1.3 基本服务和用户接口

1.3.1 基本服务和用户接口

1.3.2 程序接口与系统调用

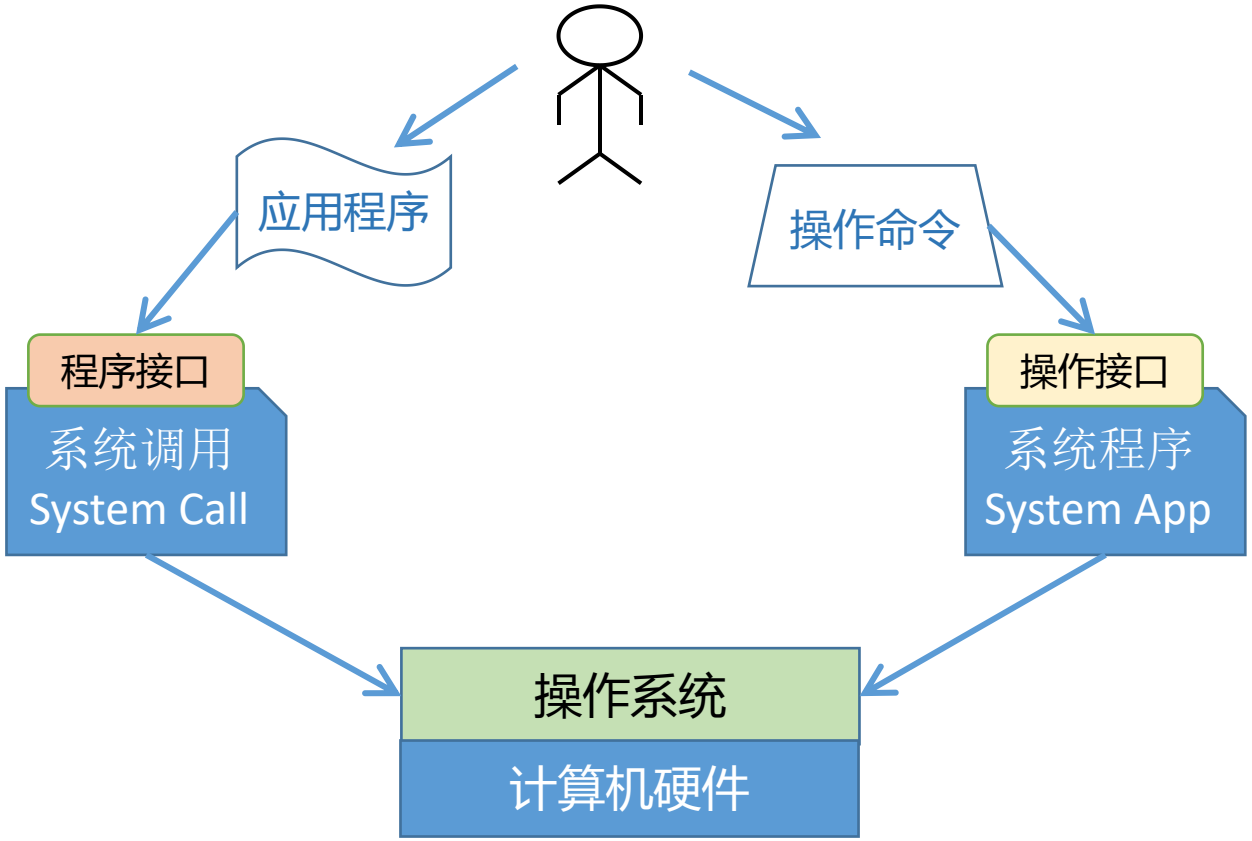
1.3.3 操作接口与系统程序

1.3.1 基本服务和用户接口

- 共性服务：
  - 创建程序：提供程序的编辑、调试、编译等生成工具
  - 执行程序：装入内存、执行、异常报告、终止程序
  - 数据I/O：以简单方式提供给用户进行I/O
  - 信息存取：文件操作
  - 通信服务：进程通信
  - 错误检测和处理
- 其他功能：
  - 资源分配
  - 统计
  - 保护

1.3.1 基本服务和用户接口

- 操作系统通过程序接口和操作接口，向用户提供服务：



1.3.1 基本服务和用户接口

- 操作系统通过程序接口和操作接口，向用户提供服务：
  - 程序接口：应用编程接口API（Application Programming Interface），允许运行程序调用操作系统的服务和功能
  - 操作接口：作业级接口，操作系统为用户提供的操作控制计算机工作和提供服务手段的集合

1.3.1 基本服务和用户接口

- 程序接口：由一组系统调用（System Call)）组成，用户程序使用“系统调用”就可获得操作系统的底层服务；使用或访问系统的各种软硬件资源
- 系统调用的主要功能是使用户可以使用操作系统提供的有关设备管理、文件系统、进程控制进程通讯以及存储管理方面的功能，而不必要了解操作系统的内部结构和有关硬件的细节问题，从而减轻用户负担和保护系统以及提高资源利用率

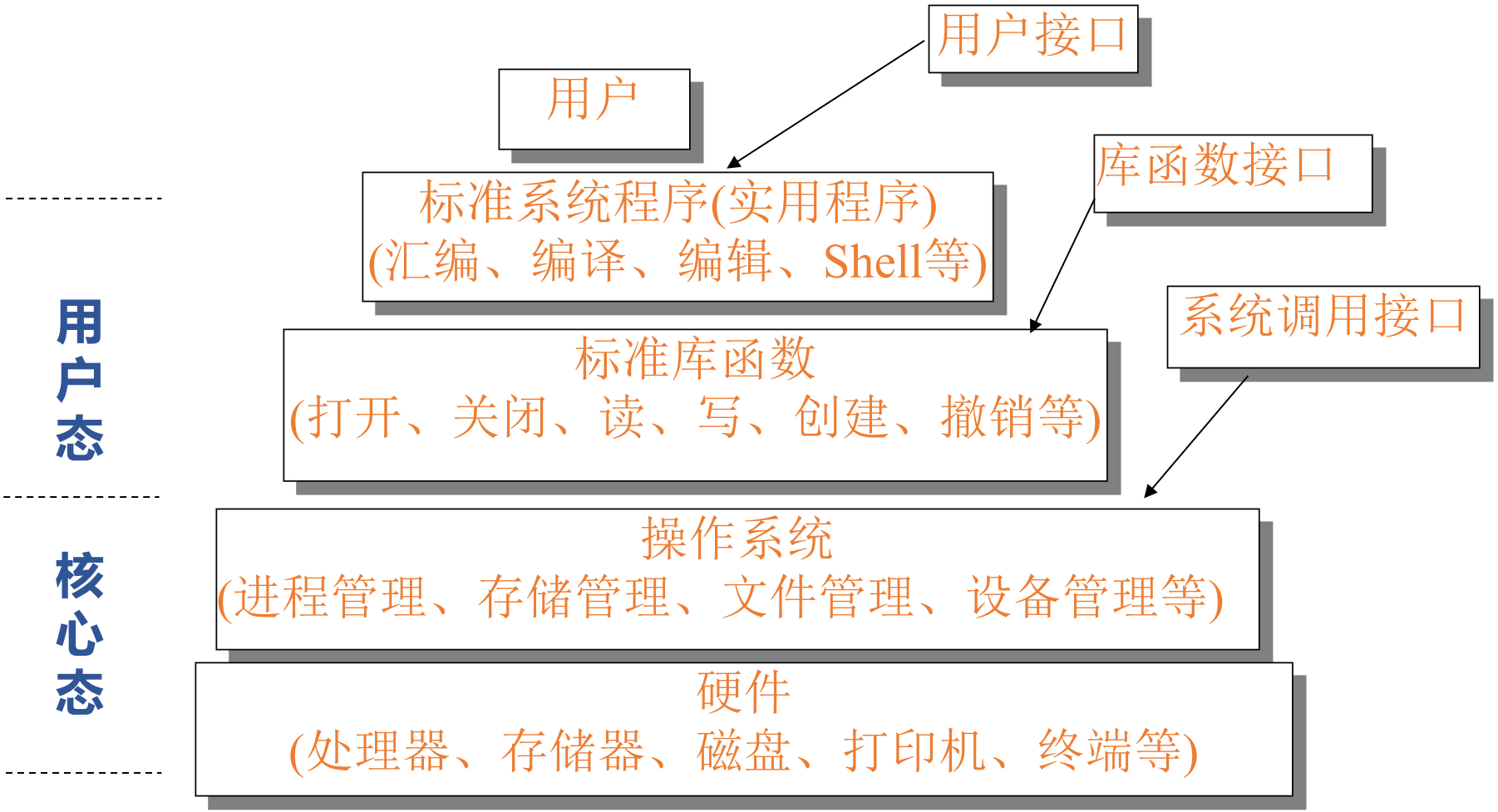
1.3.1 基本服务和用户接口

- 操作接口：为用户操作控制计算机工作和提供服务的手段的集合，通常有
  - 操作控制命令
  - 图形操作界面
  - 批处理系统提供的作业控制语言（命令）等

1.3.2 程序接口和系统调用

- 系统调用
  - 为了扩充机器功能、增强系统能力、方便用户使用而在内核中建立的特殊的公共子程序，它是用户程序或其它系统程序获得操作系统服务的唯一途径，系统调用也称为广义指令
- 系统调用与机器指令的区别：
  - 机器指令由硬件实现
  - 广义指令（系统调用）是由操作系统在机器指令基础上实现的过程或子程序

1.3.2 程序接口和系统调用



UNIX/Linux系统程序、库函数、系统调用的分层关系

1.3.2 程序接口和系统调用

- 内核态（kernel mode）：又称管态、核心态。运行在该模式的代码，可以无限制地对系统存储、外部设备进行访问。程序受硬件保护，用户不能随意篡改内容。
- 用户态（user mode）：又称目态、普通态。执行的代码被硬件限定，不能进行例如写入其他进程的存储空间这样的操作。

1.3.2 程序接口和系统调用

- 系统调用可以看作是一个所有进程共享的子程序库，但是它是在特权方式下运行，可以存取核心数据结构和它所支持的用户级数据。

系统调用作用 一：

所有进程只能通过系统调用访问系统资源，由内核基于权限提供一致性规则对资源访问进行裁决，保证系统安全性。

1.3.2 程序接口和系统调用

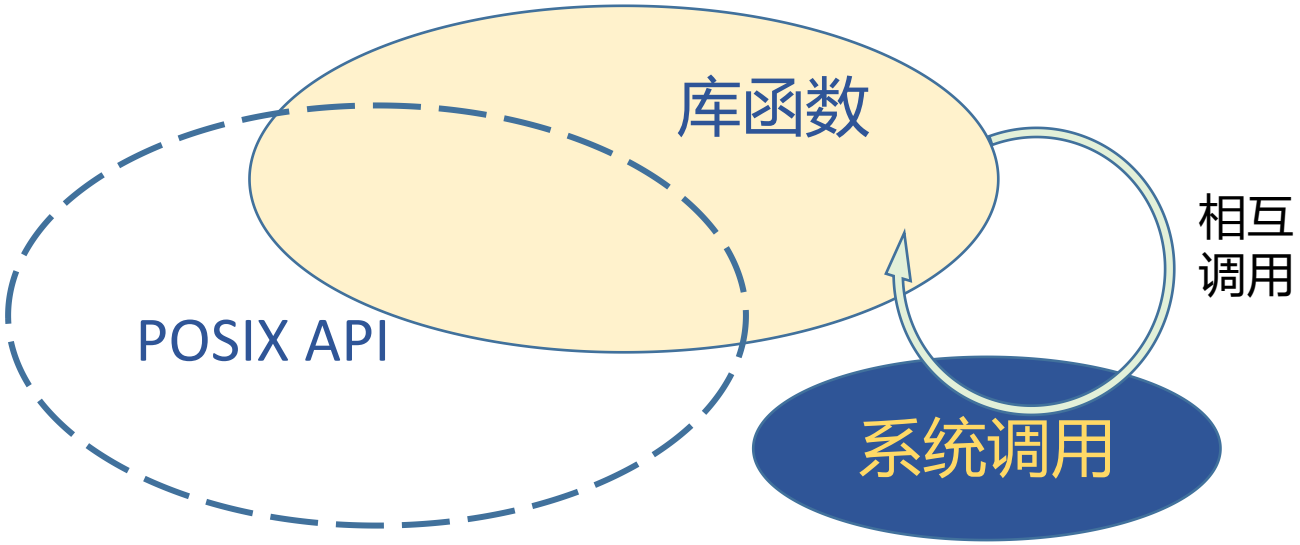
- 使用户可以使用操作系统提供的有关设备管理、文件系统、进程控制进程通讯等方面的功能，而不必了解操作系统的内部结构和有关硬件的细节问题，减轻用户负担和保护系统以及提高资源利用率

系统调用作用 二：

对系统资源的调用方法进行抽象，提供一致性接口，避免用户在使用资源时发生错误，提高编程效率。

1.3.2 程序接口和系统调用

- POSIX：Portable Operating System Interface
  - 便于应用程序在不同操作系统间的移植
  - Unix/Linux基本遵循 POSIX 标准
  - Windows NT 声称部分实现标准



1.3.2 程序接口和系统调用

1. 进程和作业管理：进程的创建、装入、执行、撤销、终止，进程属性的获取和设置
2. 文件管理：文件的建立、打开、读写、关闭、删除，文件属性的获取和设置
3. 设备管理：设备的申请、输入输出、释放、重定向，设备属性的获取和设置
4. 存储管理：内存的申请和释放
5. 进程通信：通信连接的建立、连接和断开、信息的发送和接受
6. 信息维护：日期、时间及系统数据的获取和设置

1.3.2 程序接口和系统调用

- 例：Windows支持API（应用编程接口）的三个组件：
  - Kernel：包含了多数操作系统函数，如内存管理、进程管理
  - User：集中了窗口管理函数，如窗口创建、撤销、移动、对话等相关函数
  - GDI：提供画图函数、打印函数
- Windows将三个组件置于动态链接库DLL中
  - kernel32.dll ， user32.dll, gdi32.dll
  - System32/, SysWOW64/

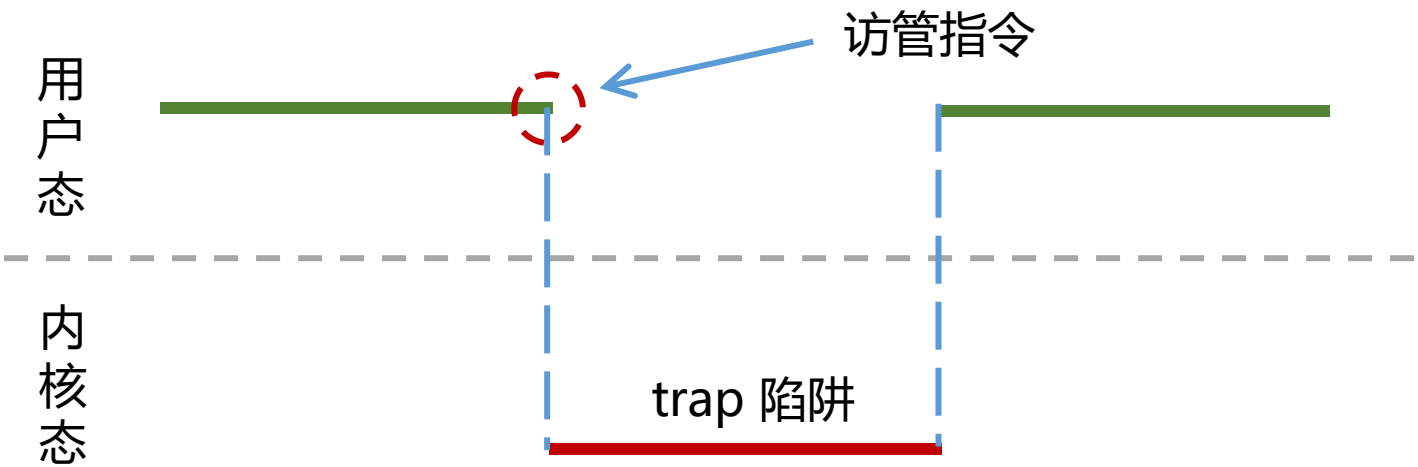
1.3.2 程序接口和系统调用

Windows和UNIX/Linux的部分系统调用：

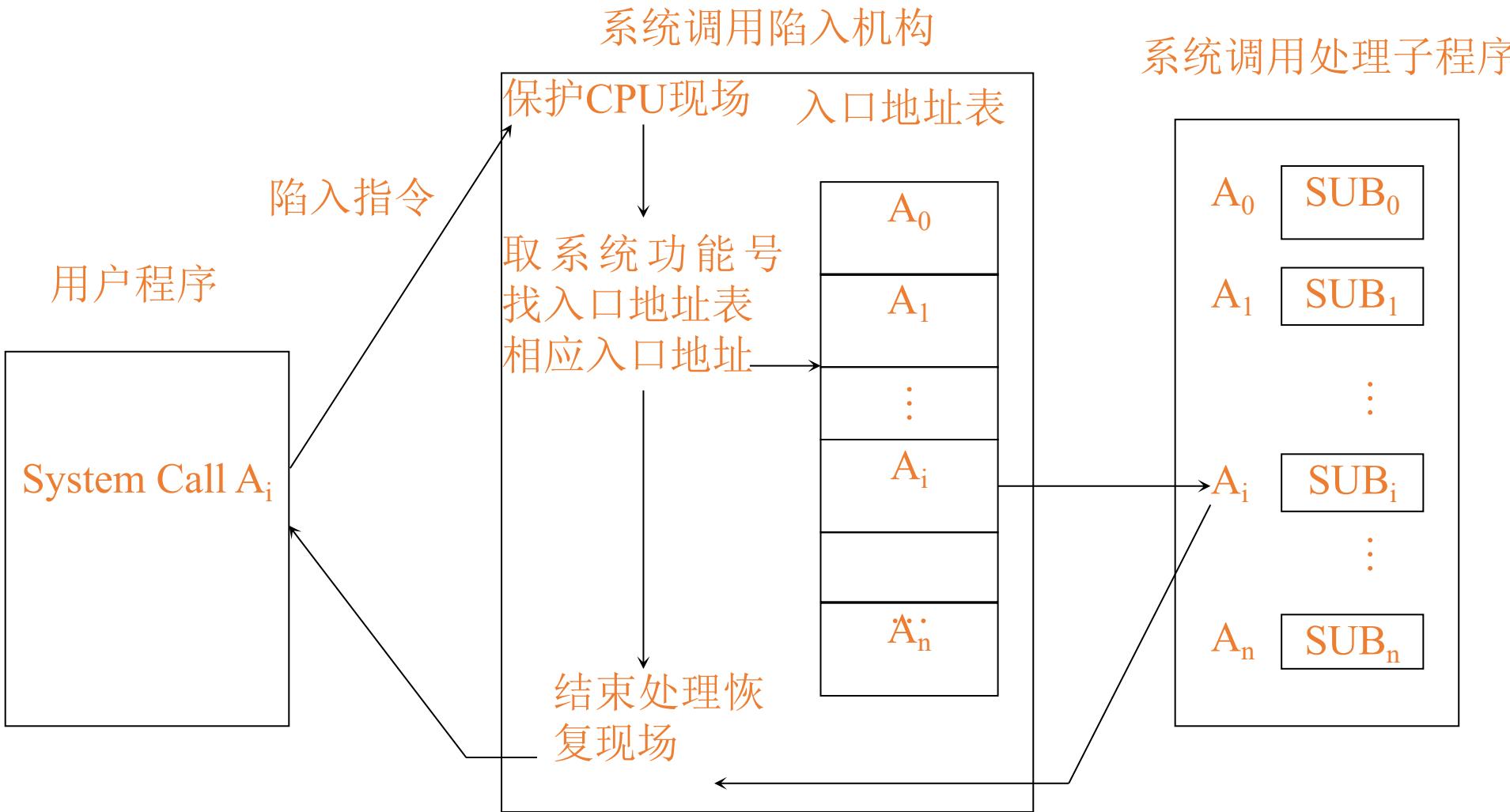
UNIX/Linux	Win32	说明
fork	CreatProcess	创建进程
waitpid	WaitForSingleObject	等待进程终止
open/close	CreatFile/CloseHandle	创建或打开/关闭文件
read/write	ReadFile/WriteFile	读/写文件
lseek	SetFilePointer	移动文件指针
mkdir/rmdir	Creat/Remove Directory	建立/删除目录
stat	GetFileAttributesEx	获得文件属性

1.3.2 程序接口和系统调用

- 异常处理机制：在操作系统中，系统调用的实现机制
  - 通过访管指令（supervisor）、自陷指令（trap）或中断指令（interrupt），引起处理器中断，实现系统调用服务例程访问。



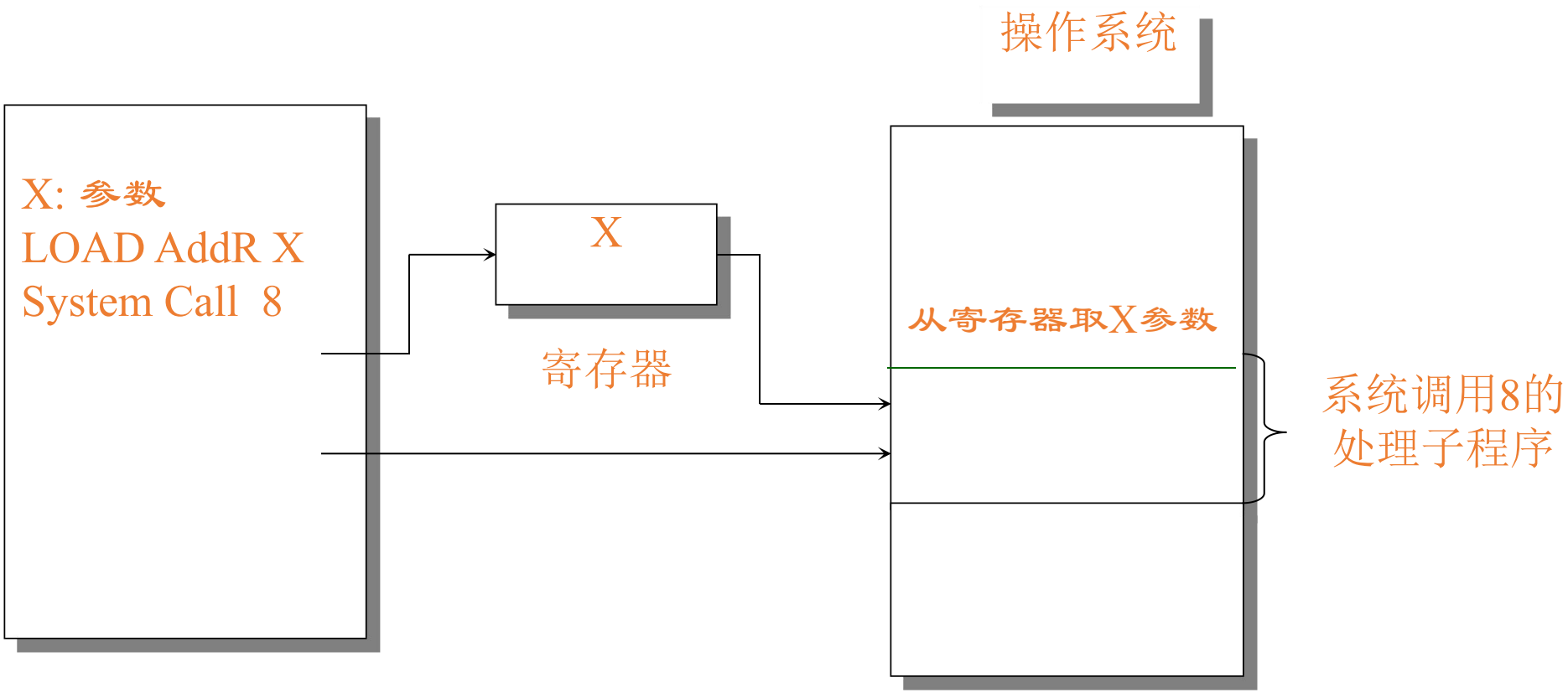
1.3.2 程序接口和系统调用



1.3.2 程序接口和系统调用

- 系统调用的参数传递
  - 不同的系统调用需传递给系统调用处理程序不同的参数
  - 系统调用执行的结果也要以参数形式返回给用户程序
- 用户程序和系统调用之间的参数传递方法：
  1. 由访管指令或陷入指令自带参数，可以规定指令之后的若干个单元存放的是参数，这称为直接参数；或者在指令之后紧靠的单元中存放参数的地址，这称为间接参数
  2. 通过CPU的通用寄存器传递参数，或在内存的一个块或表中存放参数，其首地址送入寄存器，实现参数传递
  3. 在内存中开辟专用堆栈区域传递参数

1.3.2 程序接口和系统调用



传递参数的一种方法

1.3.2 程序接口和系统调用

- 系统调用与函数调用的区别

1. 调用形式不同
   - 过程（函数）使用一般调用指令，其转向地址包含在跳转语句中
   - 系统调用不包含处理程序入口，仅仅提供功能号，按功能号调用
2. 被调用代码的位置不同
   - 在过程（函数）调用中，调用程序和被调用代码在同一程序内，经过连接编译后作为目标代码的一部分。当过程（函数）升级或修改时，必须重新编译连接
   - 系统调用的处理代码在调用程序之外（在操作系统中），系统调用处理代码升级或修改时，与调用程序无关

1.3.2 程序接口和系统调用

- 系统调用与函数调用的区别

3. 提供方式不同
   - 过程（函数）往往由编译系统提供，不同编译系统提供的过程（函数）可以不同
   - 系统调用由操作系统提供，一旦操作系统设计好，系统调用的功能、种类与数量就固定不变了
4. 调用实现不同
   - 程序使用一般机器指令（跳转指令）来调用过程（函数），是在用户态运行的
   - 程序执行系统调用，是通过中断机构来实现的，需要从用户态转变到核心态，在管理态执行

1.3.3 操作接口和系统程序

- 作业是用户提交给操作系统进行计算的一个独立任务
- 控制作业时，用到的两类作业级接口：
  - 联机作业控制接口：交互型作业处理
    - 字符型用户界面：字符界面（命令行、批处理）
    - 图形化用户界面
  - 脱机作业控制接口：批处理作业处理

1.3.3 操作接口和系统程序

- 命令行方式是以命令为基本单位来完成预定的工作任务
  - 每个命令以命令行的形式输入并提交给系统
  - 一个命令行由命令动词和一组参数构成，其一般形式如下：

Command *Option(s)* *Argument(s)*

ls [-**ABCFGHLOPRSTUW@abcdefghiklmnopqrstuwx1**] [*file* ...]

1.3.3 操作接口和系统程序

- Linux常用的五大类命令：
  1. 文件管理类
     - cd、chmod、chgrp、comm、cp、crypt、diff、file、find、ln、ls、mkdir、mv、od、pr、pwd、rm、rmdir
  2. 进程管理类
     - at、kill、mail、nice、nohup、ps、time、write、mesg
  3. 文本加工类
     - cat、crypt、grep、norff、uniq、wc、sort、spell、tail、troff
  4. 软件开发类
     - cc、f77、login、logout、size、yacc、vi、emacs、dbs、lex、make、lint、ld
  5. 系统维护类
     - date、man、passwd、stty、tty、who

1.3.3 操作接口和系统程序

- 命令解释程序
  - 主要功能：接受和执行一条用户从键盘输入的命令，它通常保存一张命令名字（动词）表，其中记录着所有操作命令及其处理程序的入口地址或有关信息
  - 当新的批作业被启动，或新的交互型用户登录时，系统就自动地执行命令解释程序，它负责读入控制卡或命令行，并作出相应解释和执行

1.3.3 操作接口和系统程序

- 命令解释程序
  - 处理过程：系统启动命令解释程序，输出命令提示符，等待键盘中断。用户打入命令并按回车换行，申请键盘中断
  - CPU响应后，控制权交给命令解释程序，它读入命令缓冲区内容，分析命令、接受参数
  - 若为简单命令立即转向命令处理代码执行。否则查找命令处理文件，装入主存，传递参数，将控制权交给其执行
  - 命令处理结束后，再次输出命令提示符，等待下一条命令

1.3.3 操作接口和系统程序

- 命令解释程序
  - 实现方式：一种是它自身包含了命令的执行代码
  - 另一种是由专门的“系统程序” 实现，自身不含命令处理代码，也不进行处理，仅仅把这条命令对应的命令文件装入内存执行

1.3.3 操作接口和系统程序

- 系统程序
  - 又称标准程序或实用程序（Utilities）。不属于操作系统的核心，但为用户程序的开发、调试、执行、和维护解决带有共性的问题或执行公共操作
  - 操作系统常以外部操作命令形式向用户提供许多系统程序，它的功能和性能很大程度上反映了操作系统的功能和性能，用户看待操作系统，不是看系统调用怎么样，而是看系统程序怎么样

1.3.3 操作接口和系统程序

- 系统程序的分类
  - 文件管理
    - 文件和目录的建立、删除、复制、改名、打印、列表、转存等管理工作
  - 状态信息
    - 获得日期、时间、可用内存、磁盘空间数量、用户数及其它状态信息
  - 程序设计语言
    - 支持编译、汇编、解释程序
  - 程序的装入和执行支持
    - 绝对装入工具、重定位装入工具、链接编辑程序、调试程序等
  - 通信
    - 机间通信、电子邮件、远程登录、文件传输
  - 其它软件工具
    - Web浏览器、字处理工具、电子表格、数据库系统、画图软件包、统计分析包、游戏程序等