

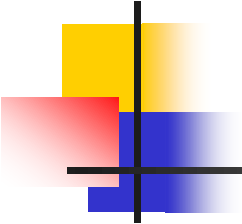
操作系统

Operating Systems

王成耀

北京科技大学计算机科学与技术系

Department of Computer Science & Technology
University of Science & Technology Beijing
2018



总学时：64学时

讲授：48学时，1-12周

实验：16学时，10、11、14、15周

分2组，每组2个班

最终成绩的构成：

平时成绩占30%（实验20%，考勤（课堂练习）、作业等10%）

期末考试（非集中考试）成绩占70%



教材与参考书

- [1] 张尧学，宋虹，张高．计算机操作系统教程（第4版）．清华大学出版社，2013（**教材**）
- [2] Andrew S. Tanenbaum, Herbert Bos. 陈向群，马洪兵等译．现代操作系统（原书第4版）．机械工业出版社，2017
- [3] 汤子瀛，哲凤屏，汤小丹．计算机操作系统．西安电子科技大学出版社，1996
- [4] William Stallings. 陈向群，陈渝等译．操作系统 - 精髓与设计原理（第八版），电子工业出版社，2017
- [5] Randal E. Bryant, David R. O'Hallaron. 龚奕利，贺莲译．深入理解计算机系统（原书第3版）．机械工业出版社，2016
- [6] 邹恒明．计算机的心智：操作系统之哲学原理，机械工业出版社，2009
- [7] Abraham Silberschatz, Peter Galvin, Greg Gagne. Applied Operating System Concepts, John Wiley & Sons, Inc., 2000
- [8] 孟庆昌，牛欣源．Linux教程（第2版），电子工业出版社，2007



讲授的主要内容

第1章 操作系统概述

第2章 进程管理

第3章 进程调度

第4章 内存管理

第5章 进程与内存管理实例

第6章 文件系统

第7章 I/O设备管理

第8章 Linux文件系统

第9章 死锁

学时安排：进程管理（多） 内存管理 文件系统 设备管理（少）



第1章 操作系统概述

1.1 认识操作系统

1.2 操作系统发展过程中形成的一些概念

1.3 OS对运行环境的要求

1.4 典型OS实例

1.5 现代操作系统的基本特征

1.6 从不同角度认识操作系统

1.7 学习操作系统课程要达到的目标

1.1 认识操作系统

1. 程序是如何执行的?

```
#include <stdio.h>

void swap();

int buf[2] = {1, 2};

int main()
{
    swap();
    printf("buf = %d, %d\n", buf[0], buf[1]);
}
```

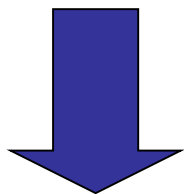
a) main.c文件

```
extern int buf[];
int *bufp0 = &buf[0];
int *bufp1;

void swap()
{
    int temp;

    *bufp1 = &buf[1];
    temp = *bufp0;
    *bufp0 = *bufp1;
    *bufp1 = temp;
}
```

b) swap.c文件



预处理
编译
链接

可执行程序（机器代码）

1.1 认识操作系统

可执行程序（示意）：**机器指令**

080483b4 <main>:

80483b4: 55
80483b5: 89 e5
80483b7: 83 e4 f0
80483ba: e8 09 00 00 00

push %ebp
mov %esp, %ebp
and \$0xffffffff0, %esp
call 80483c8<swap>

汇编语言指令

.

080483c8 <swap>:

80483c8: 55
80483c9: 8b 15 5c 94 04 08

push %ebp
mov 0x804945c, %edx

.

08049454 <buf>:

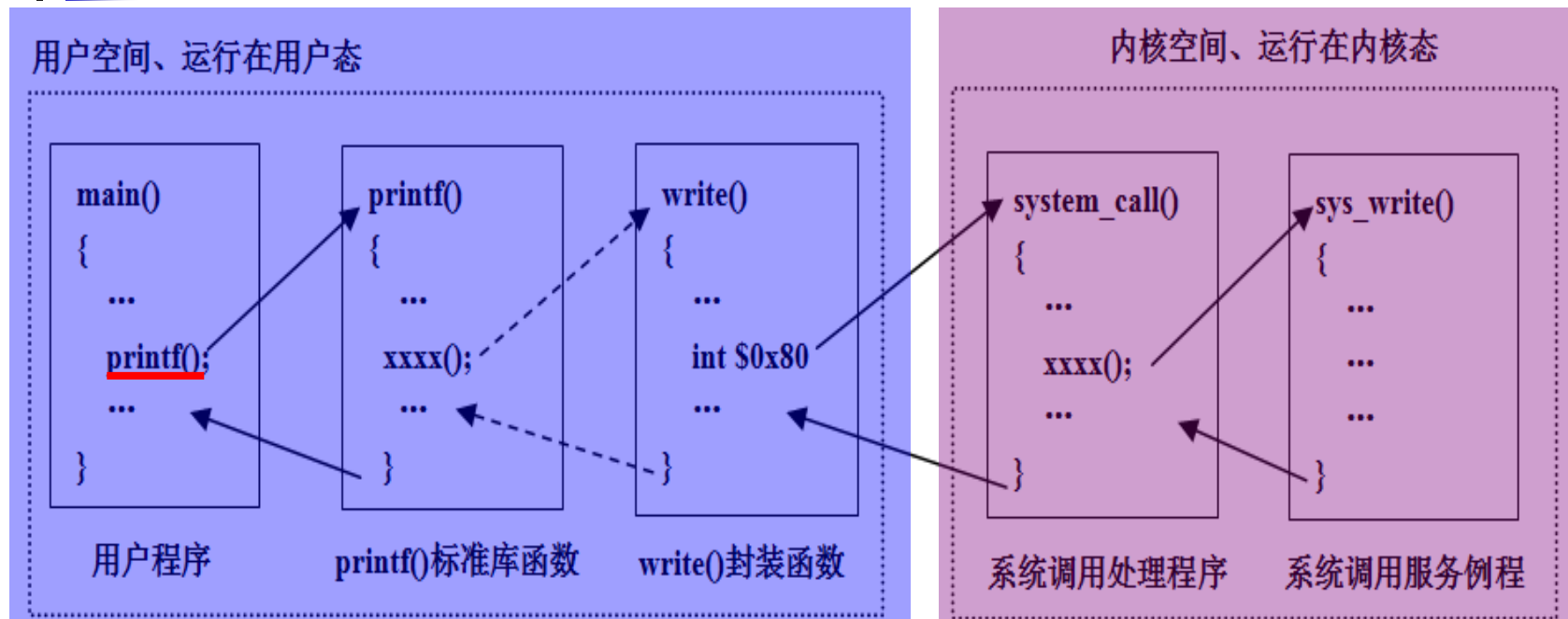
8049454: 01 00 00 00 02 00 00 00

数据

0804945c <bufp0>

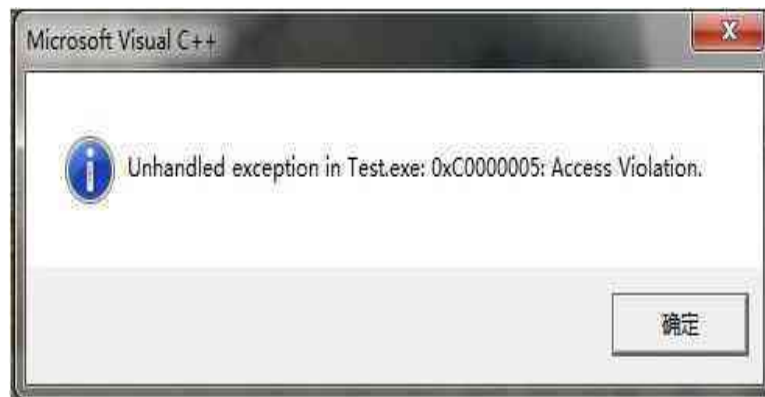
804945c: 54 94 04 08

1.1 认识操作系统



程序执行的整个生命周期都受操作系统控制：

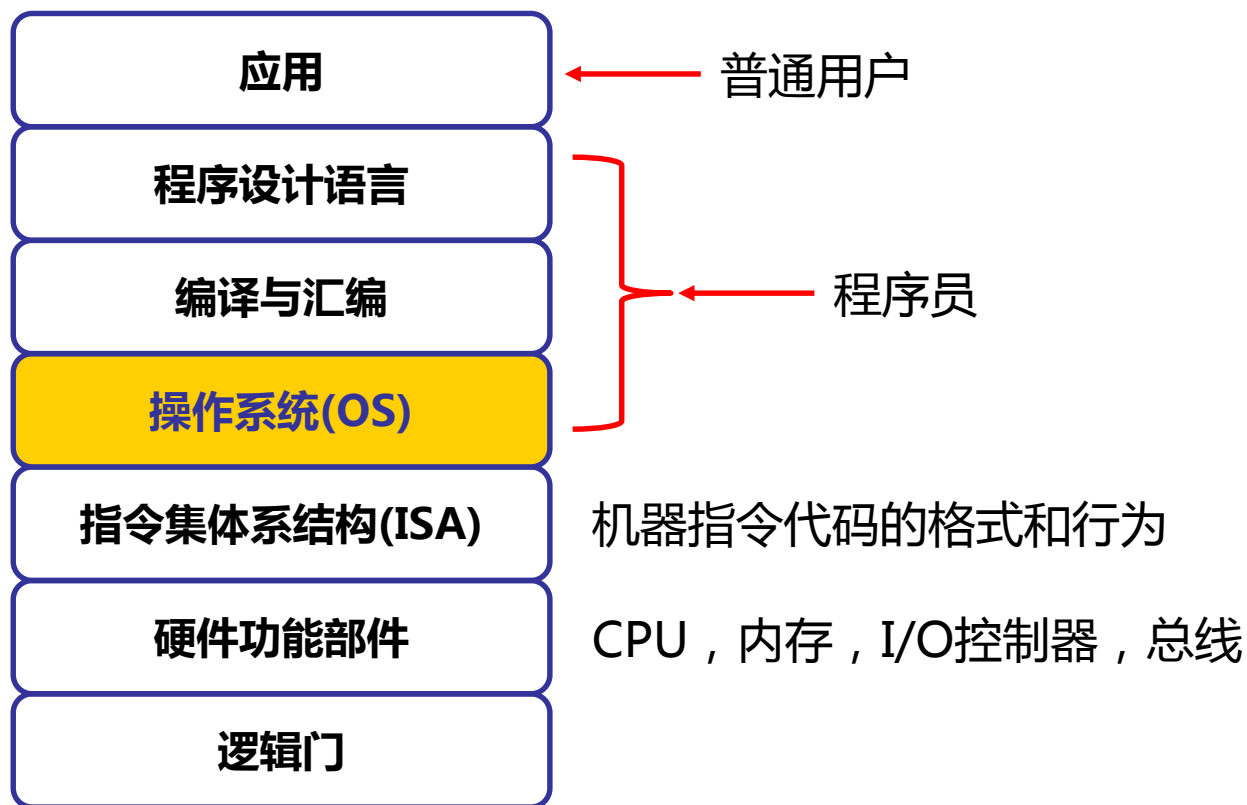
- ✓ 建立程序的内存映像，CPU的分配
- ✓ 内存分配、释放
- ✓ 文件访问，输入输出（I/O）
- ✓ 中断（异常）处理



1.1 认识操作系统

2. 什么是操作系统(Operating System , OS) ?

1) OS在计算机系统的位置





什么是操作系统(Operating System, OS)

2) OS以什么形式出现?

是一组程序。

OS与普通程序有何区别?



什么是操作系统(Operating System, OS)

3) OS的作用 (功能)

(1) 一个虚拟机 (Virtual Machine) - 用户观点

让用户 (程序员) 在使用计算机时不涉及计算机硬件的细节, 使硬件细节和用户 (程序员) 隔离开来, 即建立一种简单的高度抽象。

用户与计算机之间的接口。

- ✓ 命令接口 (面向普通用户) : 命令行, GUI, 命令脚本
- ✓ 编程接口 (面向程序员) : 系统调用, 高级语言库函数

如果没有OS, 计算机可以使用吗?



OS的作用

(2) 一个资源管理器：管理系统的软硬件资源 - 系统观点

硬件资源：构成计算机系统所必须配置的所有硬件：

CPU、内存、时钟、磁盘、显示器（适配器）、网络接口，．．．。

软件资源：程序和数据（文件）。

- ✓ 进程管理：程序的调度；处理机（CPU）的分配等；
- ✓ 内存管理：内存分配、释放与保护；内存扩充等；
- ✓ 文件管理：文件存储空间管理；文件存取；文件访问控制等；
- ✓ I/O设备管理：设备分配；缓冲区管理等。



什么是操作系统(Operating System, OS)

4) OS的定义

OS是硬件之上的**第1层软件**（系统软件）

是一组**程序**，

用来有效控制和**管理**计算机系统的各类**资源**（硬件和软件资源：设备、文件、存储器、CPU、程序（进程）），

以方便用户使用计算机（用户和计算机的**接口**）。



1.2 操作系统发展过程中形成的一些概念

1. 作业(Job)

从输入开始到输出结束，用户要求计算机所做的一次业务处理的全部工作。

作业由顺序的一组作业步组成。

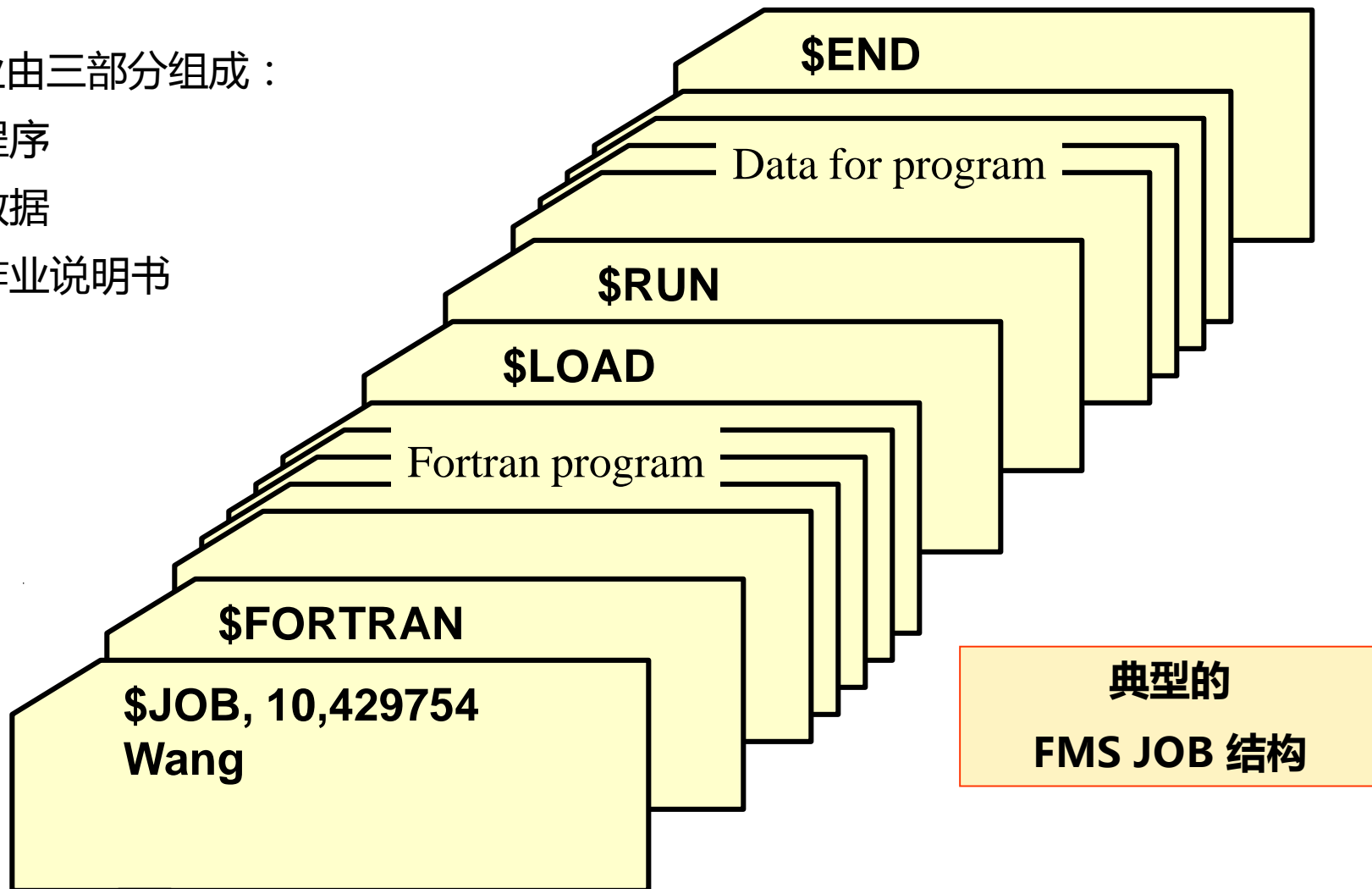
作业的概念来源于批处理系统。

分时系统中一般不存在传统作业的概念。

作业

作业由三部分组成：

- ✓ 程序
- ✓ 数据
- ✓ 作业说明书





作业

运行一个作业的步骤：

- 1) 将程序写在纸上（用高级语言或汇编语言）
- 2) 穿孔成卡片，再将卡片盒交给操作员
- 3) 计算结果从打印机上输出
- 4) 操作员到打印机上撕下运算结果送到输出室
- 5) 程序员稍后可从输出室取到结果
- 6) 操作员从输入室的卡片盒中读入另一个任务
- 7) 如果需要FORTRAN编译器，还要把它取来读入计算机

缺点：机时在走来走去浪费掉



1.2 操作系统发展过程中形成的一些概念

2. 批处理 (batch)

为改进内存和I/O设备之间的吞吐量

IBM 7094计算机引入了I/O 处理机概念

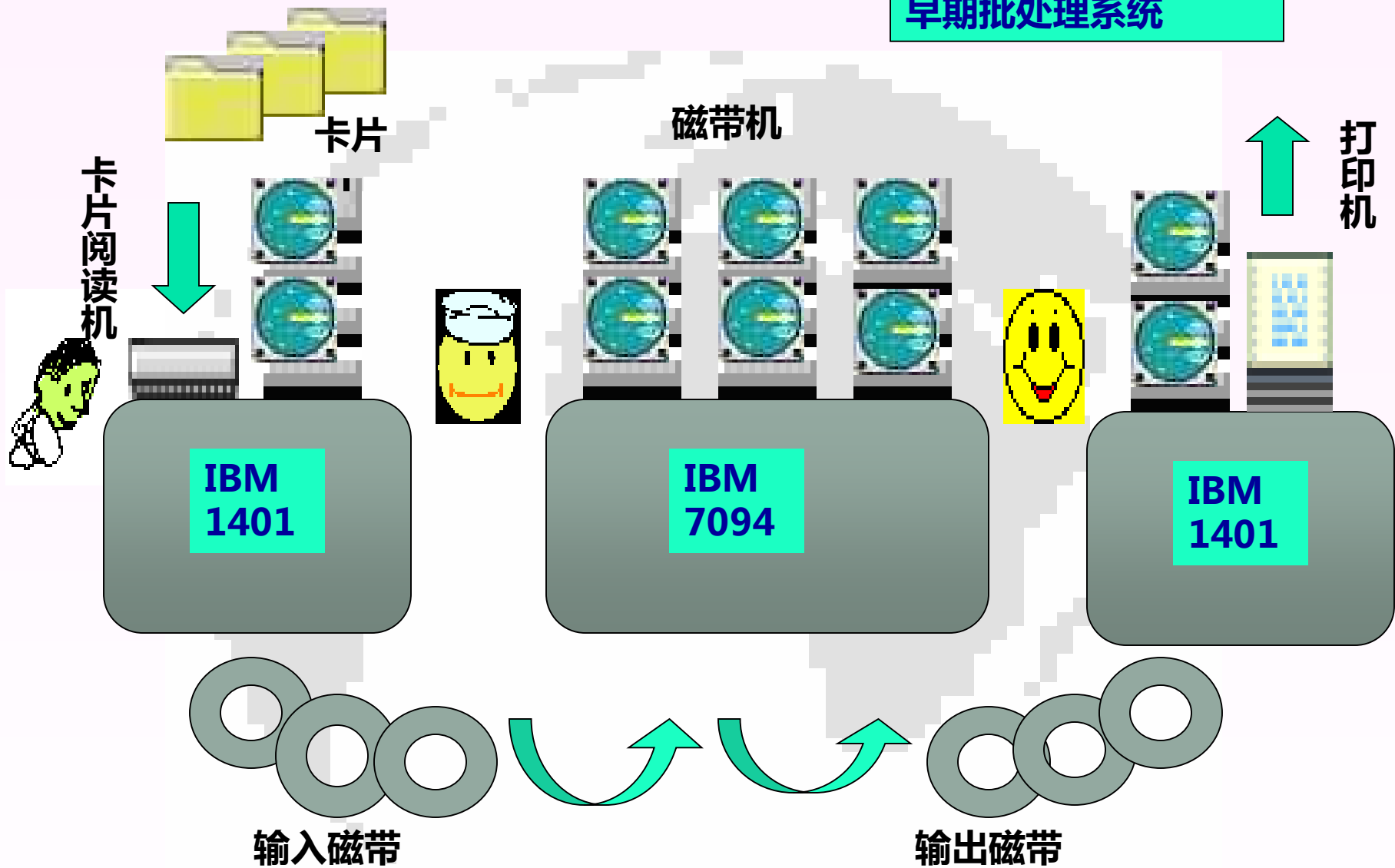
思想：

在输入室收集全部的作业，用一台相对便宜的计算机（IBM 1401计算机），
将作业读到磁带上

再用较昂贵的计算机（IBM7094）完成真正的计算

一批作业构成一个作业队列，依次处理。

早期批处理系统





批处理

批处理的含义：

- ✓ 无交互能力：作业从提交到完成，用户不能与之交互；
- ✓ 从传统的作业 - > 命令文件的扩展

把一系列命令放在一个文件中，称之为命令文件

用文件名作为命令名来执行

批处理命令可以是专门的命令，也可是系统的基本命令；还有有关的控制结构，包括循环、分支、转移等，构成一套特殊的命令语言，可以接受参数，使用变量、宏替换等

缺点？



批 处 理

成批处理：

用户不能干预自己作业的运行

一旦发现作业错误不能及时改正

延长了软件开发时间

一般只适用于成熟的程序或大型的计算程序



1.2 操作系统发展过程中形成的一些概念

3. 单道程序与多道程序

单道程序：

在内存中只能有一个用户程序（从进入到结束）

若当前程序因等待I/O而暂停，则CPU空闲

对于CPU操作密集的科学计算问题，浪费时间少

对于商业数据处理，I/O等待时间常占80% - 90%。

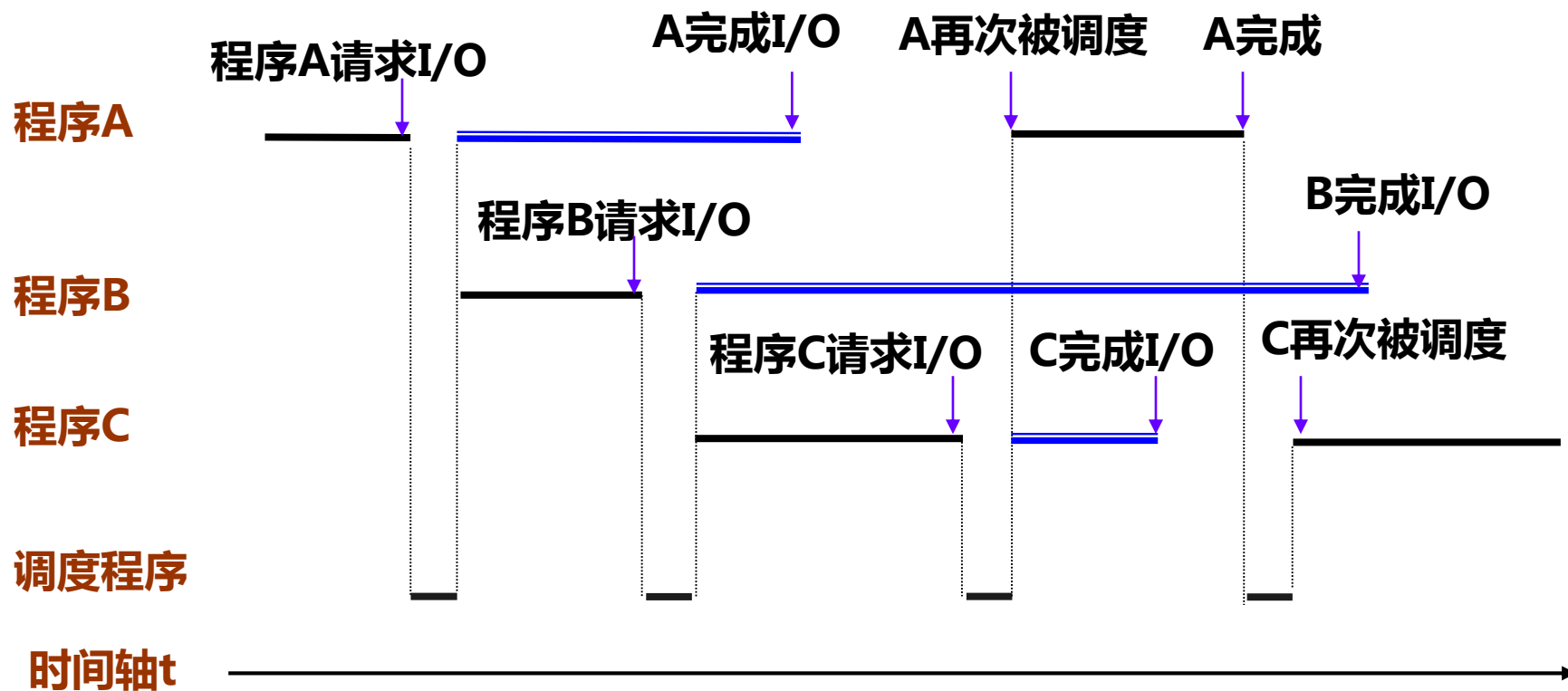
多道程序：

在内存中存放多个用户程序，同时处于**可运行状态**。

当一个程序等待I/O时，另一个程序可以使用CPU。

1.2 操作系统发展过程中形成的一些概念

多道程序示意图：



单线(黑色)表示程序占用CPU，双线(蓝色)表示外设在执行相应程序的I/O请求



1.2 操作系统发展过程中形成的一些概念

4. 多道批处理系统

批处理系统中引入多道程序技术

与单道批处理系统相比：

- ✓ 系统吞吐量（单位时间内完成的总工作量）大；
- ✓ 资源利用率高；
- ✓ 周转时间（作业从进入系统到完成所经历的时间）长。



1.2 操作系统发展过程中形成的一些概念

5. 分时系统 (Time-sharing System)

多个用户 (程序) 共享一台计算机 , 按时间片 (time slice) 轮流使用。

时间片 : OS将CPU时间划分为若干个片段

分时的特点:

- ✓ 多路性: 同时有多个用户 (程序) 使用一台计算机
宏观上 : 是多个人 (程序) 同时使用一个CPU
微观上 : 多个人 (程序) 在不同时刻轮流使用CPU
- ✓ 交互性: 用户根据系统响应结果进一步提出新请求(用户直接干预每一步)
- ✓ "独占"性: 用户感觉不到计算机为其他人服务
(OS提供虚机器 , 各个用户的虚机器互不干扰)
- ✓ 及时性: 系统对用户提出的请求能及时响应



1.2 操作系统发展过程中形成的一些概念

6. 并发 (Concurrency) 与并行 (Parallel)

- ✓ 并行：两个或多个事件在同一时刻发生。
- ✓ 并发：两个或多个事件在同一时间间隔内发生。

在单处理机系统中，多个程序的并发执行是如何体现的？



1.2 操作系统发展过程中形成的一些概念

7. 多用户 (Multiuser) 与多任务 (Multitask)

- ✓ 多用户：允许多个用户通过各自的终端使用同一台主机，共享主机系统中的各类资源。
- ✓ 多任务：允许多个程序并发执行。



1.2 操作系统发展过程中形成的一些概念

8. 实时OS

- ✓ 是指系统能够实时响应外部事件的请求，在规定的短时间内完成对该事件的处理，并控制所有实时设备和实时任务协调运行。
- ✓ 及时响应：延迟时间短
- ✓ 高可靠性：容错，冗余
- ✓ 一般是专用的，如武器系统的实时控制，生产过程的实时控制等。



1.2 操作系统发展过程中形成的几个概念

9. 网络OS

计算机网络：物理上分散的**自主**计算机通过通信系统的线路**互连**而成。

自主：具有独立处理能力

互连：计算机之间的通信和相互合作。

- ✓ 通信、信息交换、资源共享
- ✓ 互操作、协作

网络OS：提供网络通信和网络服务功能的操作系统。

网络OS的两种基本模式：

- ✓ 客户/服务器（Client/Server）模式
- ✓ 对等（Peer-to-Peer）模式



1.2 操作系统发展过程中形成的一些概念

10. 分布式OS (Distributed OS)

✓ 基础：网络

✓ 分布处理的**透明性**

运行在**不具有共享内存**的多台机器上，但在用户眼里却象一台计算机。

✓ 一个**统一的操作系统**

✓ 逻辑上紧密耦合



1.2 操作系统发展过程中形成的一些概念

网络OS和分布式OS的比较：

✓ 耦合程度

分布式OS是在各机器上统一建立的，统一进行全系统的管理；

网络OS通常容许异种OS互连，各机器上各种服务程序需按不同网络协议互操作。

✓ 并行性

分布式OS可以将一个进程分散在各机器上并行执行，包括进程迁移；

网络OS则各机器上运行的程序是相互独立的。

✓ 透明性

用户是否知道或指定资源在哪个机器上。

分布式OS的网络资源调度对用户透明，用户不了解所占有资源的位置；

网络OS中对网络资源的使用要由用户明确指定。



1.3 OS对运行环境的要求

1. CPU

1) 特权指令 - 多道程序的需要

只能由OS使用。例如，启动外部设备，建立存储保护，清内存、关中断等。

如果没有特权指令的话，会有什么问题？

CPU如何知道是OS还是用户程序在执行呢？

依赖于CPU的状态标识。



CPU

2) CPU的2种工作状态（执行模式）

- ✓ 核心态（Kernel Mode）或称管态
- ✓ 用户态（User Mode）或称目态

核心态和用户态的区别：

处理器处于核心态时：

- ✓ 全部指令（包括特权指令）可以执行
- ✓ 可使用所有资源
- ✓ 并具有改变处理器状态的能力

处理器处于用户态时：

- ✓ 只有非特权指令能执行

特权级别不同，可运行的指令集合也不同

特权级别越高，可以运行的指令集合越大

高特权级别对应的可运行指令集合包含低特权级的



CPU

3) 程序状态字PSW (Program Status Word) 和程序计数器PC (Program Counter)

✓ **PSW** : 指示程序执行的当前状态，主要包括

CPU的工作状态——指明核心态还是用户态，用来说明当前在CPU上执行的是操作系统还是应用程序，从而决定其是否可以使用特权指令或拥有其他的特殊权力

条件标志——反映指令执行后的结果特征

中断标志——指出是否允许中断

✓ **PC** : 指示下一条要执行的指令



1.3 OS对运行环境的要求

2. 内存

是支持OS运行的硬件环境的一个重要方面。

- ✓ 程序必须存放在内存中才能运行。
- ✓ 在多任务系统中，操作系统要管理、保护各任务的程序和数据，使它们不至于受到破坏和相互干扰。
- ✓ 操作系统本身也要存放在内存中并运行，不能被破坏。

内存空间：由若干个存储单元（字节或字）组成的一维连续的地址空间

- ✓ 存储的最小单位：1个二进制位
- ✓ 最小编址单位：字节，一个字节包含8个二进制位



内存

1) 内存分块

块作为分配内存空间的基本单位，如4KB为1块。

为什么要按块来分配内存空间？

旨在简化对内存的分配和管理

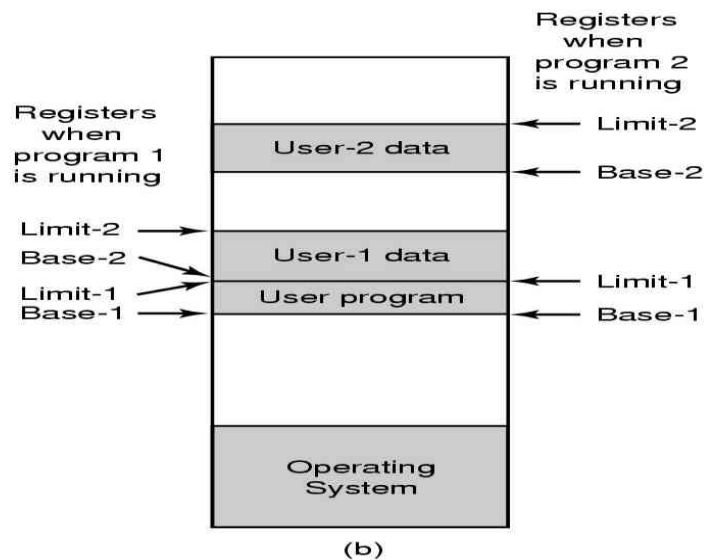
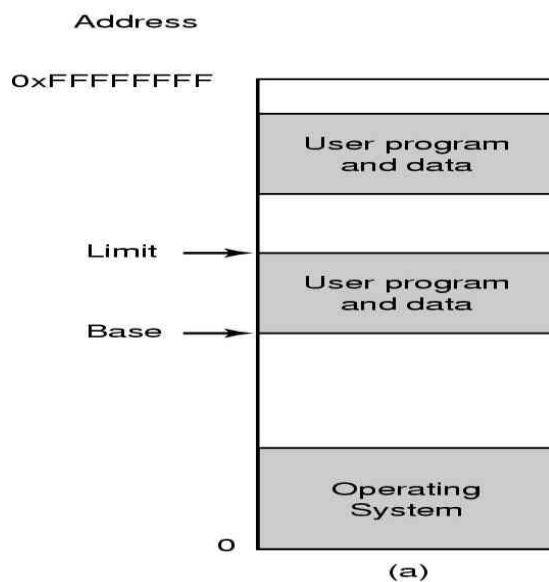
内存

2) 内存保护 - OS正常运行的基本条件

常用的保护机制：

(1) 界限寄存器

存放某任务在内存的上界和下界地址(或者下界与长度)。





内存

界限寄存器实现存储保护的方法：

- ✓ 在CPU中设置一对下界寄存器和上界寄存器
存放用户程序在内存中的下界和上界地址
- ✓ 也可将一个寄存器作为基址寄存器，另一寄存器作为限长寄存器（指示存储区长度）
- ✓ 每当CPU要访问内存，硬件自动将被访问的内存地址与界限寄存器的内容进行比较，以判断是否越界
- ✓ 如果未越界，则按此地址访问内存，否则将产生中断——越界中断（存储保护中断）



内存

(2) 存储保护键 (Key)

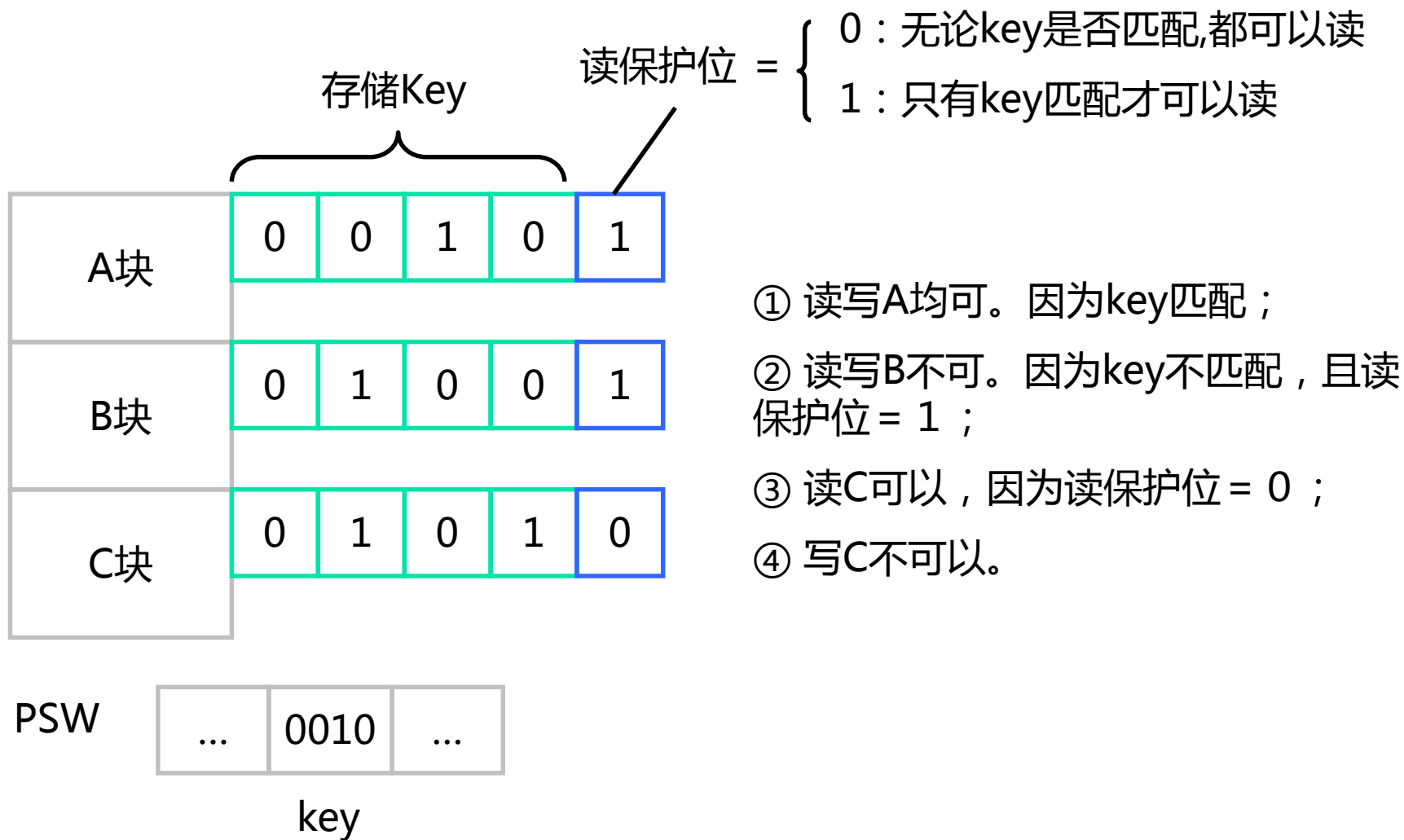
当一个程序进入内存时，OS为其分配一个唯一的Key。

同时将分配给它的每个存储块都设置成该Key。如IBM 370。

该方法的基本要点：

- ✓ 每个运行的程序及其存储块有1个Key；
- ✓ PSW中的存储Key字段存放当前运行程序的Key；
- ✓ 访问内存时，两个Key匹配；
- ✓ 通常将0（在PSW中）作为“万能键”；
- ✓ 存储块引入读保护位：0：Key不匹配时也可读，1：Key不匹配时不可读。

内存





1.3 OS对运行环境的要求

3. 中断 - 如果没有中断，OS将难以工作。

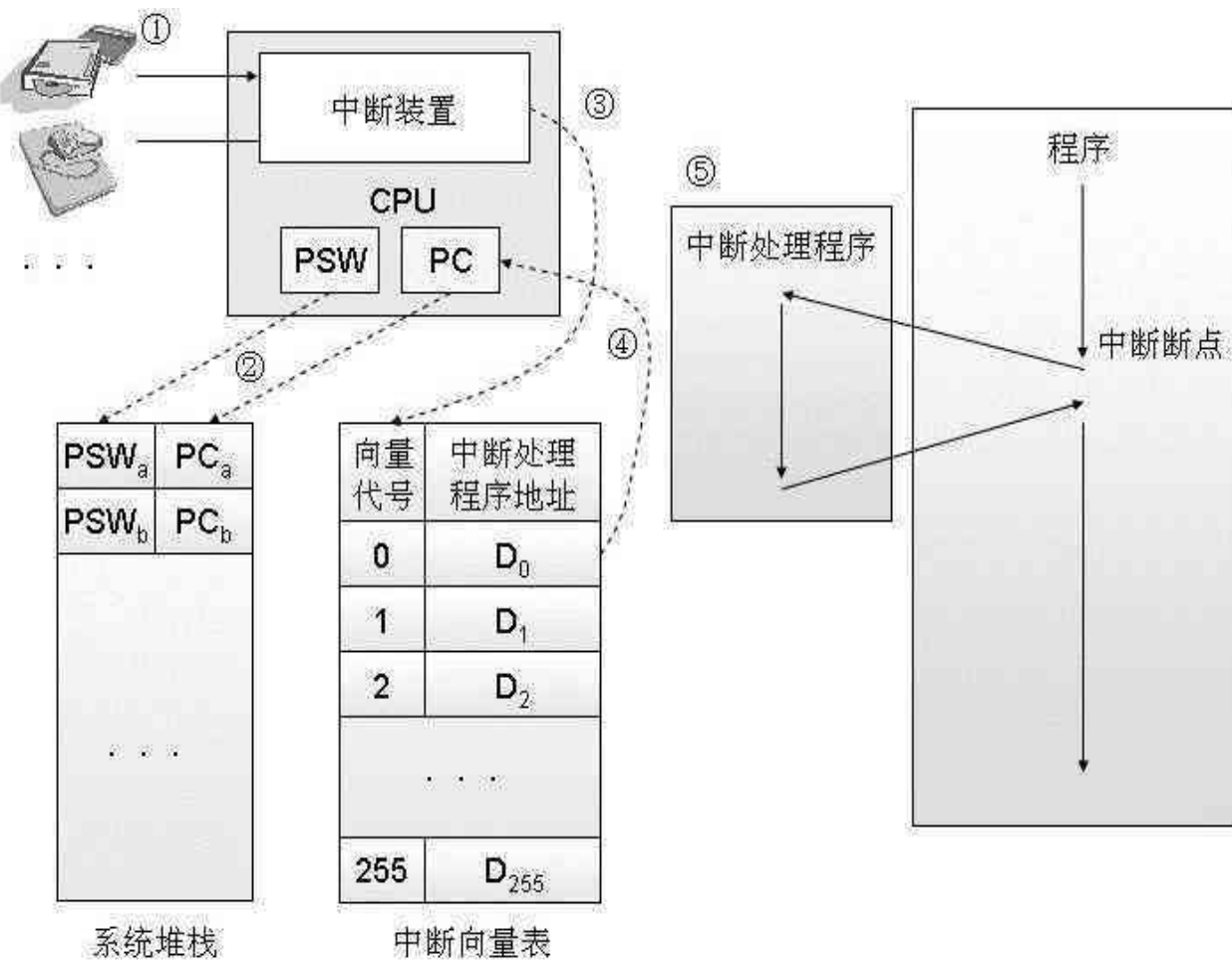
指CPU在收到外部中断信号后，停止原来工作，转去处理该中断事件，完毕后回到原来断点继续工作。

CPU对系统中发生的“异步（随机）”事件的处理

中断的类型：

- ✓ 硬件中断
- ✓ 异常（Exception）
- ✓ 陷入（Trap）- 访管中断（系统调用）

有人称“OS是中断驱动的”。



中断响应



1.3 OS对运行环境的要求

4. 时钟 - OS必不可少的硬件设施

(1) 硬件时钟：通过时钟寄存器实现。

✓ 绝对时钟：记录当前时间

✓ 相对时钟（间隔时钟）：**分时系统的基础。**

(2) 软件时钟：通过时钟队列实现。



1.3 OS对运行环境的要求

5. 重定位

将程序中的相对地址变换为绝对地址。

原因：运行前不可能知道程序将放在内存的什么位置。

- ✓静态重定位：程序装入内存时，由装入程序重定位；
- ✓动态重定位：CPU每次访问内存时，由动态地址变换机构（硬件）自动进行



1.4 典型OS实例

1. Unix

一群计算机迷在贝尔实验室开发出Unix

初衷：可以在一台无人使用的DEC PDP-7 小型计算机上玩星际探险游戏

Ken Thompson , Dennis Ritchie

1983年图灵奖获得者

1999年4月 美国国家技术金奖



1.4 典型OS实例

(1) UNICS(Uniplexed Information and Computing Service)

改名为Unix

(2) 结构：用C、汇编语言写成的

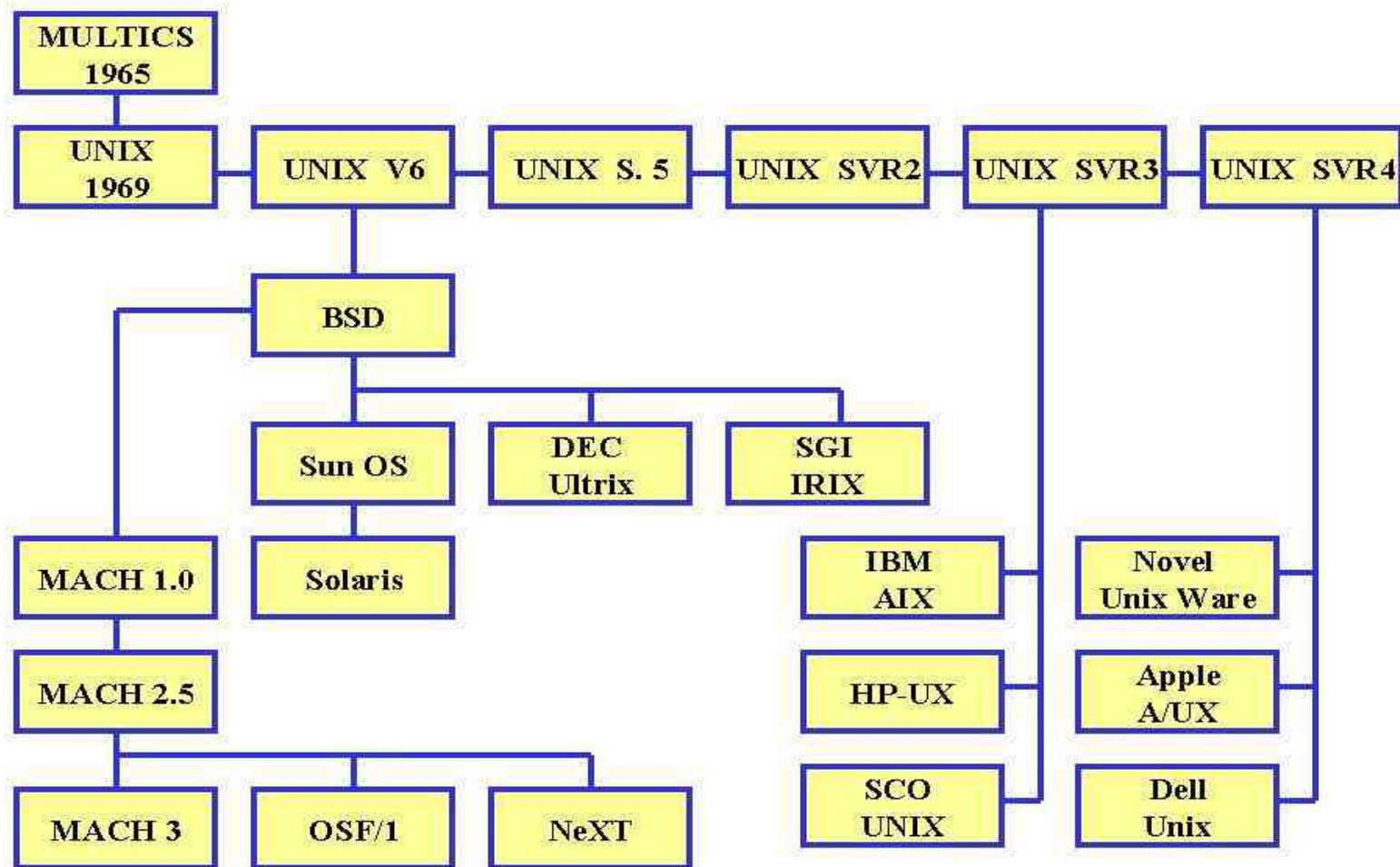
良好的、通用的、多用户、多任务、分时操作系统

(3) 多种变体

两个版本系列

- ✓ AT&T System V
- ✓ BSD (Berkeley Software Distribution)

1.4 典型OS实例





1.4 典型OS实例

2. Linux

1991年，芬兰赫尔辛基大学的一名大学生Linus Benedic Torvalds首先开发

- ✓ 9300行C语言代码，950行汇编语言代码（0.01版）
- ✓ 源代码公开
- ✓ 与Unix兼容
- ✓ 1994年，1.0版：约165000行代码

3. Windows

MS-DOS、Windows 3.1/95/98/Me、Windows NT、Windows 2000/XP、Windows CE、Windows Server 2003，Windows Vista，Windows Server 2008，Windows 7，Windows 8，Windows 10



1.4 典型OS实例

历史上一些重要的操作系统：

- ◆ FMS (FORTRAN Monitor System) 和IBSYS (IBM为7094配备的操作系统)
- ◆ OS/360 (IBM为系列机360配备的操作系统)
- ◆ CTSS (Compatible Time Sharing System)
- ◆ MULTICS (MULTiplexed Information and Computer Service)
- ◆ UNIX类、Linux
- ◆ CP/M
- ◆ MS-DOS , Windows 3.1/95/98/Me , Windows NT , Windows 2000/XP , Windows CE , Windows Server 2003 , Windows Vista , Windows Server 2008 , Windows 7
- ◆ Macintosh
- ◆ OS/390
- ◆ Mach
- ◆ VxWorks



1.5 现代操作系统的基本特征

- (1) 并发
- (2) 共享
- (3) 虚拟
- (4) 不确定



1.6 从不同角度认识操作系统

(1) 软件

外在特性：使用方式（命令，系统调用）

内在特性：结构，功能

(2) 资源管理器

底层资源共享，提高资源利用率

(3) 虚拟机

方便用户使用计算机

(4) 标准服务提供者

提供每个用户需要的标准工具

如标准库、窗口系统



1.6 从不同角度认识操作系统

(5) 仲裁者 (协调者)

使多个应用程序 (用户) 高效、公平地一起工作

保护应用程序之间不互相干扰

(6) 幻觉制造者 (illusionist)

提供硬件的高层接口, 程序员 (用户) 感觉不到硬件限制

操作系统提供 “无限大” 的内存、 “无限多” 的CPU



1.7 学习操作系统课程要达到的目标

1. 为什么要学习操作系统？

- ✓ 操作系统涉及到计算机科学的很多领域：计算机体系结构，软件设计，数据结构，算法等
- ✓ 设计操作系统或者修改现有的系统
如嵌入式系统(Embedded OS)
- ✓ 开发应用程序必须与操作系统打交道，特别是并发程序
- ✓ 开发应用程序时借鉴操作系统的设计思想和算法，特别是复杂软件系统设计的思维方法
- ✓ 操作系统的许多思想和方法可以应用到其他领域



1.7 学习操作系统课程要达到的目标

2. 学习操作系统要达到的目标

1) 理解OS的基本概念、工作原理、典型实现技术以及OS设计中考虑的各种因素并能合理运用

- 从设计OS的角度去思考、学习
- 理解（**非简单的记忆**）概念、原理的实质：是什么？为什么？有什么用？如何实现？适用于什么场合？如信号量。
- 领会其中的重要思想：如工作集，设备独立性，SPOOLing技术
- **建立OS的整体概念**
- 系统观与方法学



1.8 学习操作系统课程要达到的目标

2) 设计、实现具体的操作系统；或剪裁操作系统（如嵌入式OS）

- 能独立对小型操作系统的部分功能进行源代码分析、设计和实现
- 设计一个功能强大的实用的OS 是非常不容易的
- 需要的理论、技术涉及计算机体系结构、软件设计方法学、软件管理、数据结构与算法、网络等相关知识
- 需要考虑硬件、应用程序、将来可能的变化
概念的抽象，接口设计



1.8 学习操作系统课程要达到的目标

3) 应用软件（特别是大型软件）开发

- OS作为一个复杂软件系统，涉及到分析解决复杂工程问题所需的许多重要思想和方法
- 并发软件开发：同步，死锁
- 借鉴OS 设计中的问题及其解决方法、思想
 - 如：权衡（tradeoff）：时间与空间；性能与方便使用；通用性与效率
 - 抽象、虚拟、实证、分层
- 思想、方法在不同领域往往是相通的



1.8 学习操作系统课程要达到的目标

3. 对OS的整体理解

从设计OS（OS设计者）的角度去学习、思考

OS是软件：

- 资源管理器：管理的对象包括硬件、软件
- 用户接口

功能：

- 管理CPU、内存、外存、I/O设备
- 管理程序、文件
- 提供用户使用OS的接口：程序级（系统调用）、命令级

4大功能：进程管理、内存管理、文件管理、I/O 设备管理

整个操作系统课程就是围绕这4大功能展开的



1.8 学习操作系统课程要达到的目标

性能：

- 效率：时间；空间
- 方便使用：屏蔽细节；统一接口
- 提供用户使用OS的接口：程序级（系统调用）、命令级
- 扩展性：如新设备的加入
- 可靠性：容错
- 安全性：访问控制



1.8 学习操作系统课程要达到的目标

应用环境：

- 是否多处理机
- 是否多任务
- 存储空间限制
- 响应时间要求

总体目标：

批处理/分时/实时，是否多任务、多用户，通用/专用



进一步学习的参考书：

1) 操作系统设计与实现

- [1] Andrew S. Tanenbaum . 陈渝等译 . 操作系统设计与实现 (第三版) . 电子工业出版社 , 2007
- [2] 于渊 . Orange S : 一个操作系统的实现 . 电子工业出版社 , 2009
- [3] Scott Maxwell . 冯锐等译 . Linux内核源代码分析 . 机械工业出版社 , 2000

2) 系统程序设计

- [4] Kay A. Robbins , Steven Robbins . 陈涓等译 . Unix系统编程 . 机械工业出版社 , 2005
- [5] W. Richard Stevens , Stephen A. Rago . 戚正伟 , 张亚英 , 尤晋元译 . Unix环境高级编程 (第3版) . 人民邮电出版社 , 2014
- [6] Jeffrey Richter . 葛子昂等译 . Windows核心编程 (第5版) . 清华大学出版社 , 2008

3) 分布式操作系统

- [7] Andrew S. Tanenbaum . 陆丽娜等译 . 分布式操作系统 . 电子工业出版社 , 2008