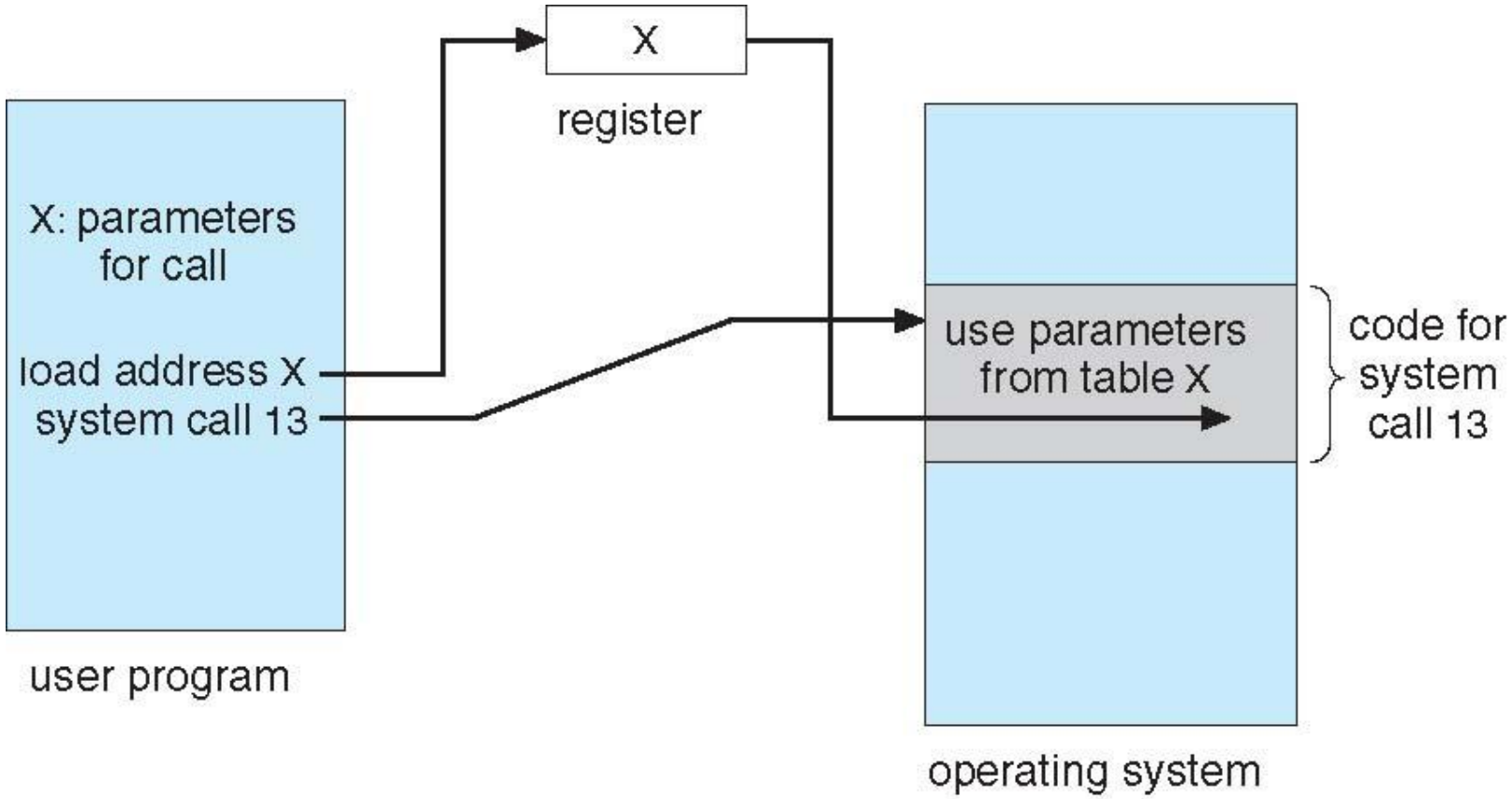
- C program invoking printf() library call, which calls write() system call



System Call Parameter Passing(参数传递)

- Often, more information is required than simply identity of desired system call
  - Exact type and amount of information vary according to OS and call
- Three general methods used to pass parameters to the OS
  - Simplest: pass the parameters in registers
    - ✓ In some cases, may be more parameters than registers
  - Parameters stored in a block, or table, in memory, and address of block passed as a parameter in a register
    - ✓ This approach is taken by Linux and Solaris
  - Parameters placed, or pushed, onto the stack by the program and popped off the stack by the operating system
- Block and stack methods do not limit the number or length of parameters being passed

Parameter Passing via Table



Outline

- System components
- Operating System Services
- User Operating System Interface
- System Calls
- Operating System Design and Implementation
- Operating System Structure

OS design and implementation

- Design and Implementation of OS are not “solvable”, but some approaches have proven successful
- Internal structure of different Operating Systems can vary widely
- The first problem in designing an OS is to define goals(目标) and specifications(规格).---requirement(需求)
- User goals and System goals
  - User goals – operating system should be convenient to use, easy to learn, reliable, safe, and fast
  - System goals – operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient

OS design and implementation

- One important principle: the separation of policy from mechanism(机制和策略相分离)
  - mechanisms(机制) determine how to do something(如何来做);
  - policies(策略) determine what will be done(做什么).
  - Example: timer(定时器)、priority(优先级)；microkernel VS. Apple Macintosh

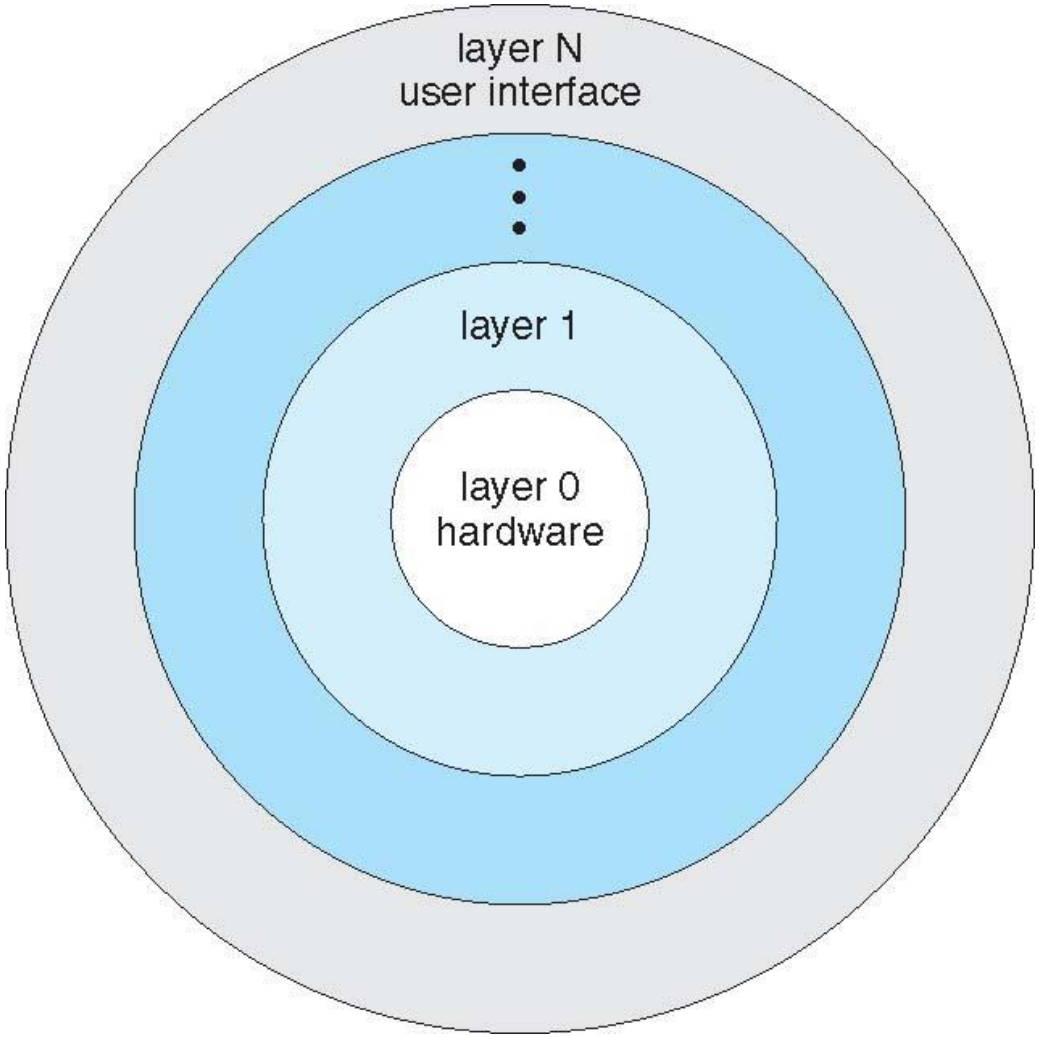
Outline

- System components
- Operating System Services
- User Operating System Interface
- System Calls
- Operating System Design and Implementation
- Operating System Structure

Operating System Structure

- Simple Structure
- Layer Structure
- Microkernel System Structure(微内核)
- Modules
- Virtual Machines

Layered Operating System



End of Chapter 1

操作系统的特征

- 1并发
  - 并行 vs. 并发
  - ✓ 并行是指两或多个事件在同一时刻发生。
  - ✓ 并发是两或多个事件在同一时间间隔内发生。
  - Program vs. Process(进程)
  - ✓ Program：静态实体；
  - ✓ Process：动态实体

    - A program in execution.

    -是系统中能独立运行并作为资源分配的基本单位。

    -引入线程后，独立运行的单位变为线程。

操作系统的特征

- 2共享
  - 系统中资源可供内存中多个并发执行的进程共同使用
  - 互斥共享 VS. 同时访问
  - ✓ 互斥共享：一段时间只允许一个进程访问该资源
  - ✓ 同时访问：微观上仍是互斥的 临界资源：在一段时间内只允许一个进程访问的资源

- 并发和共享是操作系统的两个最基本的特征。

操作系统的特征

- 3虚拟
  - 虚拟：通过某种技术把一个物理实体变为若干个逻辑上的对应物。
  - 若n是某一物理设备所对应的虚拟的逻辑设备数，则虚拟设备的速度必然是物理设备速度的1/n。
- 4异步
  - 运行进度不可预知。