

《操作系统》



主讲教师：翟高寿

联系电话：010-51684177 (办)

电子邮件：gszhai@bjtu.edu.cn

制作人：翟高寿

制作单位：北京交通大学计算机学院

教学目标

全面、系统地介绍计算机操作系统的体系结构、实现机理及相关方法和技术，培养广大学生在系统软件开发方面的理论基础及技术素养。

主要相关课程

先修课程：

数据结构

接口技术、计算机组成原理

汇编语言/ C语言程序设计

并行/后续课程：

计算机体系结构

专业实习与实训、本科毕业设计

高级操作系统、安全操作系统

教材

计算机操作系统 汤小丹等 西安电子科技大学出版社

参考资料

操作系统原理讲义 翟高寿 何永忠 黄华 等 北京交通大学

操作系统实用教程：螺旋方法 翟高寿（译） 机械工业出版社

操作系统原理与实践 邹鹏(第3&6章：翟高寿) 高等教育出版社

系统程式(2006版&2007版) 翟高寿 台湾新文京出版公司

操作系统教程 孙仲秀 高等教育出版社

Windows操作系统原理 尤晋元史美林陈向群 机械工业出版社

The Design of The Unix Operating System Manrice J. Bach 人民邮电出版社

计算机操作系统教程 张尧学 史美林 清华大学出版社

操作系统教程 陈向群 杨芙清 北京大学出版社

操作系统基础 屠立德 屠祁 清华大学出版社

操作系统教程-原理和实例分析 孟静 高等教育出版社

Applied Operating System Concept Abrahan Silberschatz 高等教育出版社

现代操作系统 陈向群等译 机械工业出版社

课程主要内容及教学安排

- ❑ 操作系统引论（6学时）
 - ❑ 处理机管理（14学时）
 - ❑ 存储管理（10学时）
 - ❑ 设备管理（8学时）
 - ❑ 文件系统（8学时）
 - ❑ 操作系统设计实例（2学时）
 - ❑ 实验16学时（计算机专业实验室）
- 课程总结
（1学时）

课程作业及考核说明3-1

- 1、课程总评成绩由平时成绩和期末考试成绩两部分组成，前者占40%，后者占60%；
- 2、平时成绩根据同学参与中国大学慕课课程《操作系统-北京交通大学》的课程成绩（涵盖理论作业暨单元测试、实验作业暨单元作业和综合测评），并参考书面作业（标★者）成绩最终确定；
- 3、要求以自己的交大邮箱在中国大学慕课网<http://www.icourse163.org>注册，并选择参加慕课课程《操作系统-北京交通大学》（课程负责人：翟高寿），或者直接通过网址参加：<https://www.icourse163.org/course/NJTU-1003245001>



操作系统

分享

第3次开课

开课时间: 2021年02月26日 ~ 2021年06月30日 进行至第1周, 共18周

学时安排: 3-5小时每周

已有 1083 人参加

立即参加

课程详情

课程评价(99)

操作系统是计算机科学与技术专业的核心课程。我们开设的操作系统课程被评为北京市优质本科课程, 主要面向计算机科学与技术专业学生, 但其中大部分内容也适用于软件工程、信息安全、人工智能等计算机类其它专业的必修性操作系统课程, 不少内容并适用于电子等理工类专业的选修性操作系统课程以及从事计算机系统研发和高级应用开发人员的需要。

—— 课程团队



北京交通大学
BEIJING JIAOTONG UNIVERSITY

5 位授课老师



霍高寿
副教授

课程作业及考核说明3-2

4、中国大学慕课课程《操作系统-北京交通大学》已于2021年2月26日正式开课（并将于**2021年6月30日前结课**），相关作业提交时间窗口也会相继开放，请各位同学尽快完成注册加入事宜；

5、中国大学慕课课程《操作系统-北京交通大学》成绩由理论习题作业（单元测试）成绩35%、实验课题报告（单元作业）成绩30%、期末测评成绩30%及慕课平台学习表现（专门用于奖励那些积极参与慕课课程讨论、质疑和答疑的同学）5%四部分组成；

6、理论习题作业（**截止时间均为2021年6月18日**）采用**单元测试和客观题模式**，由计算机自动评阅产生成绩，**多次测试以最高分为准**；

课程作业及考核说明3-3

7、实验课题作业采用单元作业方式（提交截止时间一般为作业发布两周之内），每位同学可选做**6项**实验课题，且成绩评定采用学生互评为主、教师评定为辅的方式（互评截止时间一般为提交截止时间后一周之内）；

8、每选做和完成一项实验课题，均要求评价与自己选做实验课题相同的五份作业，若做不到会在成绩生成时有惩罚措施，详参中国大学慕课课程《操作系统-北京大学》成绩评价标准及作业互评相关页面。

第一章 操作系统引论

1.1 什么是操作系统

1.2 计算机体系结构及基础

1.3 操作系统用户接口及实现

1.4 操作系统启动模块及自装入控制

1.5 操作系统的发展

1.6 操作系统的功能与

1.7 操作系统的结构设计



1.1 什么是操作系统

1.1.1 操作系统的地位和目标

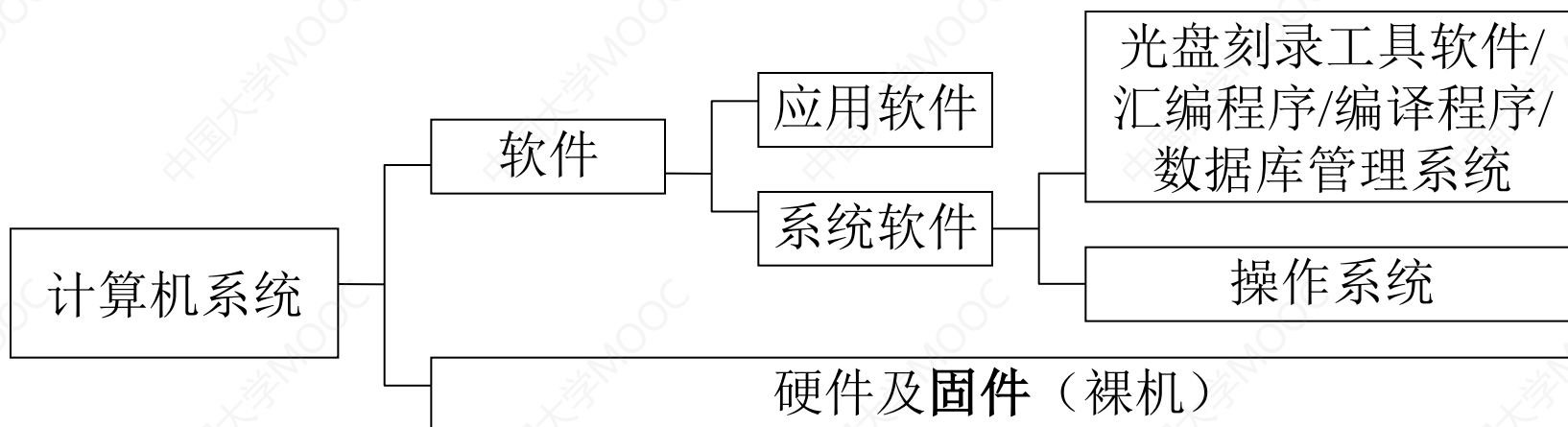
1.1.2 操作系统的作用

1.1.3 操作系统的组成及层次模型

1.1.4 操作系统的定义

1.1.5 操作系统举例

计算机系统的组成



操作系统在计算机系统中的地位



操作系统的设计目标

❑ 方便性

- 提供用户接口，使计算机系统更方便使用

❑ 有效性

- 通过有效管理和分配软、硬件资源及合理组织计算机工作流程来改善资源利用率、提高系统吞吐量

❑ 可扩充性

- 适应计算机硬件和体系结构的迅猛发展及其所对应的更高的功能和性能要求

❑ 开放性

- 适应不同厂家与不同类型的计算机及其设备的网络化集成和协同工作，实现应用程序可移植性和互操作性

1.1 什么是操作系统

1.1.1 操作系统的地位和目标

1.1.2 操作系统的作用

1.1.3 操作系统的组成及层次模型

1.1.4 操作系统的定义

1.1.5 操作系统举例

操作系统的作用(1)

用户与计算机硬件系统之间的接口

❑ 命令方式（操作系统外壳）

- 面向一般用户
- 命令行/菜单式/命令脚本式/图形用户接口

❑ 系统调用方式（操作系统内核）

- 面向程序开发人员
- 形式上类似于过程调用，编制程序中使用

操作系统的作用 (2)

计算机系统资源的管理者

□ 管理对象

- 处理器、存储器、外围设备以及信息（数据和软件）

□ 管理内容

- 资源的分配、回收和访问操作
- 记录资源的当前状态（数量和使用情况）、相应管理策略（共享、保护及用户权限）

操作系统的作用(3)

用作扩充机器（或虚拟机）

- 在裸机上添加处理机管理、存储管理、设备管理、文件管理、作业管理、图形化用户接口等功能，使计算机系统功能显著增强、使用更为方便

1.1 什么是操作系统

1.1.1 操作系统的地位和目标

1.1.2 操作系统的作用

1.1.3 操作系统的组成及层次模型

1.1.4 操作系统的定义

1.1.5 操作系统举例

操作系统的组成及层次模型

用户接口

（命令接口、程序接口、图形用户接口）

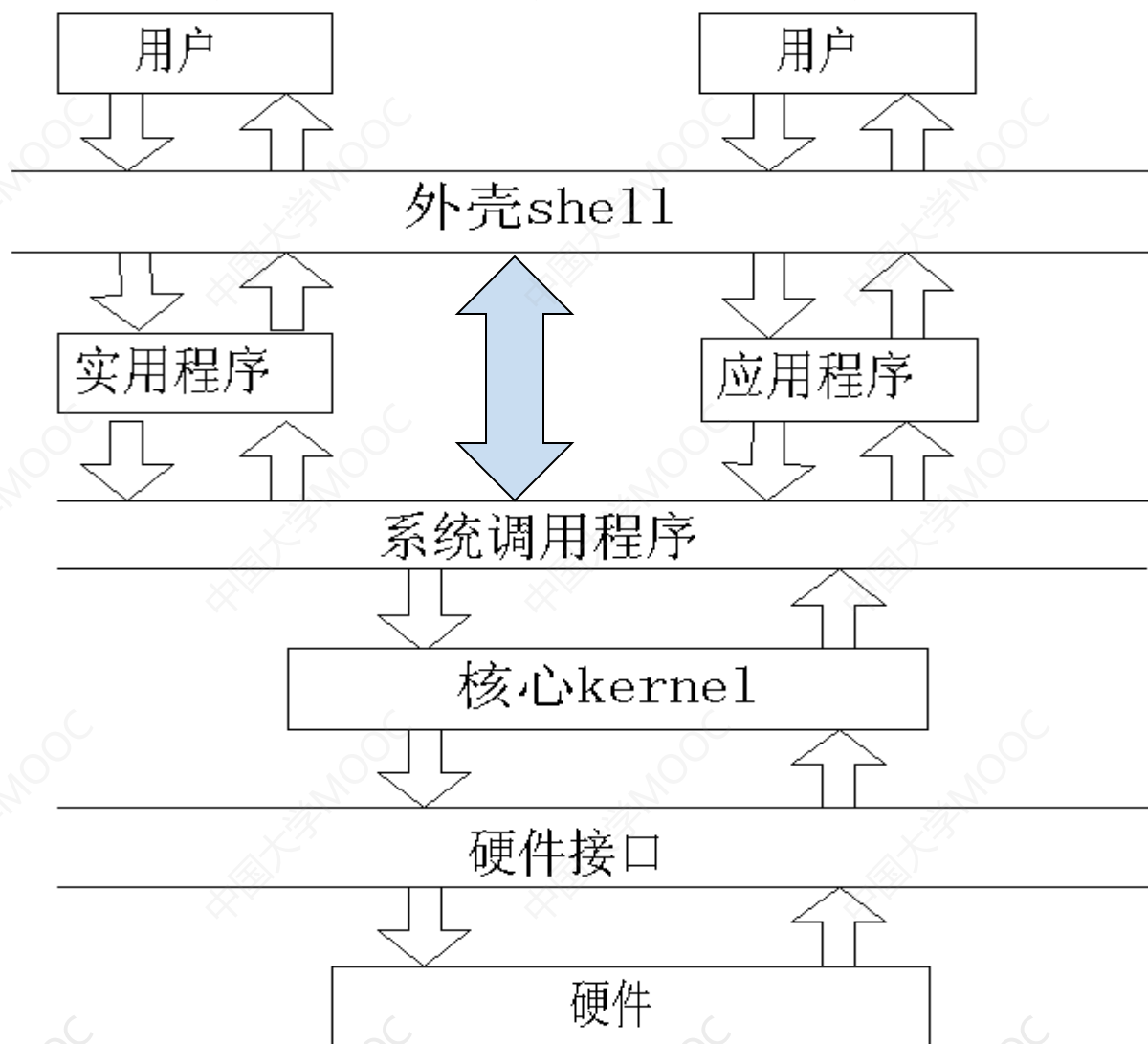
对对象进行操纵和管理的软件集合

（处理机/存储器/设备/文件/作业管理软件）

操作系统对象

（处理机、存储器、设备、文件和作业）

操作系统层次模型细化



1.1 什么是操作系统

1.1.1 操作系统的地位和目标

1.1.2 操作系统的作用

1.1.3 操作系统的组成及层次模型

1.1.4 操作系统的定义

1.1.5 操作系统举例

操作系统的定义

- ❑ 操作系统是最基本的系统软件，是一组有效管理和控制计算机硬件和软件资源、合理地对各类作业进行调度以组织和控制系统工作流程，并方便用户使用计算机的程序的集合。

1.1 什么是操作系统

1.1.1 操作系统的地位和目标

1.1.2 操作系统的作用

1.1.3 操作系统的组成及层次模型

1.1.4 操作系统的定义

1.1.5 操作系统举例

操作系统举例

- ❑ MS DOS
- ❑ Windows (NT、9X、200X、Vista)
- ❑ UNIX (Solaris)
- ❑ Linux (RedHat、Ubuntu、RedFlag、麒麟)
- ❑ Novell Netware
- ❑ VxWorks、**SylixOS** (翼辉公司)
- ❑ AIX
- ❑ Android、iOS

1.1 什么是操作系统

1.1.1 操作系统的地位和目标

1.1.2 操作系统的作用

1.1.3 操作系统的组成及层次模型

1.1.4 操作系统的定义

1.1.5 操作系统举例

作业题

- 1.1 什么是操作系统？用自己的话谈谈你对操作系统概念的认识与理解。

第一章 操作系统引论

1.1 什么是操作系统

1.2 计算机体系结构及操作系统硬件基础

1.3 操作系统用户接口及系统调用实现

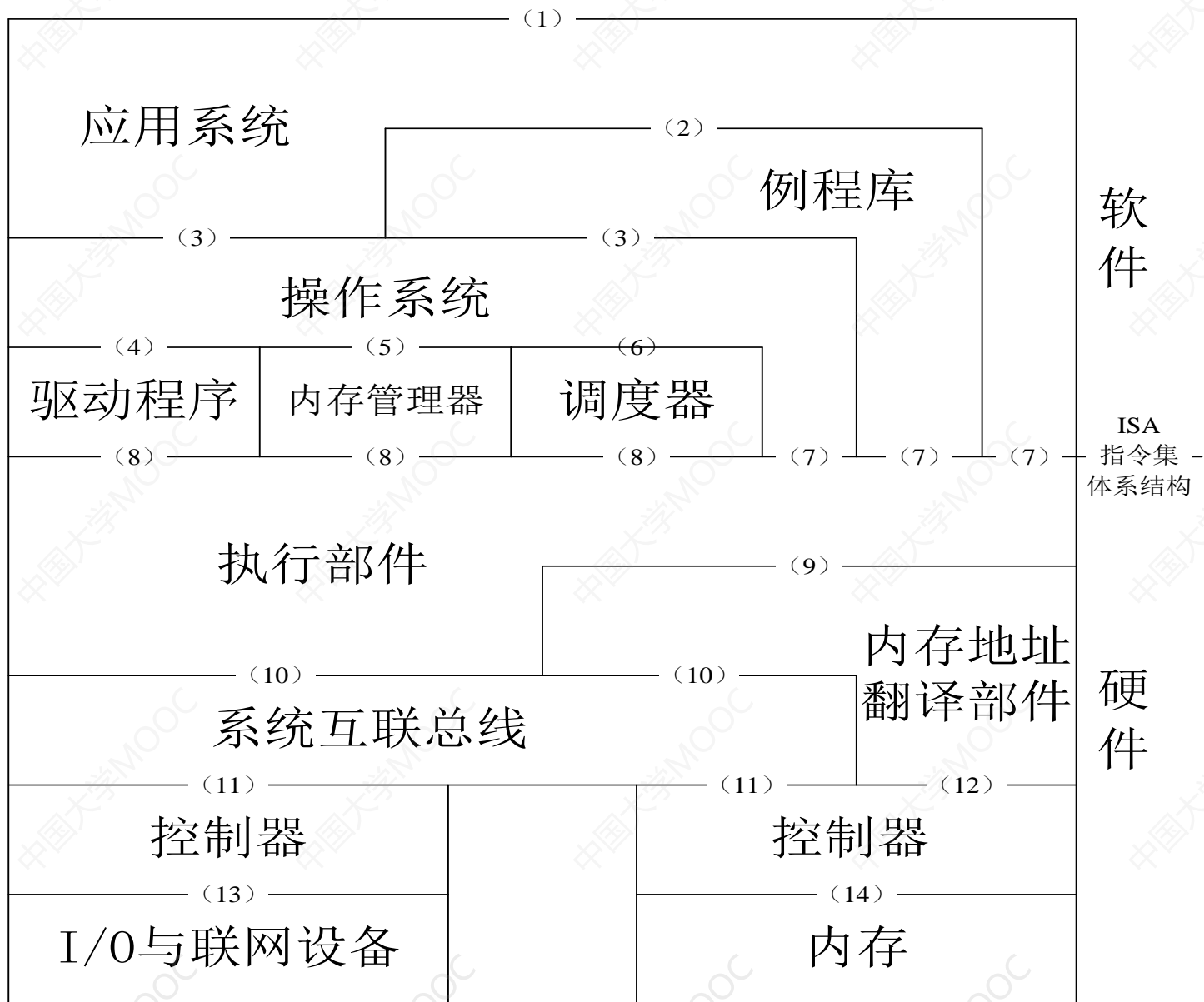
1.4 操作系统启动模块及自装入机制

1.5 操作系统的发展

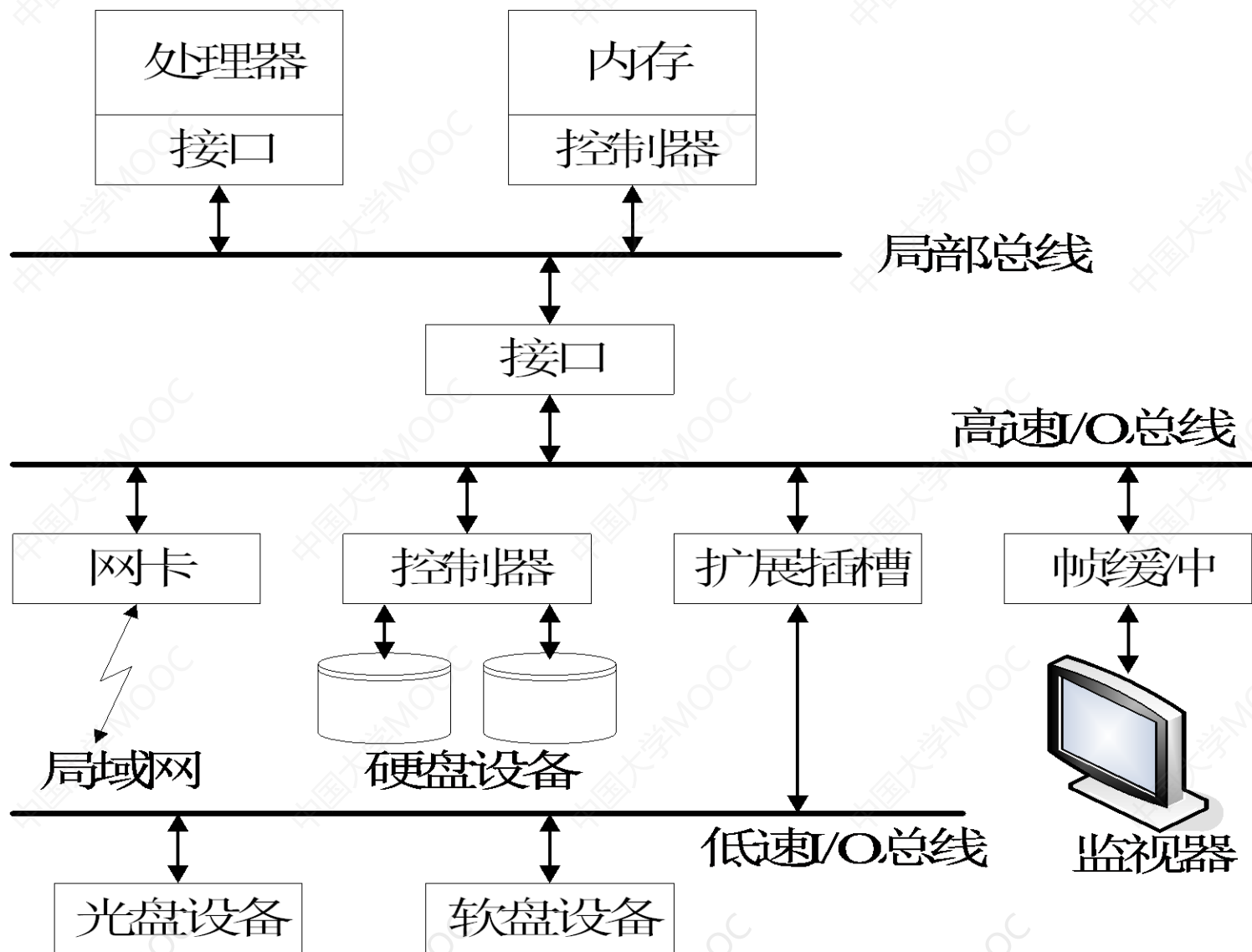
1.6 操作系统的功能与特征

1.7 操作系统的结构设计

计算机系统体系结构



计算机系统硬件组成



内存系统Memory Systems

□ 主存Main Memory

➤ RAM & ROM

➤ 实际地址空间**real address space**

A. 或称作物理地址空间，按字节编址Bytes

B. RAM&ROM

C. 保留用于设备或以后它用

□ 高速缓冲Cache Memories

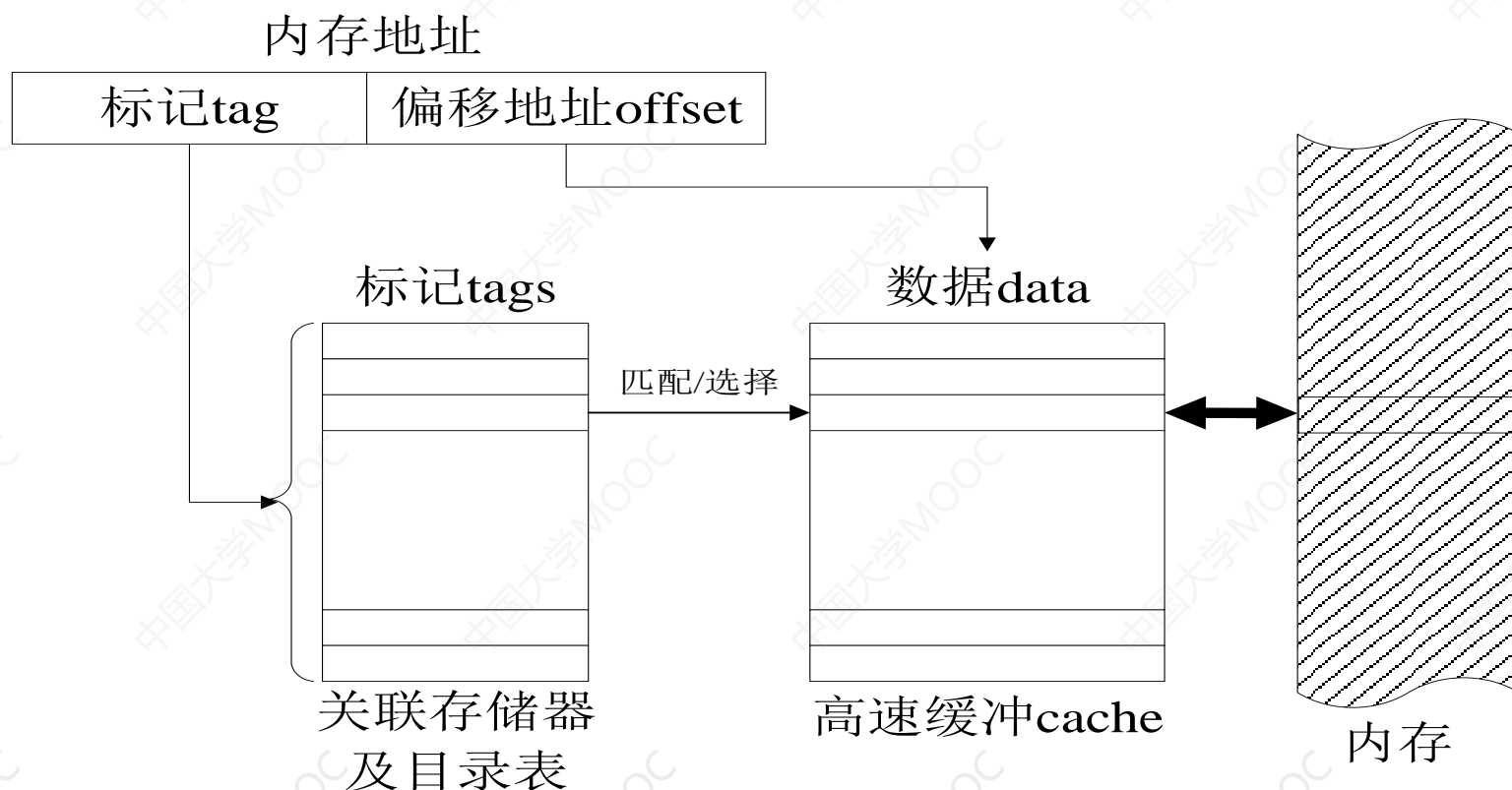
➤ 因内存寻址访问效率和**局部性原理**而引入

➤ 由硬件管理，对软件不可见

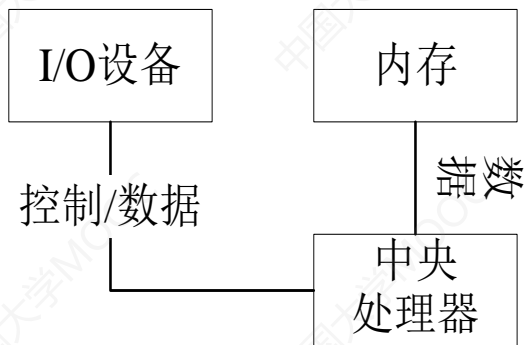


高速缓存Cache Memories

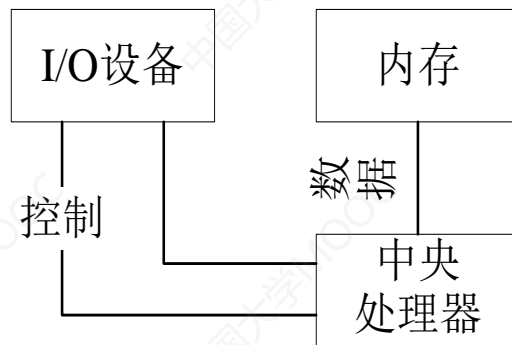
❑ 关联存储器及置换（淘汰）算法



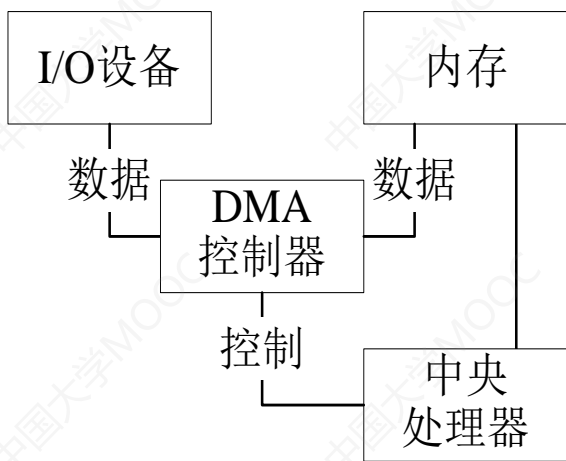
输入输出系统组织方式



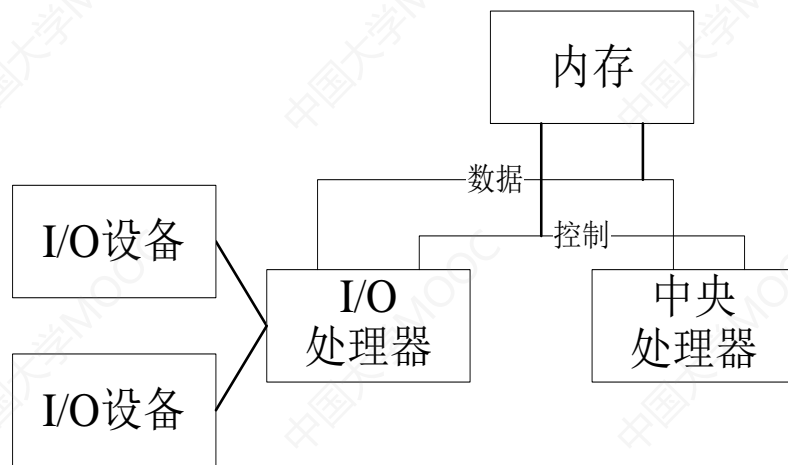
(a)



(b)



(c)



(d)

用户指令集体系结构 User ISA

□ 寄存器架构

- 通用寄存器、特定类型寄存器（如浮点数）
- 专用寄存器（PC、链接/栈指针/条件码/循环计数寄存器）

□ 内存架构

➤ 逻辑/虚拟地址空间

A. 线性编址 eg. $0x0000\ 0000 \leftrightarrow 0x8000\ 0000 \leftrightarrow 0xFFFF\ FFFF$

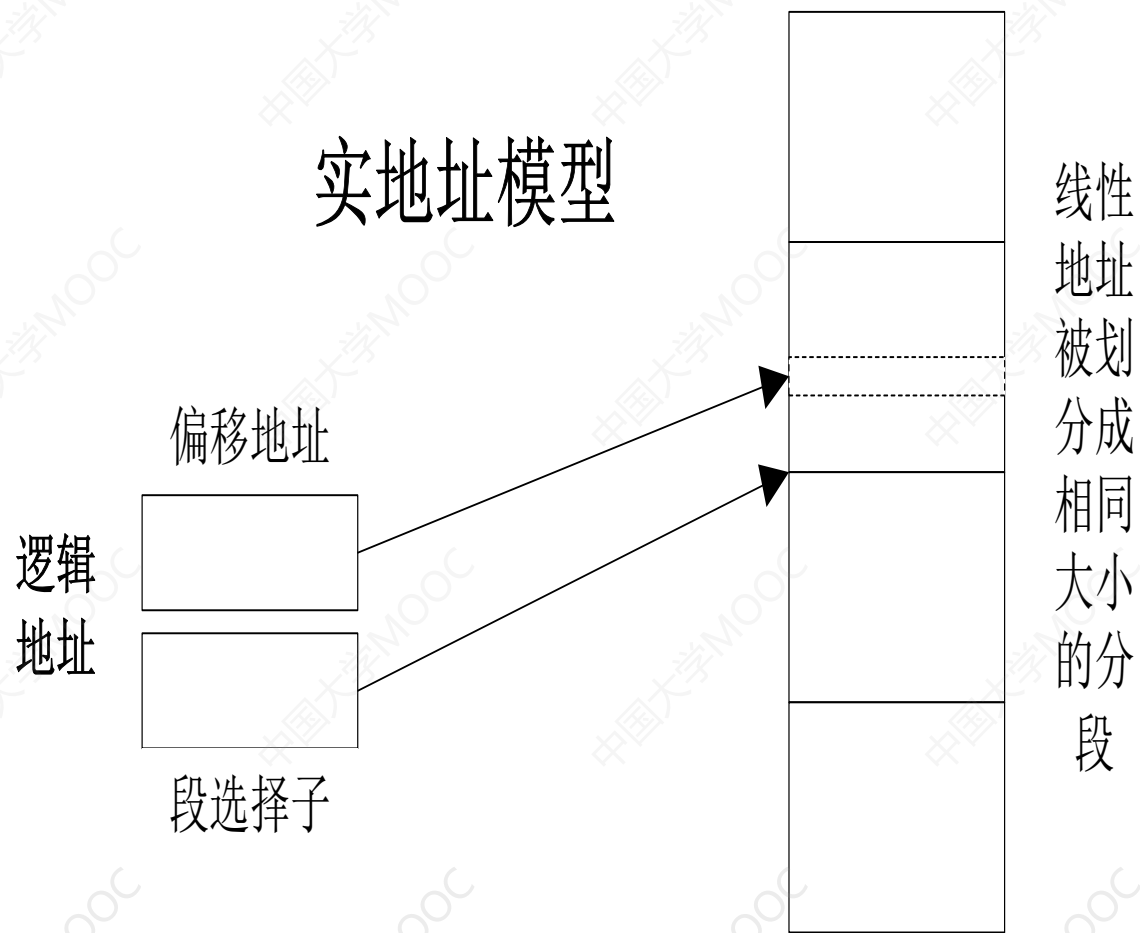
B. 分段编址 eg. $0\#\sim 15\#\text{分段基址} \leftrightarrow \text{段寄存器值}$

□ 用户指令 —— 运算型

- 内存存取指令、分支跳转指令
- 整数算术、逻辑及移位指令
- 浮点指令

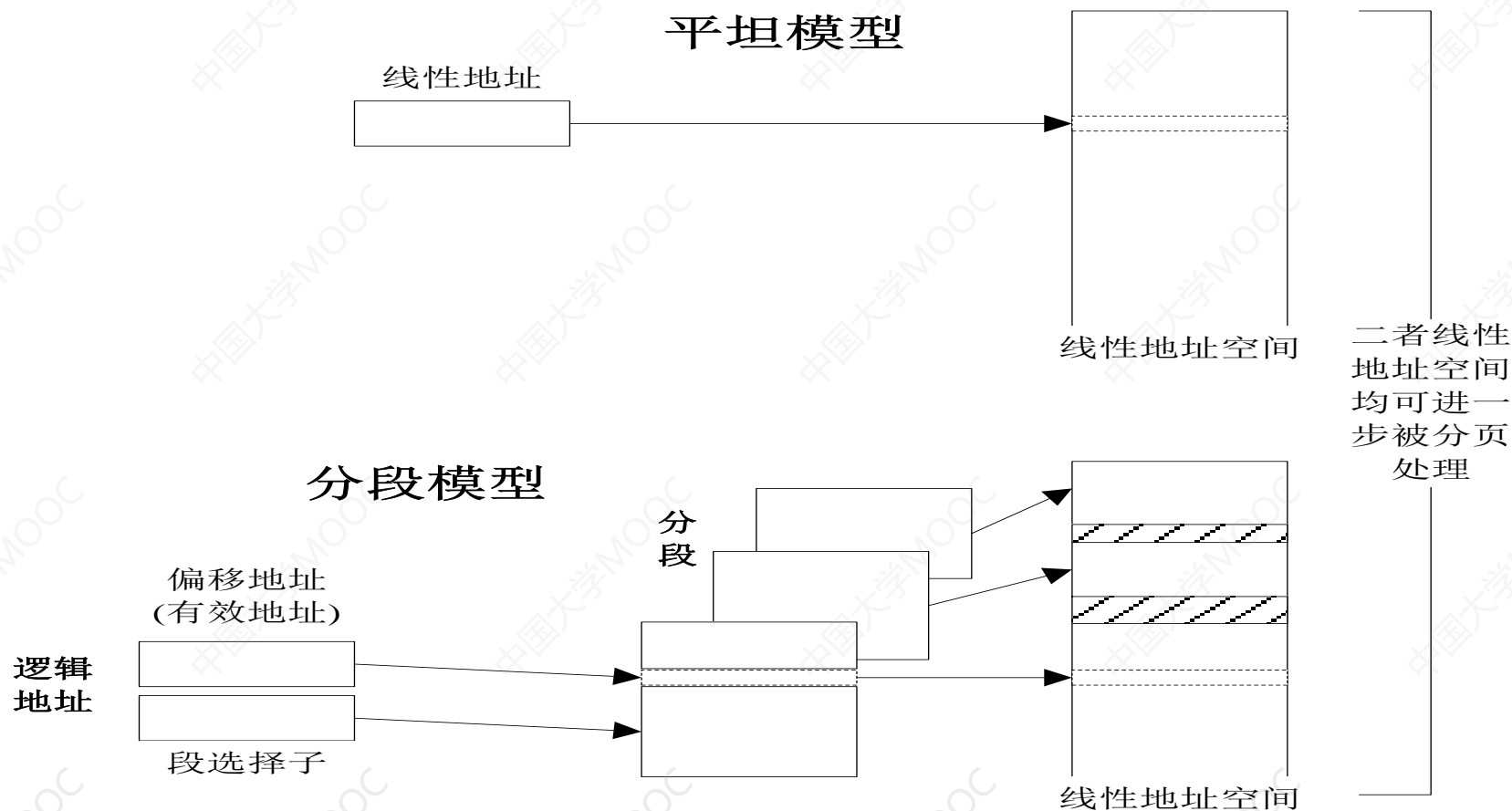
内存管理模型

实地址模型



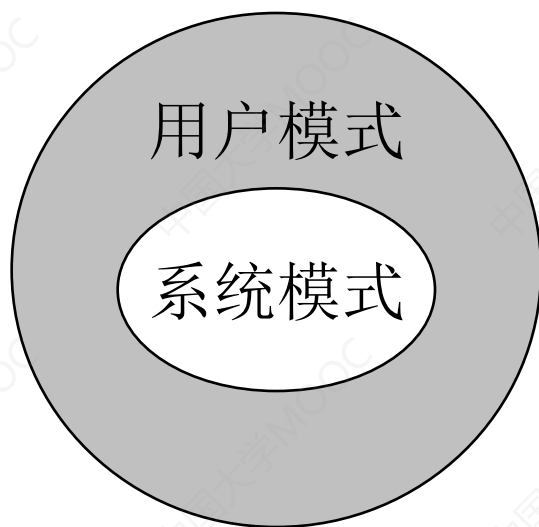
内存管理模型

平坦模型/分段模型

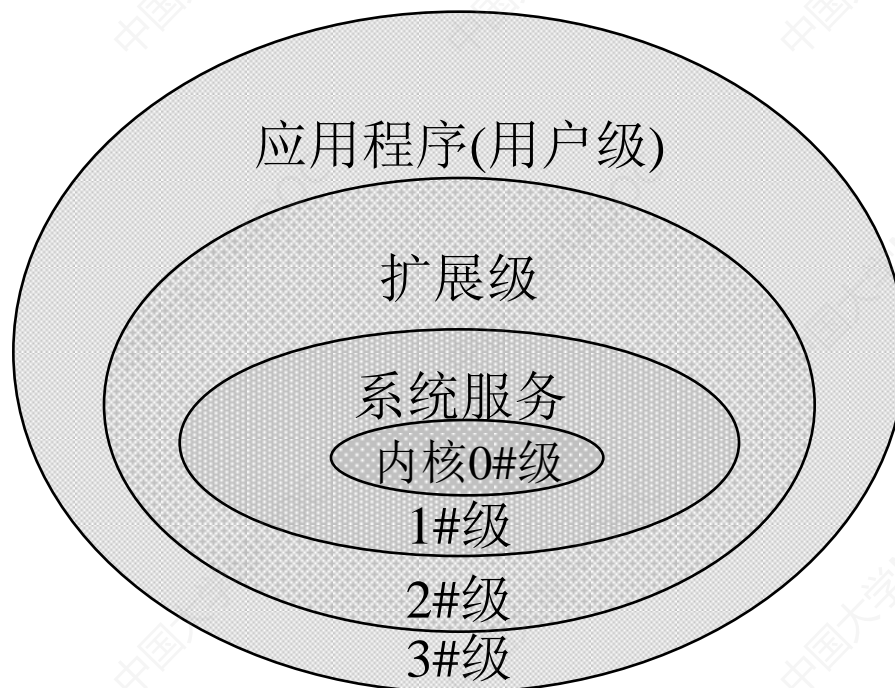


特权级别与环结构

□ 用户模式、特权/超级/系统模式



(a)



(b)

系统指令集体系结构 System ISA

□ 系统寄存器架构

- 系统时钟寄存器、陷入与中断寄存器
- 陷入与中断屏蔽寄存器、页/段表指针寄存器

□ 处理器资源管理支持

- 系统调用/返回指令、中断定时器及设置机制

□ 内存资源管理支持

- 虚拟地址空间到物理地址空间的映射（页表&TLB）

□ I/O资源管理支持

- I/O设备寻址与I/O指令

□ 陷入与中断

陷入与中断

❑ 陷入

- 指令执行时因异常情况（如运算溢出、缺页、违规内存访问、非法操作码等）产生的控制迁移副效应
- 陷入流程（异常→设置陷入寄存器→查看陷入屏蔽寄存器裁决→终止“陷入”指令并置处理器准确状态→保存程序计数及各寄存器值→处理器置特权模式并交操作系统控制权→操作系统保存未被硬件保存的“陷入”进程的关键状态信息，经分析转特定异常陷入处理例程，待完成返回后恢复现场和转原“陷入”进程“异常”指令处继续执行）
- 另“系统调用”式陷入（访管指令int 21H/int 0x80）

❑ 中断

- 由相对于当前执行进程的外部事件（如I/O操作、定时）引起，与特定指令执行无关

流程？

IA-32体系结构 举例说明

□ 系统寄存器架构

- 标志寄存器EFLAGS（通用标志、系统标志）
- 内存管理寄存器GDTR、IDTR、LDTR、TR
- 控制寄存器CR0、CR1、CR2、CR3

□ 保护模式内存管理

- 内存寻址（分段寻址技术）
- 地址变换（分段[段选择符→段描述符]+分页）
- 内存保护（全局/局部描述符表、特权级）

□ 中断和异常处理

- 中断向量与中断描述符表、中断源与异常源

□ 任务管理（任务状态段）

任务切换与过程调用不同！

操作系统其它硬件基础 举例说明

- ❑ 内存地址空间布局、基本输入输出系统BIOS
- ❑ CMOS存储器与实时时钟RT
- ❑ I/O端口寻址和访问控制方式
 - 主要使用独立编址方式，部分使用统一编址方式
- ❑ 中断控制器Intel 8259A、定时计数器Intel 8254
- ❑ DMA控制器Intel 8237A、键盘控制器Intel 8042
- ❑ 串行控制卡RS-232标准
- ❑ 显示卡MDA/CGA/EGA/VGA
- ❑ 软盘控制器和硬盘控制器

作业题

- 1.2 设想由你自己负责组织一个项目团队来构建操作系统，你应当要求项目成员事先学习和掌握哪些硬件基础知识？并给出你对相关知识的理解与总结。

第一章 操作系统引论

1.1 什么是操作系统

1.2 计算机体系结构及操作系统硬件基础

1.3 操作系统用户接口及系统调用实现

1.4 操作系统启动模块及自装入机制

1.5 操作系统的发展

1.6 操作系统的功能与特征

1.7 操作系统的结构设计

1.3 操作系统用户接口 及系统调用实现

1.3.1 操作系统接口分类

1.3.2 联机命令接口

1.3.3 图形用户接口

1.3.4 系统调用

操作系统接口分类

□ 基于接口表现形式划分

➤ 用户接口

A. 命令接口（具体可分为联/脱机命令接口）

B. 图形用户接口（图形化操作界面）

➤ 程序接口（方便用户程序访问系统资源，由一组系统调用组成）

□ 基于接口使用者类型划分

➤ 本地用户接口

➤ 远程用户接口（网络用户接口）

1.3 操作系统用户接口 及系统调用实现

1.3.1 操作系统接口分类

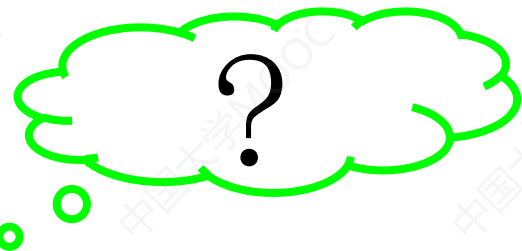
1.3.2 联机命令接口

1.3.3 图形用户接口

1.3.4 系统调用

联机命令接口构成

- 一组联机命令
- 键盘终端处理程序
- 命令解释程序



联机命令及其格式与分类

□ 联机命令格式

<命令> [<可选项>] <参数序列>

□ 联机命令类型

- 系统访问类 (
- 磁盘操作类、
- 网络通信类
- 输入输出重定向、管道连接、过滤命令
- ◆ 批处理方式 (批处理文件/脚本文件)

```
ls
```

```
ls -l >file.txt
```

```
ls ps -A | grep ssh
```


键盘终端处理程序

键盘扫描码到
ASCII的转换

□ 基本功能

I. 接收用户从终端输入的字符

➤ 面向字符/行方式

II. 管理字符缓冲，以暂存所接收的字符

➤ 专用缓冲区、公用缓冲池方式

III. 将用户键入字符回送屏幕显示

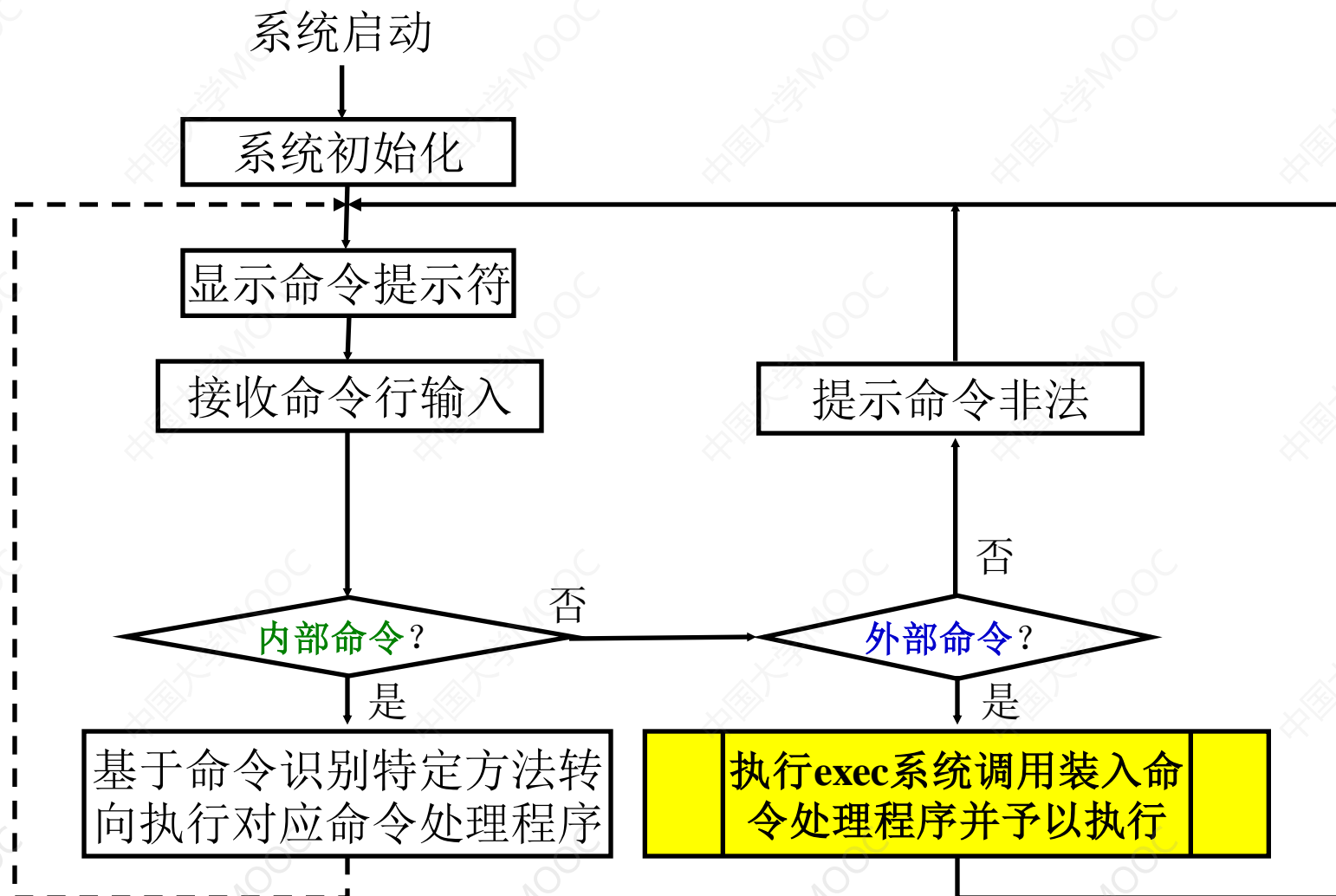
➤ 硬件/软件实现方式

◆ 提供屏幕编辑（编辑键）

◆ 特殊字符处理（中断/停止或恢复上卷）

优劣
比较

命令解释程序工作流程



1.3 操作系统用户接口 及系统调用实现

1.3.1 操作系统接口分类

1.3.2 联机命令接口

1.3.3 图形用户接口

1.3.4 系统调用

图形用户接口元素及操作

- ❑ 桌面、图标、鼠标指针
- ❑ 窗口、标题栏、菜单栏、工具栏
- ❑ 菜单
 - 菜单条
 - 弹出式菜单
 - 下拉式菜单
- ❑ 对话框

图形用户接口特点

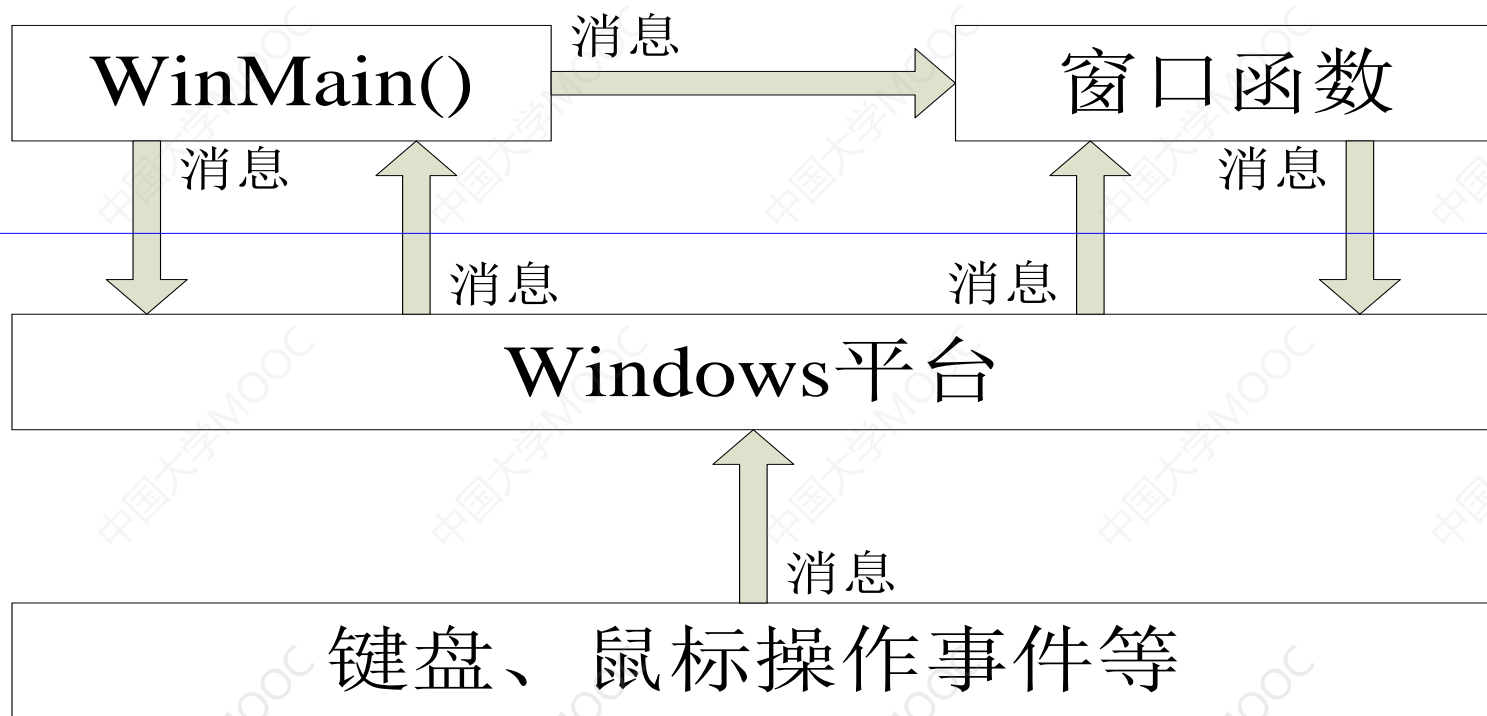
- ❑ 基于图形元素来表示功能，方便用户操纵和触发对应功能
- ❑ 同屏多窗口与并发进程相对应
- ❑ 支持即时交互，鼠标点击和键盘输入并举
- ❑ 操作直观，不必死记命令行参数，传递信息量大

图形用户接口实现及运行机制

- 面向对象程序设计方法
- 消息产生、传递及处理
 - 消息作为窗口的输入，如用户操作、其它窗口或系统发出的请求或通知
 - 消息由各窗口自己的窗口过程进行处理
- 事件驱动模式

Windows系统运行机理

Windows应用程序



1.3 操作系统用户接口 及系统调用实现

1.3.1 操作系统接口分类

1.3.2 联机命令接口

1.3.3 图形用户接口

1.3.4 系统调用

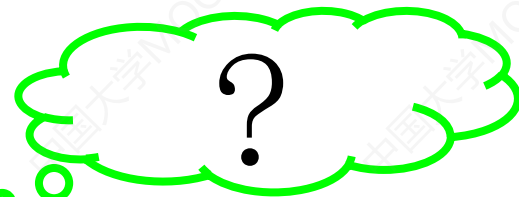
系统调用的基本概念

□ 定义

❖ 系统过程→系统服务→系统调用命令

□ 与普通过程调用的区别

- 运行在不同的系统状态
- 软中断进入机制
- 返回及重新调度问题
- 嵌套调用



系统调用的类型

□ 进程控制

- 进程的创建、结束、等待子进程结束
- 进程属性设置与获取
- 执行一个文件（进程映像替换）

□ 文件操纵

- 文件的创建、打开、关闭、读/写

□ 进程通信

- 连接打开与关闭、消息发送与接收

□ 系统信息维护

- 时间设置与获取、文件访问/修改时间设置

系统调用举例说明

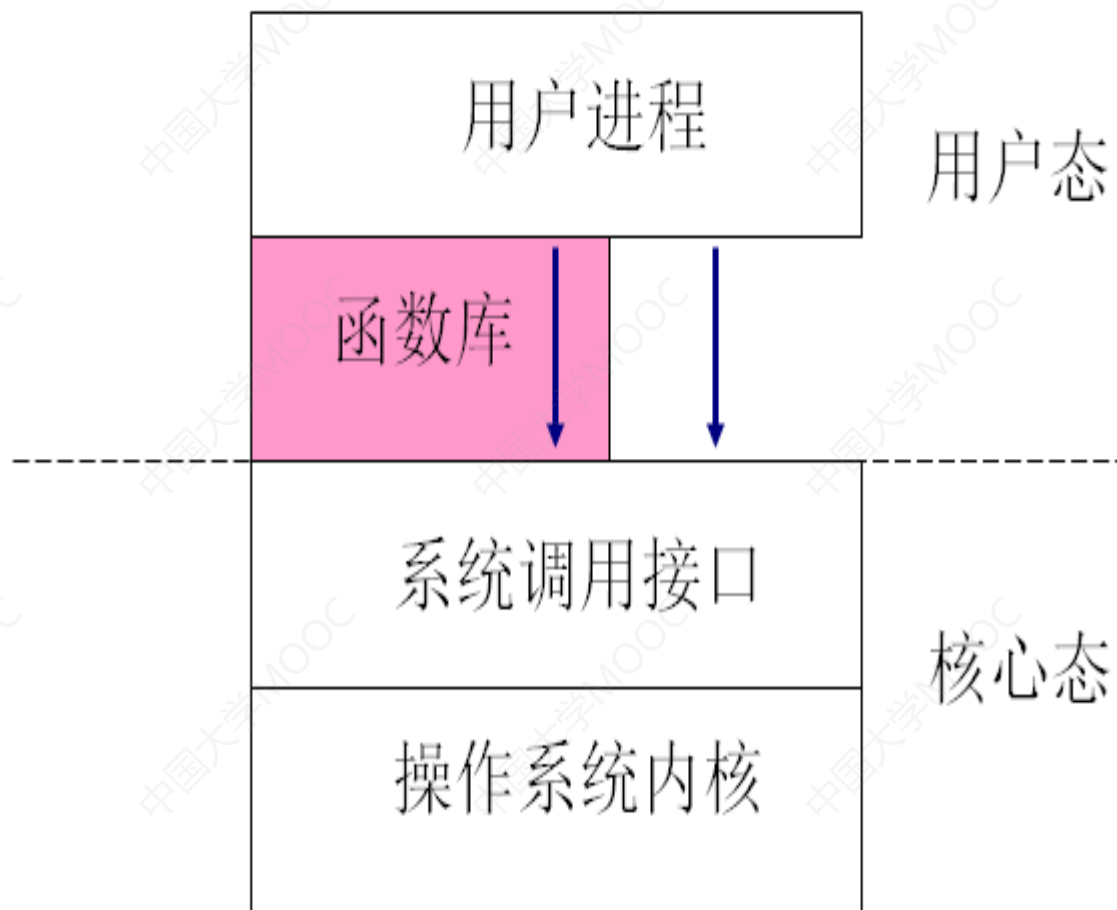
□ 简单用户程序例子

- ❖ 从一个文件读取数据，再将它们拷贝到另一文件中（两个文件名已获得）

□ 系统调用分析


- 源数据文件打开
- 目标数据文件创建
- 文件数据读入到缓冲
- 缓冲数据写出到文件
- 关闭数据文件

系统调用使用层次图示



通过库函数使用系统调用举例说明

```
#include <fcntl.h>
#define O_RDONLY 00
#define O_RDWR 0666
int main(int argc, char *argv[])
{
    int fdOld, fdNew, count;
    char buffer[2048];
    fdOld = open(argv[1], O_RDONLY);
    fdNew = creat(argv[2], O_RDWR);
    while (count = read(fdOld, buffer,
        sizeof(buffer)))
        write(fdNew, buffer, count);
    close(fdOld); close(fdNew);
}
```



直接使用系统调用举例说明

```
#include <unistd.h>
#include <sys/syscall.h>
#define O_RDONLY 00
#define O_RDWR 0666
int main(int argc, char *argv[])
{
    int fdOld, fdNew, count;
    char buffer[2048];
    fdOld = syscall(SYS_open, argv[1], O_RDONLY);
    fdNew = syscall(SYS_creat, argv[2], 0666);
    while (count = syscall(SYS_read, fdOld, buffer,
        sizeof(buffer)))
        syscall(SYS_write, fdNew, buffer, count);
    syscall(SYS_close, fdOld); syscall(SYS_close, fdNew);
}
```

/usr/include/unistd.h

/usr/include/i386-linux-gnu/bits/syscall.h

直接进行系统调用的前提准备1

/usr/include/i386-linux-gnu/sys/syscall.h:

(1) #include <asm/unistd.h> ⇔ /usr/include/i386-linux-gnu/...

#include <asm/unistd_32.h> ⇔ /usr/include/i386-linux-gnu/...

#include <asm/unistd_64.h> ⇔ /usr/include/i386-linux-gnu/...

#define __NR_read 3 //0

#define __NR_write 4 //1

#define __NR_open 5 //2

#define __NR_close 6 //3

#define __NR_creat 8 //85

(2) #include <bits/syscall.h> ⇔ /usr/include/i386-linux-gnu/...

#define SYS_close __NR_close

#define SYS_creat __NR_creat

#define SYS_open __NR_open

#define SYS_read __NR_read

#define SYS_write __NR_write

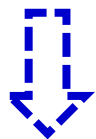


Linux-4.8.8

直接进行系统调用的前提准备2

/usr/include/unistd.h:

```
extern long int syscall(long int __sysno, ...) __THROW;
```

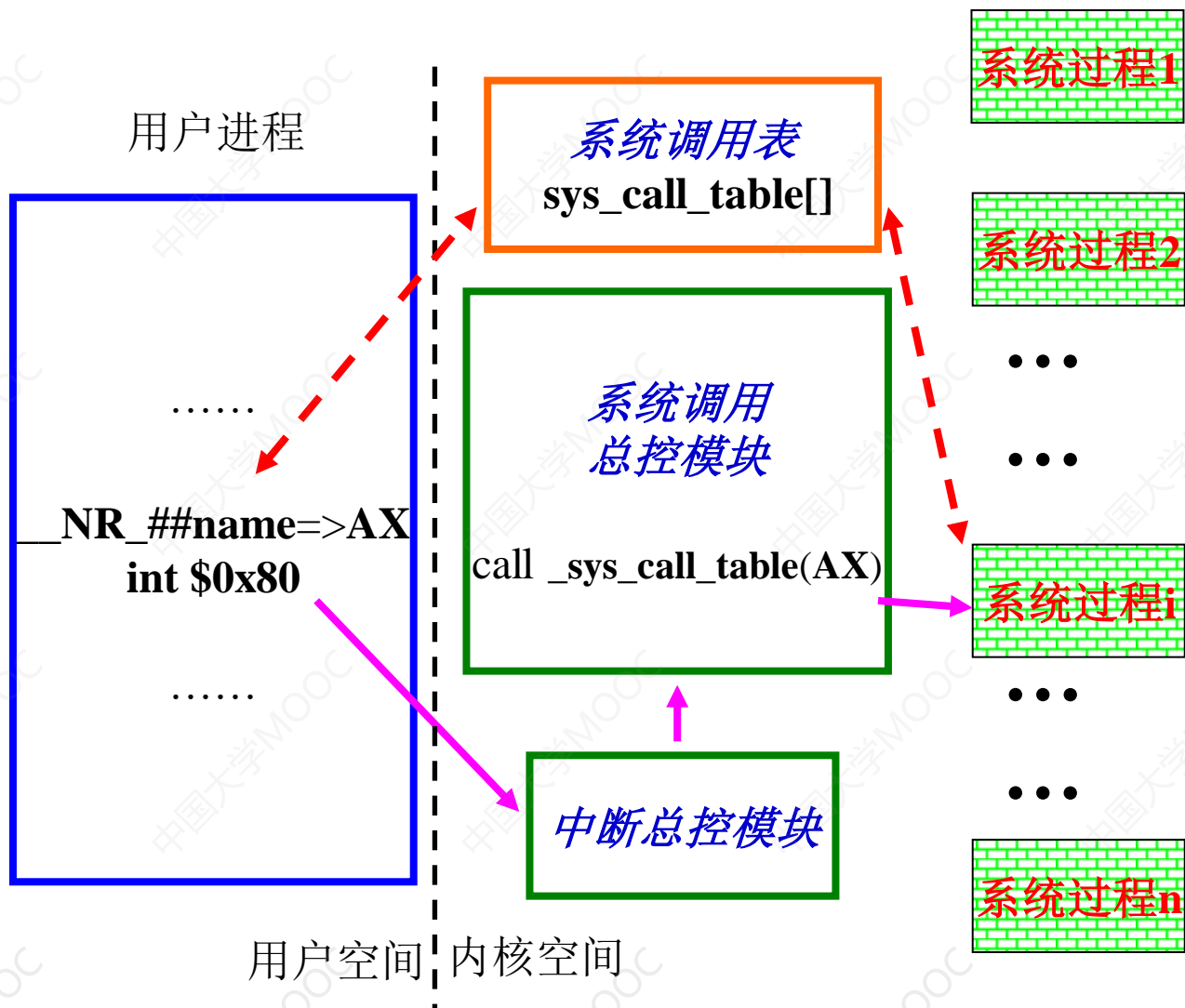


```
_syscall2(long, open, const char*, filename, int, flags);  
_syscall2(long, creat, const char*, filename, int, flags);  
_syscall3(ssize_t, read, unsigned, fd, char*, buf, size_t, count);  
_syscall3(ssize_t, write, unsigned, fd, char*, buf, size_t, count);  
_syscall1(void, close, unsigned, fd);
```

系统调用号与系统调用之间的联系建立在**系统调用表sys_call_table**中，本质上利用函数指针数组实现，譬如：

```
FUNPTRTYPE sys_call_table[] = {sys_read, sys_write,  
sys_open, sys_close, ..., sys_creat, ...};
```


系统调用实现机制



混合编程要领

C调用汇编
模块呢?

□ 汇编语言源程序调用C函数模块

- 按逆向顺序把C函数模块参数压入栈中
- `call _CModuleName| jmp _CModuleName`
- 清除先前压入栈中的C函数模块参数

□ C语言程序嵌入汇编指令举例说明

```
#define _syscall1(type, name, atype, a) \  
type name(atype a) \  
{ \  
    long _res; \  
    __asm__ volatile("int $0x80" \  
                      : "=a" (_res) \  
                      : "0" (__NR_##name), "b" ((long)(a))); \  
    if (_res >= 0) return (type)_res; \  
    errno = -_res; return -1; \  
}
```

系统调用的实现要领

I. 设置系统调用号和参数

- 系统调用号（指定寄存器/内存单元）
- 参数（直接[寄存器]、间接[参数表指针]）
- UNIX(CHMK命令)/DOS(INT21软中断)

II. 系统调用命令的一般性处理

- 将处理机状态由用户态转为系统态
- 保护CPU现场，将PSW、PC、系统调用号、**用户栈**指针、通用寄存器等压入堆栈
- 用户定义参数送至指定位置

III. 分析系统调用类型，转相应处理子程序

- **中断和陷入向量表**（入口地址、PSW）

1.3 操作系统用户接口 及系统调用实现

1.3.1 操作系统接口分类

1.3.2 联机命令接口

1.3.3 图形用户接口

1.3.4 系统调用

作业题

- ❑ **1.3** 什么是输入输出重定向？什么是管道联接？分别加以举例说明。
- ❑ **1.4** 试阐述程序接口与用户交互接口（即命令接口和图形化接口）之间的关系？并给出你对系统调用实现机制及处理过程的完整理解与总结。

实验课题

□ 实验课题1

Linux命令解释程序设计与实现

□ 实验课题4

Linux系统调用设计与添加实现

第一章 操作系统引论

1.1 什么是操作系统

1.2 计算机体系结构及操作系统硬件基础

1.3 操作系统用户接口及系统调用实现

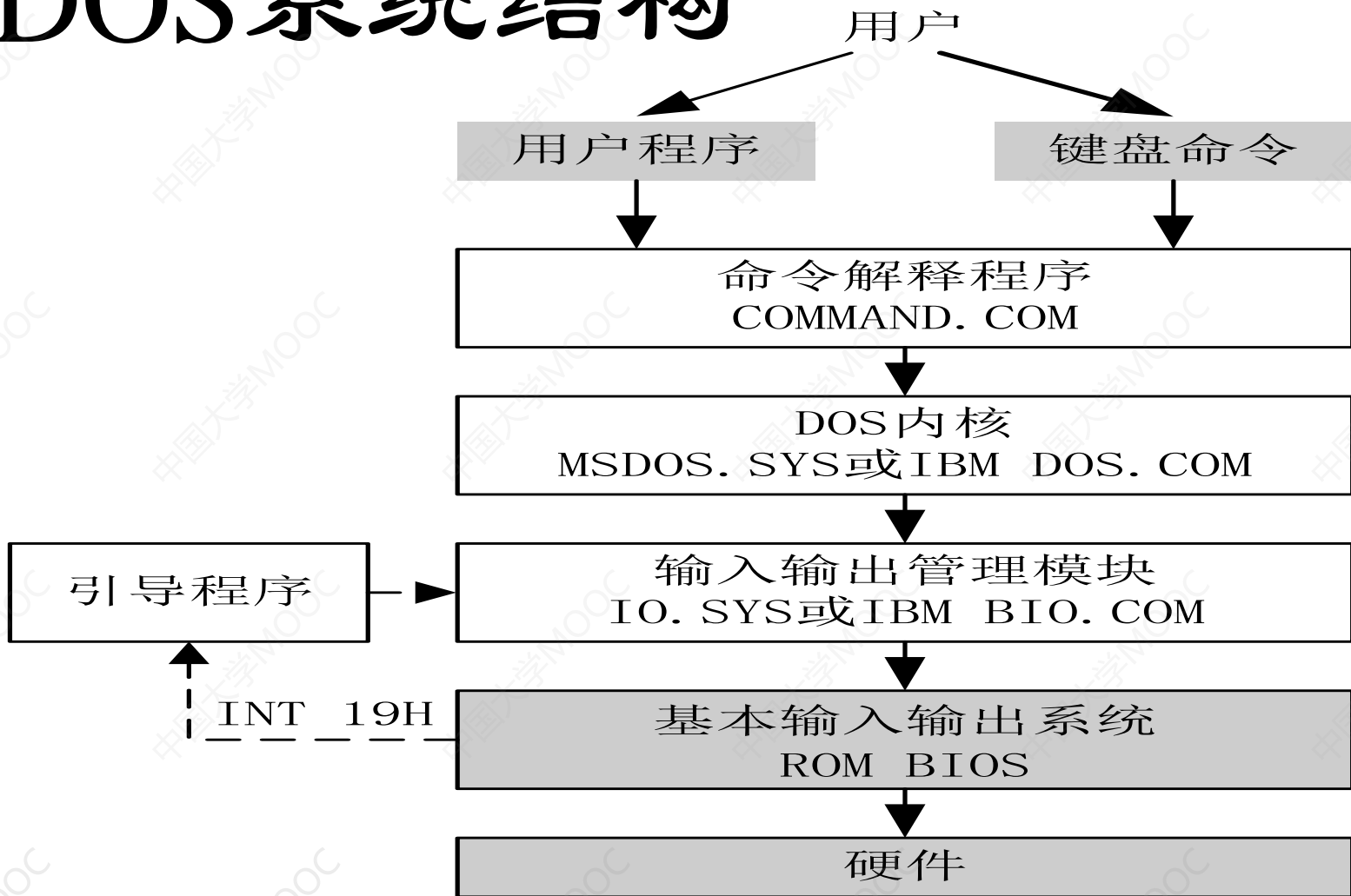
1.4 操作系统启动模块及自装入机制

1.5 操作系统的发展

1.6 操作系统的功能与特征

1.7 操作系统的结构设计

DOS系统结构



引导扇区与系统

0FFFF0

+ 0000

0FFFF0

□ 系统启动过程

- 机器加电 => BIOS加电自检(0FFFF: 0000)进行硬件检测 => 读入启动盘引导扇区(0:0:1)512字节到内存 **0000: 7C00**处 => 检查内存0000: 7DFE~7DFF是否为0x55AA => 跳转至0000: 7C00执行引导记录程序

□ 引导扇区

- 末两个字节为0x55AA

引导程序
载入位置

□ 引导程序编制基础

- 开发工具NASM或GCC
- 引导代码必须编译成plain binary file类型且为512B

最简单的引导程序

❑ 汇编程序代码MinBoot.asm

```
hang: jmp hang
```

```
times 510 - ($ - $$) db 0
```

```
dw 0xAA55
```

❑ 程序汇编

➤ `nasm MinBoot.asm -o MinBoot.bin`

❑ 将二进制文件MinBoot.bin内容拷贝到引导扇区

二次装载引导程序设计

□ 二次装载含义

1. 由BIOS将引导记录载入内存执行
2. 由引导记录将操作系统内核载入内存

□ 软驱复位与读取

- BIOS之INT 13H功能调用

□ 字符串显示

- BIOS之INT 10H功能调用

系统启动时内存布局

0xFFFFFFFF

0xFFFFF

内核

0xF0000



0x10000

0x7C00

0x04F0

0x0400

0x0000

BIOS代码区

BIOS数据区

中断向量区

二次装载引导程序之引导程序设计

[ORG 0]

jmp 07C0h: start

start:

mov ax, cs

mov ds, ax

mov es, ax

reset:

mov ax, 0

mov dl, 0

int 13h

jc reset

read:

mov ax, 1000h

mov es, ax

mov bx, 0

mov ah, 2

mov al, 5

mov ch, 0

mov cl, 2

mov dh, 0

mov dl, 0

int 13h

jc read

软驱0:0:2起
五个扇区读
入到ES:BX

jmp 1000h:0000

times 510-(\$-\$\$) db 0
dw 0xAA55

二次装载引导程序之内核程序设计

```
[ORG 0]
    jmp start2

msg db "Kernel!", $0

start2:
    mov ax, cs
    mov ds, ax
    mov es, ax
```

```
    mov si, msg
print:
    lodsb ...
    cmp al, 0
    je hangup
```

DS:SI => AL

```
    mov ah, 0Eh
    mov bx, 7
    int 10h
    jmp print
hangup:
    jmp hangup
```

实验课题

□ 实验课题2

最简操作系统设计与实现

□ 实验课题3

Linux启动初始化过程探析

第一章 操作系统引论

1.1 什么是操作系统

1.2 计算机体系结构及操作系统硬件基础

1.3 操作系统用户接口及系统调用实现

1.4 操作系统启动模块及自装入机制

1.5 操作系统的发展

1.6 操作系统的功能与特征

1.7 操作系统的结构设计

1.5 操作系统的发展

1.5.1 推动操作系统发展的主要动力

1.5.2 无操作系统时的计算机系统

1.5.3 单道批处理系统

1.5.4 多道批处理系统

1.5.5 分时系统

1.5.6 实时系统

1.5.7 操作系统的进一步发展

推动操作系统发展的主要动力

- ❑ 不断提高计算机资源利用率和系统性能的需要
- ❑ 改善和方便用户使用计算机的需要
- ❑ 适应器件不断更新换代的需要
- ❑ 适应计算机体系结构不断发展变化的需要

1.5 操作系统的发展

1.5.1 推动操作系统发展的主要动力

1.5.2 无操作系统时的计算机系统

1.5.3 单道批处理系统

1.5.4 多道批处理系统

1.5.5 分时系统

1.5.6 实时系统

1.5.7 操作系统的进一步发展

早期计算机人工操作方式

- ❑ 1946 ~ 50年代中期
- ❑ 计算机资源昂贵、集中计算
- ❑ 工作方式
 - 用户：同时兼有程序员/操作员双重身份
 - 输入输出：纸带或卡片
 - 编程语言：机器语言
- ❑ 工作特点
 - 用户独占全部资源，资源利用率低
 - 计算前后，CPU因等待人工操作而空闲

史前速度矛盾及缓和途径

□ 人机矛盾

- 人工操作方式与资源利用率之间的矛盾
- 伴随CPU速度提高、系统规模的扩大而日趋严重

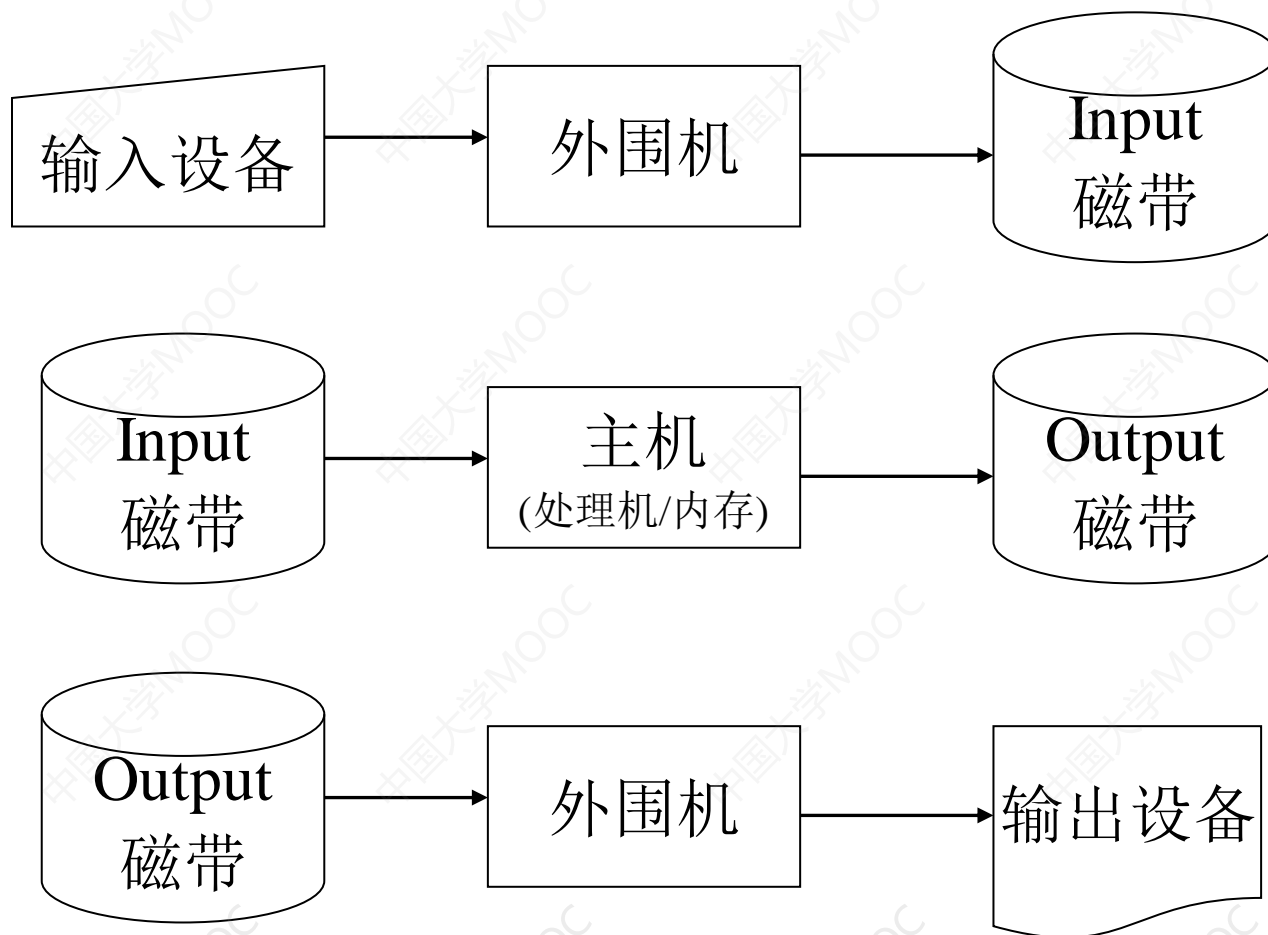
□ CPU与I/O设备间矛盾

- CPU速度迅速提高而I/O设备速度提高缓慢

□ 缓和途径

- 通道技术、中断技术、缓冲技术
- 脱机输入输出技术
- 专门的操作员及批处理技术

脱机输入输出技术



1.5 操作系统的发展

1.5.1 推动操作系统发展的主要动力

1.5.2 无操作系统时的计算机系统

1.5.3 单道批处理系统

1.5.4 多道批处理系统

1.5.5 分时系统

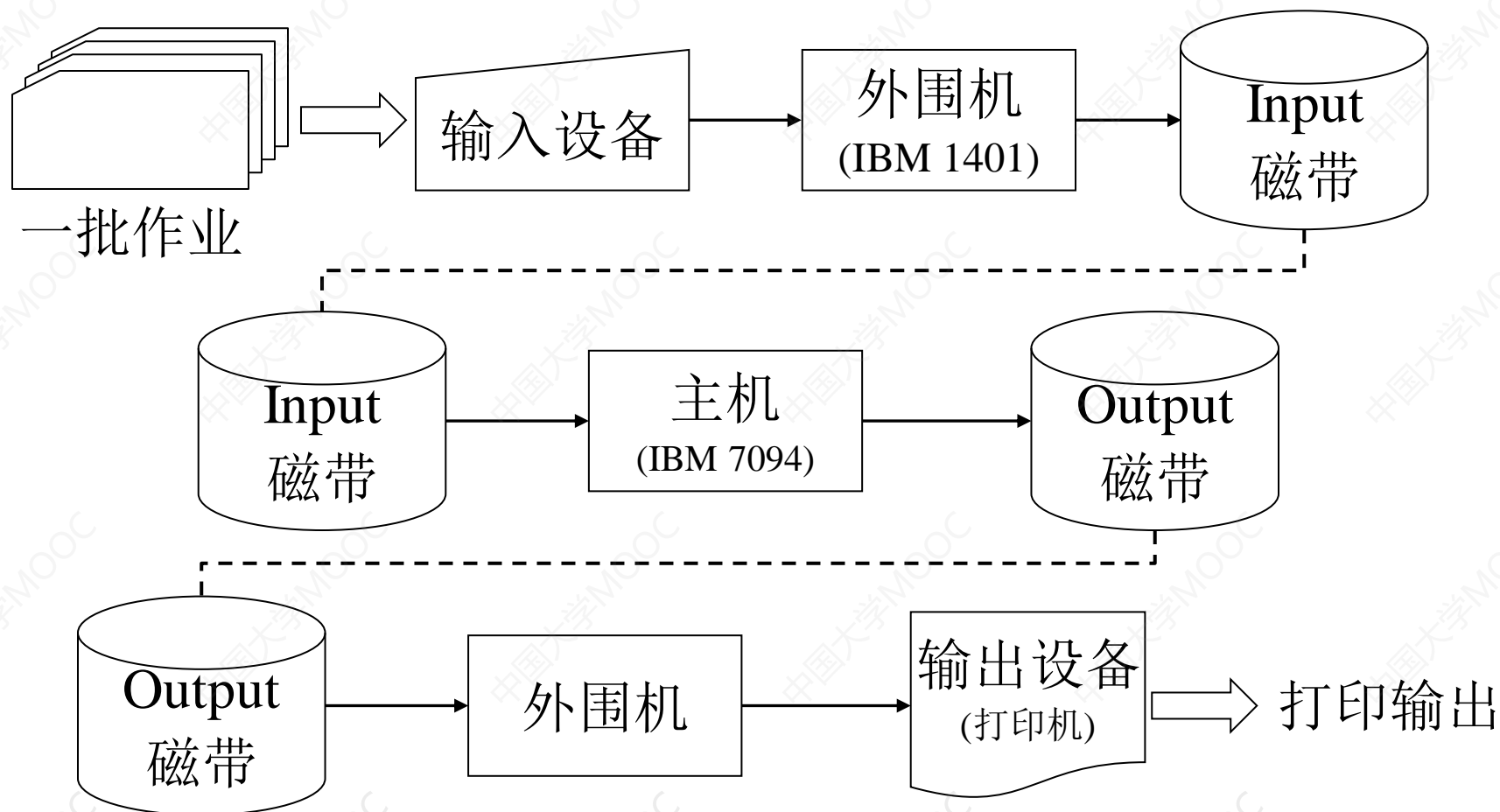
1.5.6 实时系统

1.5.7 操作系统的进一步发展

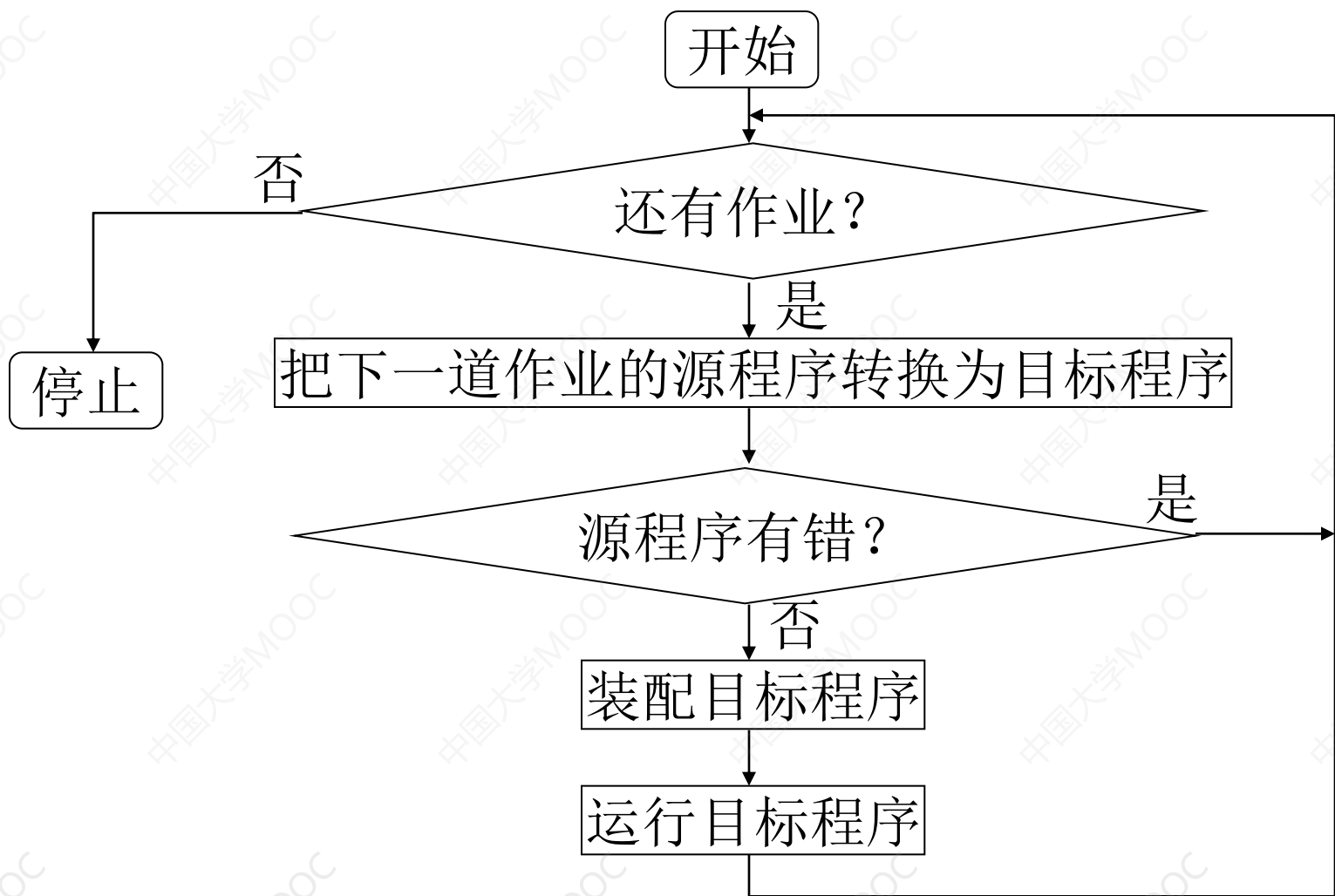
单道批处理系统

- ❑ 为解决人机矛盾和CPU与I/O设备速率不匹配的矛盾而形成，也即其旨在提高系统资源的利用率和系统吞吐量
- ❑ 把一批作业以脱机输入方式输入到磁带上，并在系统所配置的**监督程序**的控制下使这批作业能一个接一个地自动依次连续处理。
- ❑ 系统对作业的处理成批进行，但在内存中始终之保存着一道作业

单道批处理系统示意图



单道批处理系统**监控程序**处理流程



单道批处理系统的特征

□ 自动性

- 磁带上的一批作业能自动地逐个依次执行，而无需人工干预

□ 顺序性

- 磁带上的各道作业是顺序地进入内存，各道作业完成的顺序与它们进入内存的顺序完全一致

□ 单道性

- **监控程序**每次仅从磁带上调入一道程序进入内存运行，仅当该程序完成或发生异常情况时，才调入其后继程序进入内存运行

1.5 操作系统的发展

1.5.1 推动操作系统发展的主要动力

1.5.2 无操作系统时的计算机系统

1.5.3 单道批处理系统

1.5.4 多道批处理系统

1.5.5 分时系统

1.5.6 实时系统

1.5.7 操作系统的进一步发展

多道程序设计基本概念

❑ 单道批处理系统缺陷

- 系统资源空闲问题

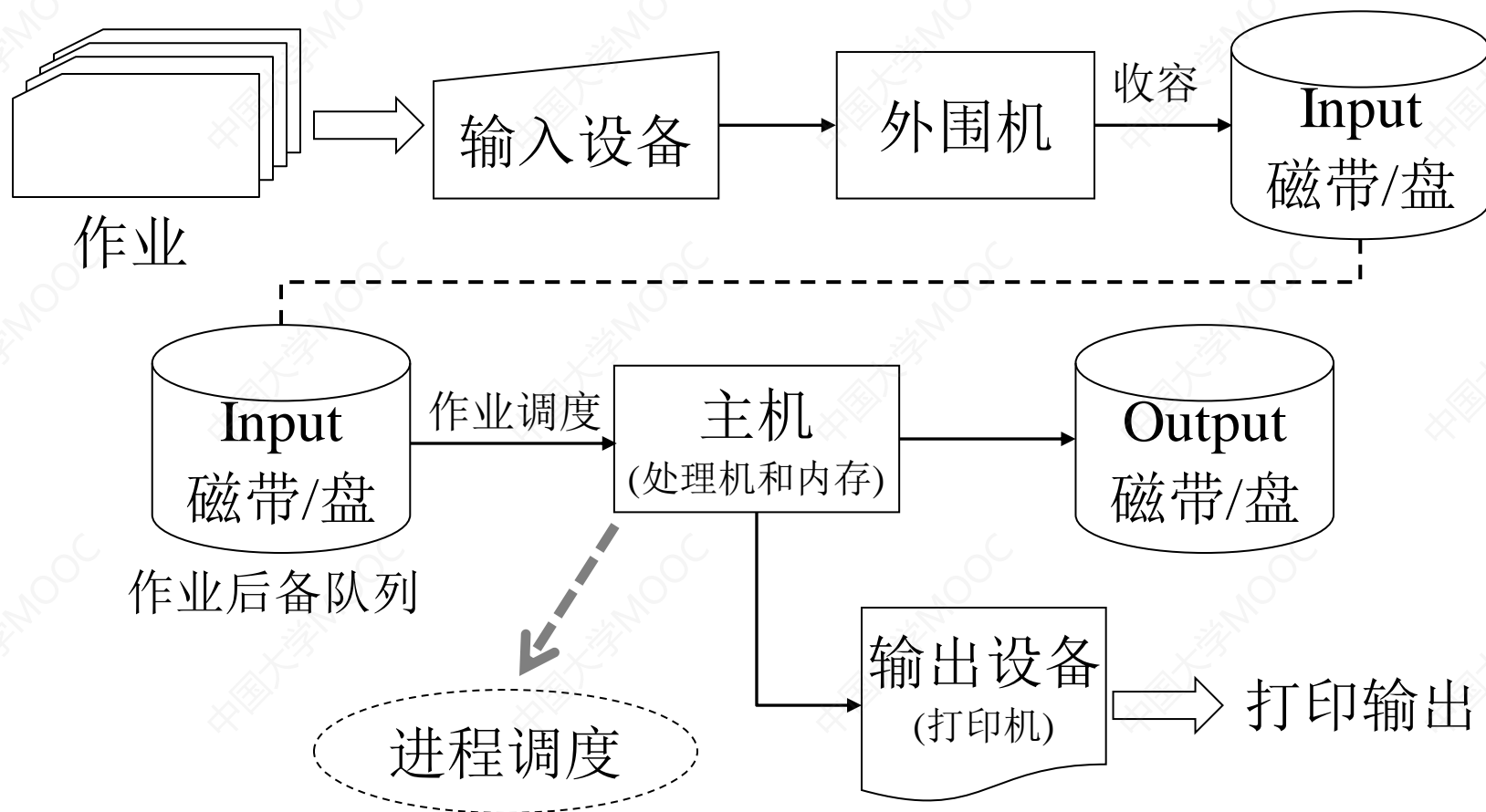


?

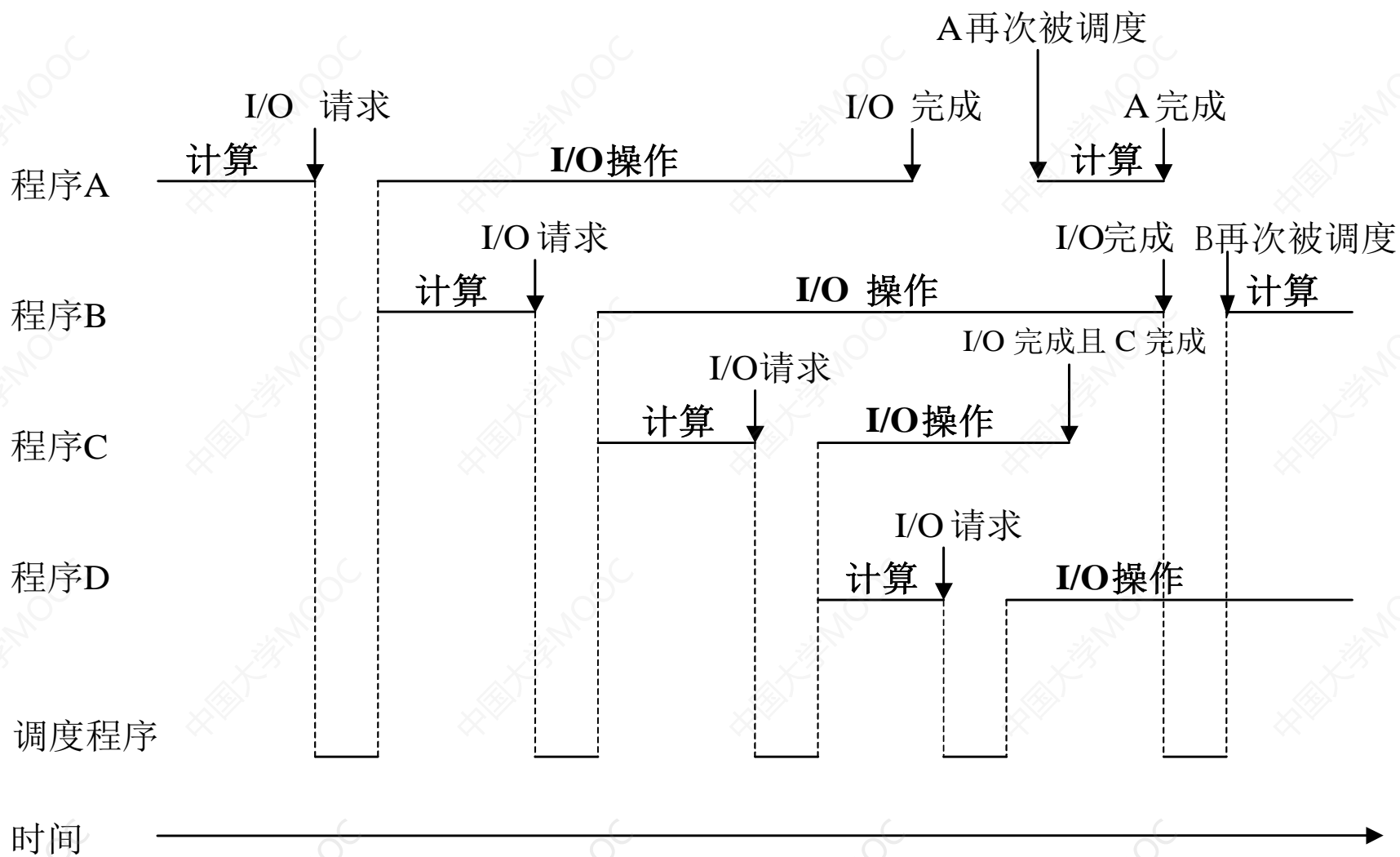
❑ 多道程序设计技术

- 作业后备队列/作业调度算法/系统资源共享
- 包括CPU、内存和I/O设备在内的系统资源利用率的提高
- 系统吞吐量增加

多道批处理系统示意图



多道程序运行情况



多道批处理系统特征

□ 多道性

- 内存中同时驻留多道程序，并允许并发执行

□ 无序性

- 多个作业完成的先后次序与它们进入内存的顺序之间，并无严格的对应关系

□ 调度性

- 作业调度
- 进程调度

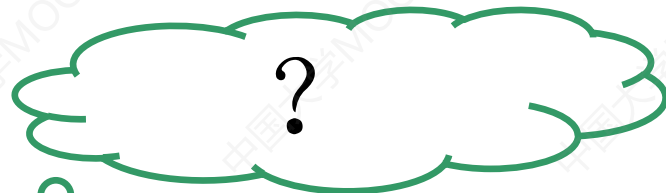
多道批处理系统优缺点

□ 优点

- 系统资源利用率高
- 系统吞吐量大

□ 缺点:

- 作业平均周转时间长，特别对短作业不公平
- 无交互能力，不利于程序调试和修改



多道批处理系统需求分析

❑ 处理机管理问题

- 多道程序之间应如何分配被它们共享的处理机，使正确运行且提高处理机利用率；分配与回收

❑ 内存管理问题

- 内存分配与保护

❑ I/O设备管理问题

- 设备共享、分配及利用率提高

❑ 文件管理问题

- 文件组织方便用户使用
- 数据安全性及一致性保证

❑ 作业管理问题

- 作业调度及系统工作流程组织

1.5 操作系统的发展

1.5.1 推动操作系统发展的主要动力

1.5.2 无操作系统时的计算机系统

1.5.3 单道批处理系统

1.5.4 多道批处理系统

1.5.5 分时系统

1.5.6 实时系统

1.5.7 操作系统的进一步发展

分时系统的产生

□ 人机交互

- 程序修改与调试、直接控制

□ 共享主机

- 20世纪60年代计算机十分昂贵

□ 便于用户上机

- 通过自己终端直接将作业传送到机器上进行处理，并能对自己的作业进行控制

□ 分时系统概念

- 一台主机、多个终端、多用户同时以交互方式使用计算机

分时系统实现中的关键问题

□ 及时接收用户输入命令和数据

- 配置多路卡及设置多路缓冲区

□ 及时处理

- 使所有用户作业都直接进入内存
- 在不长的时间内就能使每个作业都运行一次

分时系统的实现方法与方式

□ 作业应直接进入内存

- 这与批处理系统用户作业先进入磁盘、然后再调入内存不同

□ 时间片轮转策略

- 时间片

□ 实现方式

- 单道分时系统
- 具有前台和后台的分时系统
- 多道分时系统

分时系统的特征

□ 多路性

- 宏观上多个用户同时工作和共享系统资源
- 微观上每个用户作业轮流运行一个时间片

□ 独立性

- 各用户在各自终端上独立操作，互不干扰

□ 及时性

- 响应时间

□ 交互性

- 用户可通过终端与系统进行广泛的人机对话

1.5 操作系统的发展

1.5.1 推动操作系统发展的主要动力

1.5.2 无操作系统时的计算机系统

1.5.3 单道批处理系统

1.5.4 多道批处理系统

1.5.5 分时系统

1.5.6 实时系统

1.5.7 操作系统的进一步发展

实时系统

□ 实时系统的引入

- 实时控制系统
- 实时信息处理系统

□ 实时系统的概念

- 指系统能及时或即时响应外部事件的请求，在规定的时间内完成对该事件的处理，并控制所有实时任务协调一致地运行

□ 实时任务的类型

- 按任务执行是否呈现周期性来划分
- 根据对截止时间的要求来划分

实时系统与分时系统的比较

- ❑ 多路性
- ❑ 独立性
- ❑ 及时性
- ❑ 交互性
- ❑ 系统高度可靠

1.5 操作系统的发展

1.5.1 推动操作系统发展的主要动力

1.5.2 无操作系统时的计算机系统

1.5.3 单道批处理系统

1.5.4 多道批处理系统

1.5.5 分时系统

1.5.6 实时系统

1.5.7 操作系统的进一步发展

操作系统的进一步发展

- ❑ 通用操作系统
- ❑ 微机操作系统
- ❑ 多处理机操作系统
- ❑ 网络操作系统 & 分布式操作系统
- ❑ 集群操作系统
- ❑ 网格操作系统 & 云操作系统
- ❑ 嵌入式操作系统 & 移动操作系统

需求特征?

1.5 操作系统的发展

1.5.1 推动操作系统发展的主要动力

1.5.2 无操作系统时的计算机系统

1.5.3 单道批处理系统

1.5.4 多道批处理系统

1.5.5 分时系统

1.5.6 实时系统

1.5.7 操作系统的进一步发展

作业题

- **1.5** 谈谈你对脱机I/O和联机I/O的认识与理解。
- **1.6** 试从多个角度来阐述单道/多道批处理系统与分时系统及实时系统的区别。

第一章 操作系统引论

1.1 什么是操作系统

1.2 计算机体系结构及操作系统硬件基础

1.3 操作系统用户接口及系统调用实现

1.4 操作系统启动模块及自装入机制

1.5 操作系统的发展

1.6 操作系统的功能与特征

1.7 操作系统的结构设计

1.6 操作系统的功能与特征

1.6.1 处理机管理功能

1.6.2 存储器管理功能

1.6.3 设备管理功能

1.6.4 文件管理功能

1.6.5 操作系统的特征

处理机调度

□ 作业调度

- 作业后备队列
- 作业选择与资源分配
- 调入内存与建立进程

□ 进程调度

- 就绪进程队列
- 进程选择与处理机分配
- 设置运行现场与启动运行

□ 调度算法

- 先来先服务/优先权高者优先调度算法

进程控制

□ 主要任务

- 创建进程
- 撤销进程
- 进程状态转换

□ 进程控制机制

- 原语
- 进程&线程

进程同步

□ 主要任务

- 进程/线程并发执行协调
- 互斥/同步方式
- 临界资源&临界区

□ 进程同步机制

- 开/关锁原语
- 信号量机制
- 管程

进程通信

□ 主要任务

- 进程（或线程）间信息交换

□ 进程通信方式

- 共享存储器
- 管道方式
- 消息传递系统
 - A. 消息缓冲队列
 - B. 邮箱

1.6 操作系统的功能与特征

1.6.1 处理机管理功能

1.6.2 存储器管理功能

1.6.3 设备管理功能

1.6.4 文件管理功能

1.6.5 操作系统的特征

内存分配

□ 主要任务

- 使程序各得其所
- 提高存储器利用率
- 适应程序和数据动态增长的需要

□ 内存分配机制

- 内存分配用数据结构
- 内存分配
 - A. 连续/离散分配方式
 - B. 静态/动态分配方式
- 内存回收

内存保护

□ 主要任务

- 确保程序间互不干扰
- 存取访问控制

□ 内存保护机制

- 越界检查
- 硬件实现
- 保护方式
 - A. 上下界限寄存器
 - B. 页号 < 页表长度
 - C. 段号 < 段表长度 && 段内地址 < 段长
 - D. 特权级比较

地址映射

□ 主要任务

- 逻辑地址转换为物理地址

□ 基本概念

- 地址空间&内存空间
- 逻辑/相对/有效地址
- 物理/绝对地址

□ 地址映射机制

- 硬件实现
- 重定位寄存器、页表/段表、快表

内存扩充

□ 主要任务

- 从逻辑上扩充内存容量

□ 内存扩充可行性

- 程序运行局部性原理
- 离散分配方式

□ 内存扩充机制

- 虚拟存储技术
- 硬件（页/段表、缺页/段中断、地址变换）
- 软件（请求调入功能/置换功能）

1.6 操作系统的功能与特征

1.6.1 处理机管理功能

1.6.2 存储器管理功能

1.6.3 设备管理功能

1.6.4 文件管理功能

1.6.5 操作系统的特征

设备分配

□ 主要任务

- 设备及相应设备控制器和通道的分配

□ 设备分配机制

- 设备分配用数据结构
- 设备类型与设备分配方式相对应
- 设备分配与回收算法
- 设备独立性
- 虚拟设备

缓冲管理

□ 主要任务

- 管理各种类型的缓冲区
- 缓和CPU和I/O设备速度不匹配矛盾
- 提高资源利用率和系统吞吐量

□ 缓冲管理机制

- 单/双/多缓冲类型
- 字符缓冲与盘块缓冲
- 公用缓冲池机制

设备处理

□ 主要任务

- 实现CPU和设备控制器间通信

□ 设备处理过程

- I/O请求提出
- I/O请求合法性检查
- 了解设备状态
- 传递相关参数并设置设备工作方式
- 通道程序自动构成
- 发出I/O指令和启动I/O设备
- 及时响应中断请求

1.6 操作系统的功能与特征

1.6.1 处理机管理功能

1.6.2 存储器管理功能

1.6.3 设备管理功能

1.6.4 文件管理功能

1.6.5 操作系统的特征

文件存储空间管理

□ 主要任务

- 使每个文件各得其所
- 提高外存空间利用率
- 提高外存访问速度

□ 文件存储空间管理机制

- 存储空间管理用数据结构
- 存储空间分配与回收功能
- 连续/离散分配方式
- 以盘块/簇为基本分配单位

目录管理

□ 主要任务

- 实现文件的按名存取
- 提高文件查找速度
- 支持文件重名、共享与保护

□ 目录管理机制

- 文件控制块与索引结点
- 目录结构
- 目录检索手段

文件的读写管理和存取控制

□ 文件的读写管理

- 从外存中读取数据或将数据写入外存
- 目录检索->外存地址->读写指针->读写操作
- 文件描述符表和文件表

□ 文件的存取控制

- 防止未经核准的用户存取文件
- 防止冒名顶替存取文件
- 防止以不正确的方式使用文件
- 系统级/用户级/文件级存取控制

作业题

- 1.7 谈谈你对操作系统应当具备的资源管理功能（包括所引入的核心概念）的认识与理解。为保证资源的有效管理与控制，操作系统于是体现出哪些特征？

1.6 操作系统的功能与特征

1.6.1 处理机管理功能

1.6.2 存储器管理功能

1.6.3 设备管理功能

1.6.4 文件管理功能

1.6.5 操作系统的特征

操作系统的特征

□ 并发

- 并行与并发、程序与进程/线程

□ 共享

- 互斥共享方式、同时访问方式

可重入码

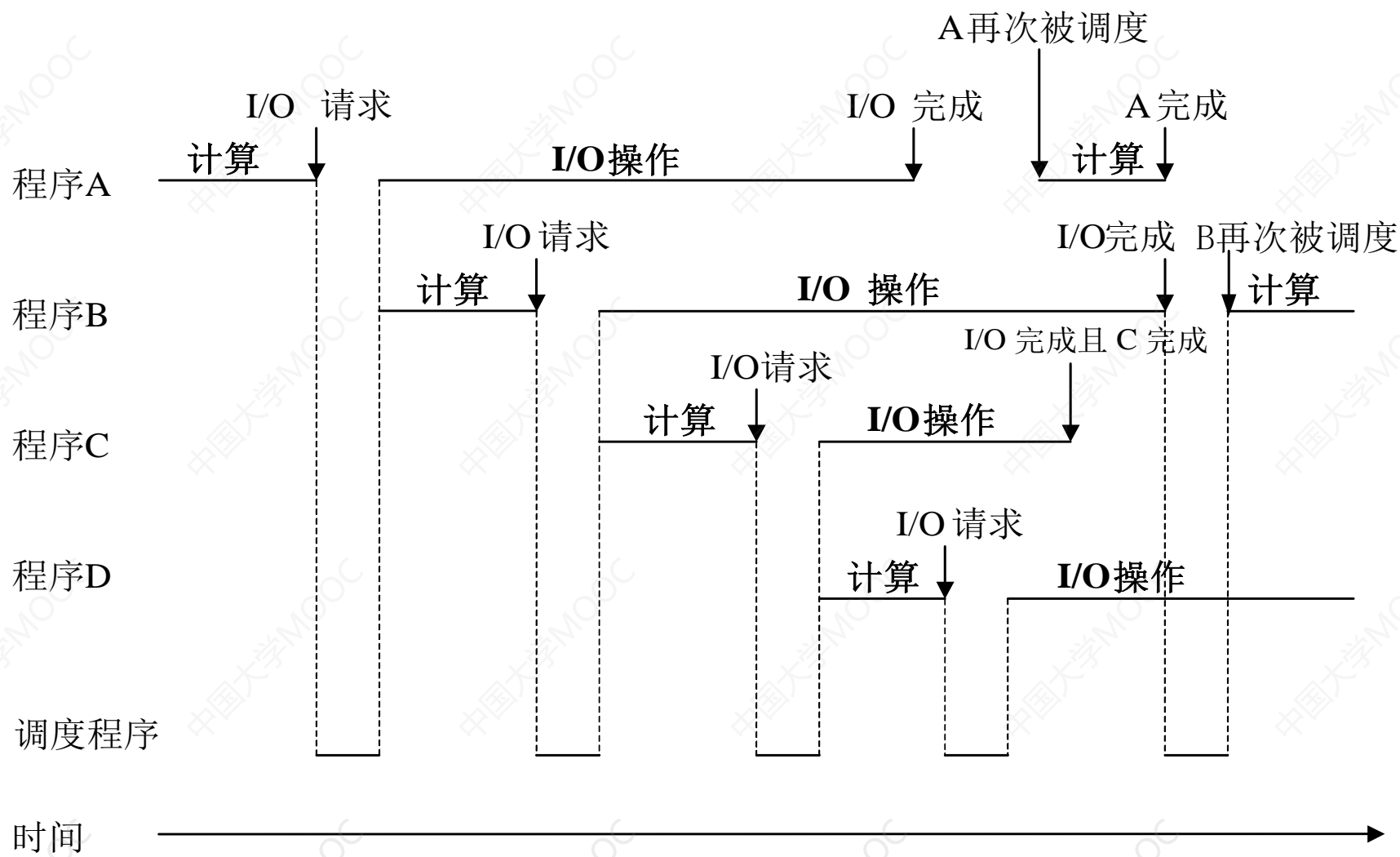
□ 虚拟

- 虚拟处理机、虚拟内存、虚拟盘、虚拟设备

□ 异步性

- 进程执行顺序与执行时间的不确定性

异步性举例说明



第一章 操作系统引论

1.1 什么是操作系统

1.2 计算机体系结构及操作系统硬件基础

1.3 操作系统用户接口及系统调用实现

1.4 操作系统启动模块及自装入机制

1.5 操作系统的发展

1.6 操作系统的功能与特征

1.7 操作系统的结构设计

1.7 操作系统的结构

1.7.1 操作系统的设计原则

1.7.2 无结构操作系统

1.7.3 模块化操作系统结构

1.7.4 分层式操作系统结构

1.7.5 微内核操作系统结构

操作系统的设计原则

❑ 可维护性

- 纠错性/适应性/完善性/预防性维护

❑ 可靠性

- 正确性/健壮性

❑ 可理解性

❑ 可用性

❑ 性能

- 系统资源利用率及用户请求响应

1.7 操作系统的结构

1.7.1 操作系统的设计原则

1.7.2 无结构操作系统

1.7.3 模块化操作系统结构

1.7.4 分层式操作系统结构

1.7.5 微内核操作系统结构

无结构操作系统 (整体系统结构)

- ❑ 致力于功能实现和提高效率
- ❑ 缺乏首尾一致的设计思想
- ❑ 过程集合内各过程间可相互调用
- ❑ 操作系统内部不存在任何结构
- ❑ GOTO语句不加限制
- ❑ 系统易出错、调试困难、维护麻烦

1.7 操作系统的结构

1.7.1 操作系统的设计原则

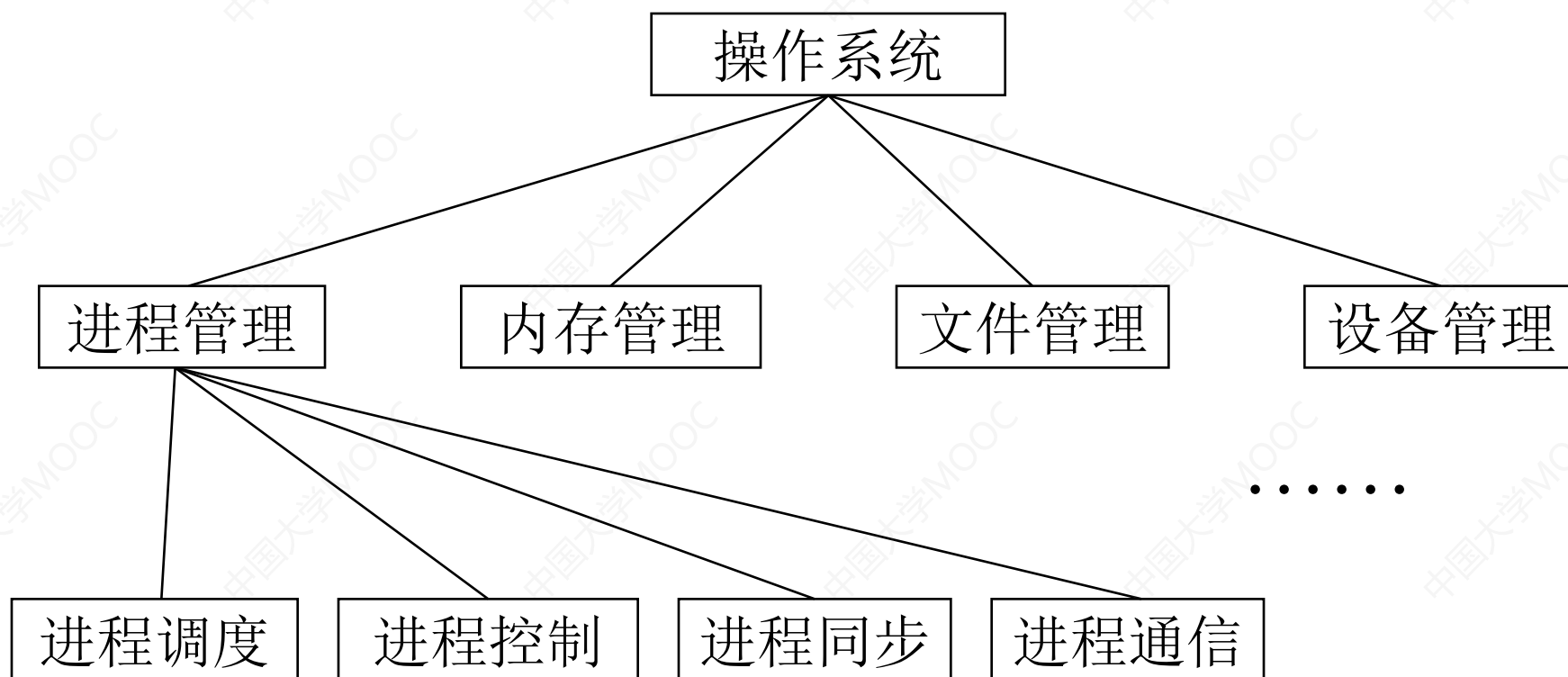
1.7.2 无结构操作系统

1.7.3 模块化操作系统结构

1.7.4 分层式操作系统结构

1.7.5 微内核操作系统结构

模块化操作系统结构



模块化操作系统结构评价

□ 优点

- 提高了设计的正确性、可理解性和可维护性
- 增强了操作系统的可适应性
- 加速了操作系统的开发过程

□ 尚存改进空间方面

- 模块划分和接口规定难保正确和合理
- 未能区别共享资源和独占资源
- 管理差异导致模块间依赖关系复杂

1.7 操作系统的结构

1.7.1 操作系统的设计原则

1.7.2 无结构操作系统

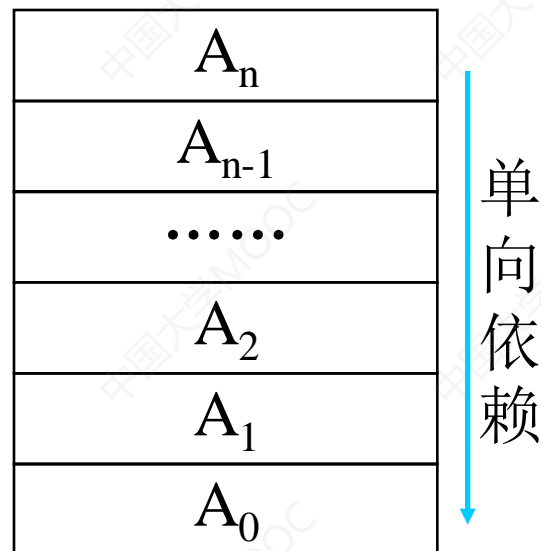
1.7.3 模块化操作系统结构

1.7.4 分层式操作系统结构

1.7.5 微内核操作系统结构

分层式操作系统结构及分层原则

- ❑ 被调用功能放在较低层次
 - 作业调度->进程控制->内存管理
- ❑ 活跃功能放在低层
 - 时钟管理、进程调度
- ❑ 资源分配策略放在高层
 - 便于修改或适应不同环境
- ❑ 最低层
 - 资源管理公用模块如队列、堆栈、信号量操作等
- ❑ 最高层
 - 用户接口



分层式操作系统结构评价

- ❑ 从资源管理角度出发进行层次划分
- ❑ 规定模块间调用的有序性
- ❑ 结构特点及评价
 - 调用关系清晰（高层对低层单向依赖）
 - 低层和高层可分别实现（可扩充性）
 - 高层错误不会影响到低层（正确性）
 - 避免了递归调用
 - 可增强系统可维护性
 - 降低了系统运行效率

1.7 操作系统的结构

1.7.1 操作系统的设计原则

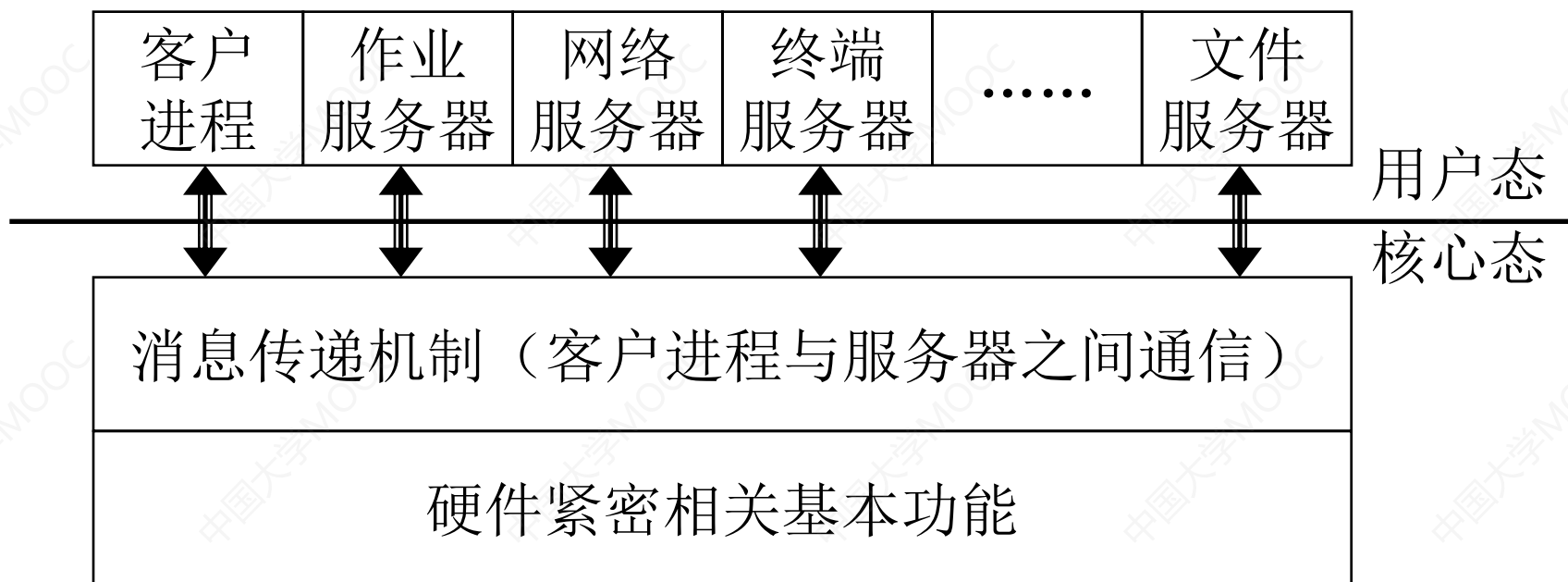
1.7.2 无结构操作系统

1.7.3 模块化操作系统结构

1.7.4 分层式操作系统结构

1.7.5 微内核操作系统结构

微内核操作系统结构客户/服务器模型



微内核操作系统结构要领及评价

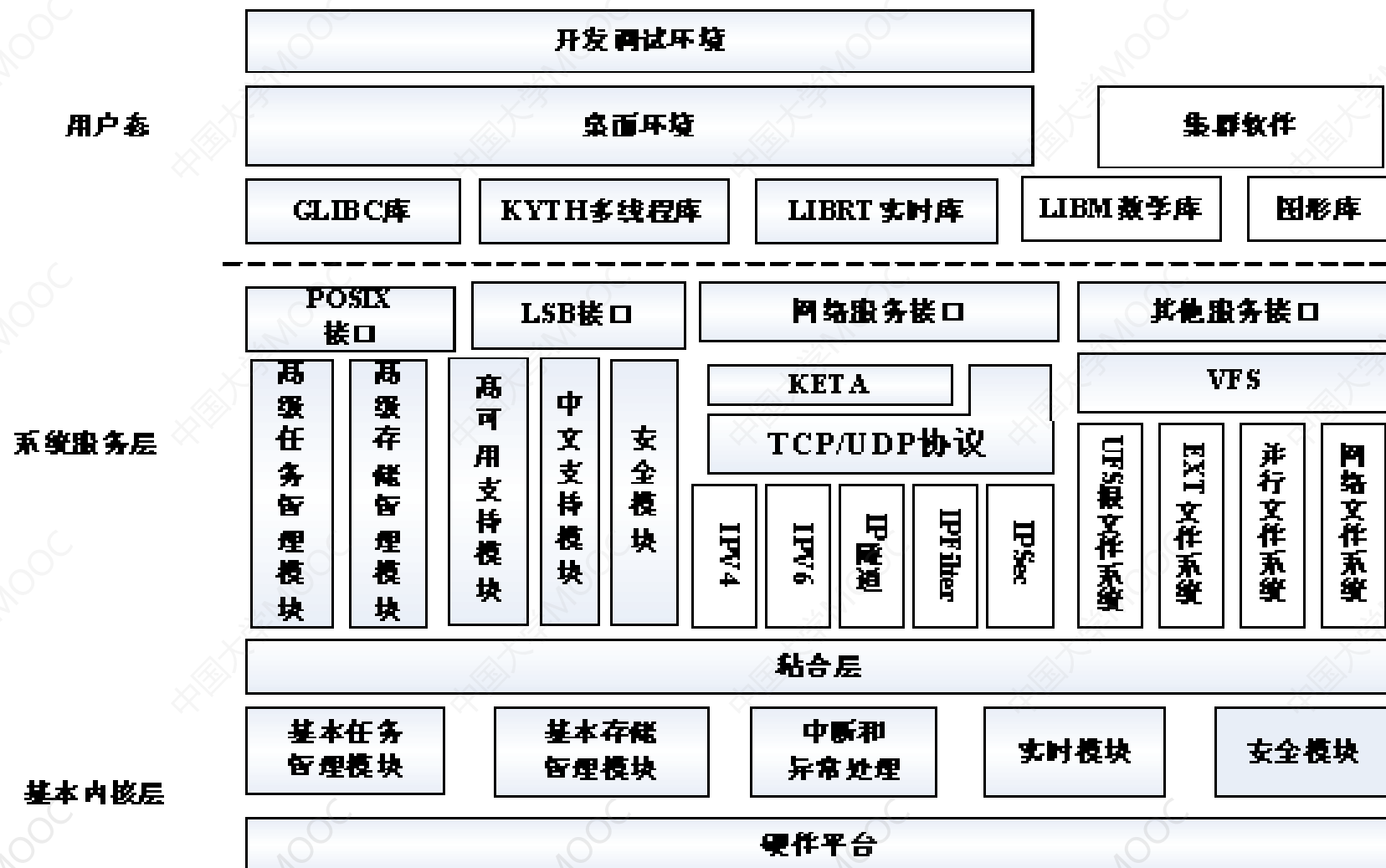
□ 机制与策略分离、客户/服务器模式

- 内核仅实现硬件相关及最基本功能：进程调度与控制、进程同步与通信、地址转换机制、中断/陷入处理、设备驱动
- 其余更多系统功能放在内核之外：用户进程分类及优先级计算、页面淘汰算法、分配策略

□ 结构评价

- 可扩充性、正确性、可靠性
- 便于网络服务和实现分布式处理
- 系统效率受到影响

Kylin麒麟操作系统结构



1.7 操作系统的结构

1.7.1 操作系统的设计原则

1.7.2 无结构操作系统

1.7.3 模块化操作系统结构

1.7.4 分层式操作系统结构

1.7.5 微内核操作系统结构

作业题

- 1.8 操作系统的基本设计原则有哪些？
谈谈你对操作系统发展历程中所采用的
几种结构的认识与理解。

第一章 操作系统引论

1.1 什么是操作系统

1.2 计算机体系结构及操作系统硬件基础

1.3 操作系统用户接口及系统调用实现

1.4 操作系统启动模块及自装入机制

1.5 操作系统的发展

1.6 操作系统的功能与特征

1.7 操作系统的结构设计

小结

- ❑ 操作系统的目标、作用和组成
- ❑ 操作系统的发展及主要动力
- ❑ 各类操作系统之间的比较
- ❑ 操作系统的功能
- ❑ 操作系统的特征和服务
- ❑ 操作系统的设计结构
- ❑ 主流操作系统

第一章习题讲解

作业题

- **1.1** 什么是操作系统？用自己的话谈谈你对操作系统概念的认识与理解。
- **1.2** 设想由你自己负责组织一个项目团队来构建操作系统，你应当要求项目成员事先学习和掌握哪些硬件基础知识？并给出你对相关知识的理解与总结。
- **1.3** 什么是输入输出重定向？什么是管道联接？分别加以举例说明。

作业题

- **1.4** 试阐述程序接口与用户交互接口（即命令接口和图形化接口）之间的关系？并给出你对系统调用实现机制及处理过程的完整理解与总结。
- **1.5** 谈谈你对脱机I/O和联机I/O的认识与理解。
- **1.6** 试从多个角度来阐述单道/多道批处理系统与分时系统及实时系统的区别。

作业题

- **1.7** 谈谈你对操作系统应当具备的资源管理功能的认识与理解。为保证资源的有效管理与控制，操作系统于是体现出哪些特征？
- **1.8** 操作系统的基本设计原则有哪些？谈谈你对操作系统发展历程中所采用的几种结构的认识与理解。



**同学们，
再见！**

2021年4月27日星期二