

UNIX/Linux

操作系统内核结构

刘玏 教授

电子科技大学信软学院

教师介绍

刘玏 教授

大型主机系主任

数据处理与可视化计算科研团队负责人

大型主机教学团队负责人

Email: liudi@uestc.edu.cn

电话: 83206317（办），18980803219（手机）

主要研究方向: 操作系统、大型主机、大数据处理

课程概述

一. 课程内容简介

1、讲授范围

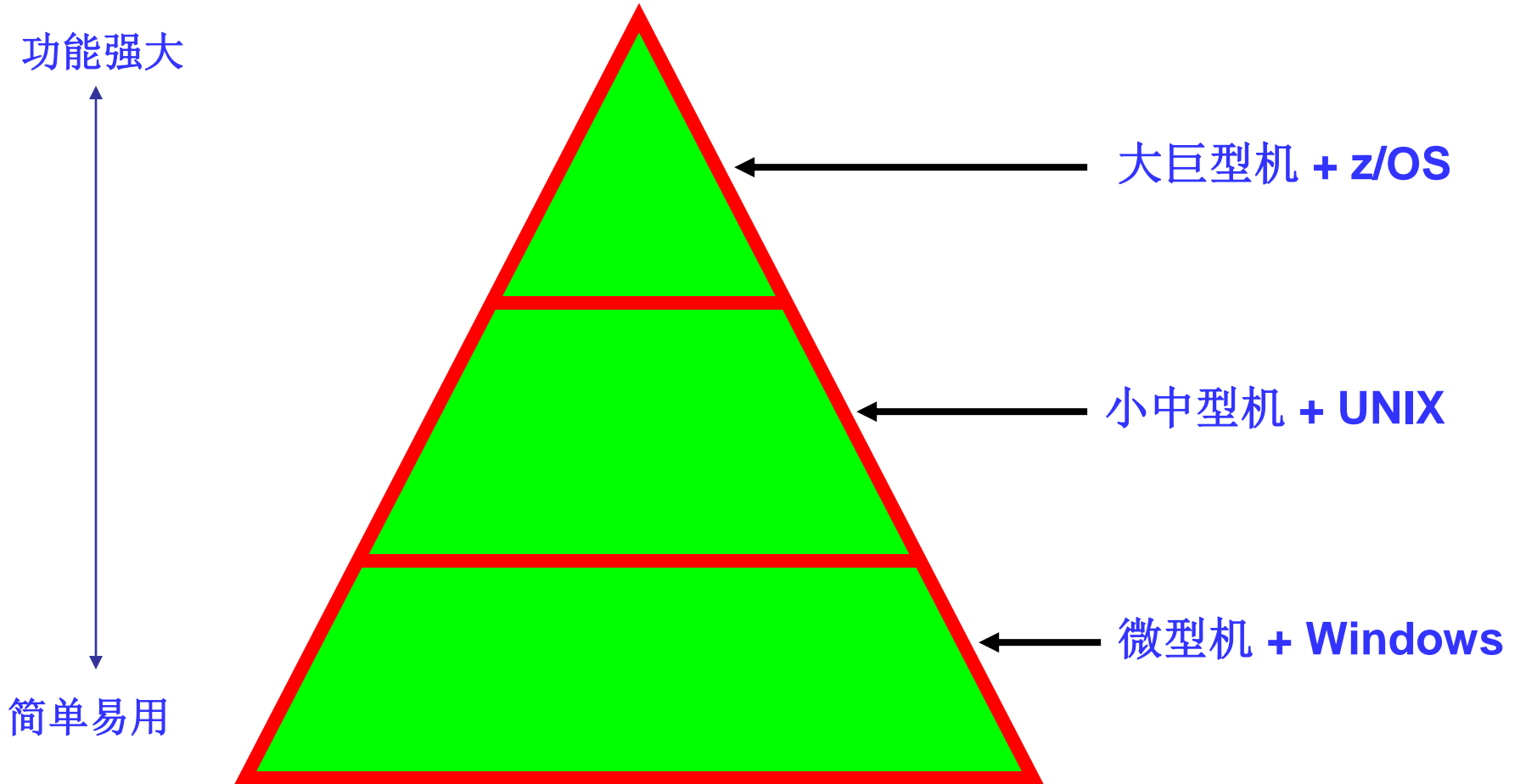
具体的技术系统及其算法和实现流程，而不是操作系统基本概念；

2、通用操作系统的现状和分类

DOS类 ---- 结构简单、使用方便、效率低、安全性低

UNIX类 ---- 运行高效、结构通用、安全可靠、适应能力强、系统较复杂

MVS类 ---- 功能强大、处理能力巨大、系统复杂、较封闭



3、UNIX操作系统的根本特点

分时多用户、开放性

分时多用户：

多个用户多个进程同时在一个系统中运行

系统资源高度共享、有效协调——并发

开放性：

标准化——结构上的一致性

可移植性——应用程序的编码及系统应用接口

可互操作性——可保持用户原来的使用习惯

异种机之间的互操作

4、教学难点

多用户多进程——同步/互斥、数据一致性、访问安全性

开放性——硬件依赖性、结构伸缩性、广泛适应性

二、教学目的

1、了解主流操作系统的发展方向

低端操作系统 VS 高端操作系统

2、掌握UNIX类操作系统的内部结构和主要算法

文件、文件系统、进程、时钟、输入输出

3、学习大型程序设计的方法和理念

系统结构、功能流程、数据安全、思维模式

4、奠定系统开发和应用开发的基础

功能选择、层次划分、应用系统模式的确定

三、教材

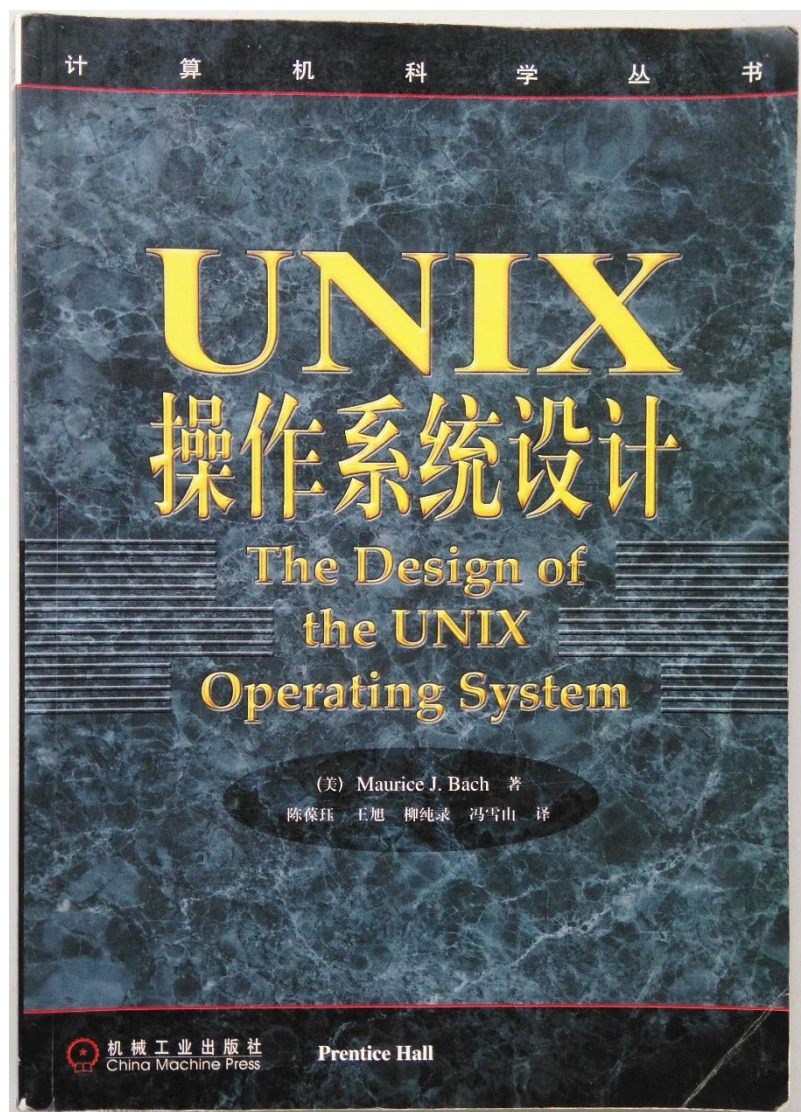
《UNIX操作系统设计》

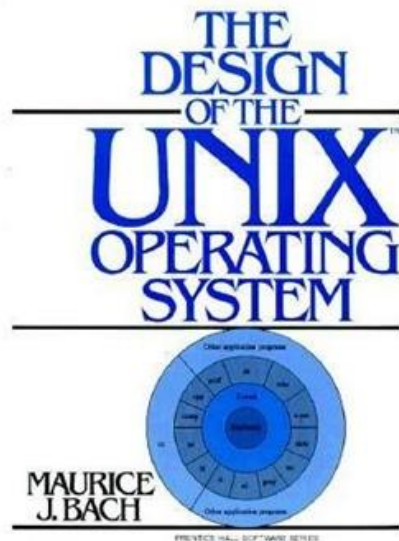
(The Design of the UNIX
Operating System)

(美) Maurice J. Bach 著

陈葆珏 王旭 柳纯录 冯雪
山 译

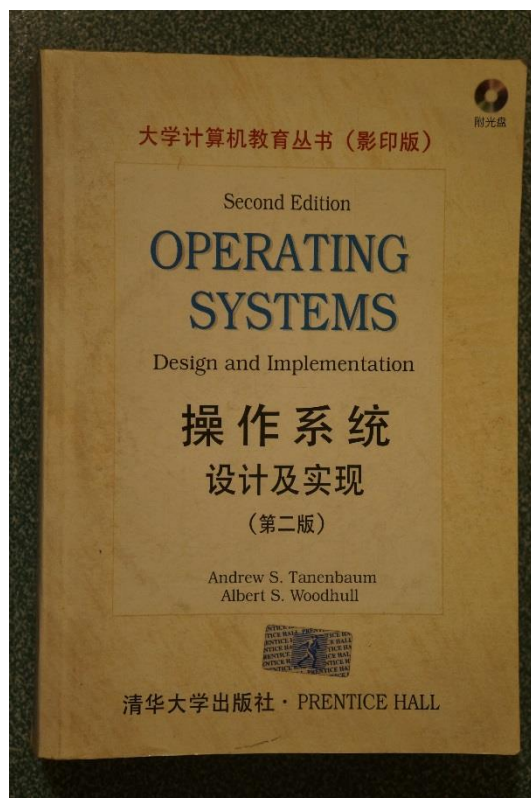
机械工业出版社 2005年10
月出版





参考资料:

1. The Design of the UNIX Operating System（影印）. Maurice J. Bach，人民邮电出版社。



2. Operating Systems: Design and Implementation（第二版）（影印）. Audreus. Tanenbaum 等，清华大学出版社。

进一步阅读:

1、《UNIX编程艺术》

The art of UNIX Programming

[美] Raymond 著

电子工业出版社, 2006年

2、《软件艺术》

Software Craftsmanship

[美] 麦克布林 著

人民邮电出版社, 2004年

3、《程序开发心理学》

The Psychology of Computer Programming

[美] Weinbery 著

清华大学出版社, 2003年

四、考核说明

成绩构成：平时成绩 **20%** + 期末报告**80%**

第一章 系统概貌

- 1.1 发展状况

- 1、发展历史及版本

- v.0 1970年

- Ken Thompson 和 Dennis Ritchie

- PDP-7 汇编语言

- UNICS

- v.1 1971年

- PDP-11 汇编语言

- UNIX

- v.2 1972年 增加管道功能

v.5 1973年
Dennis Ritchie
B language ---- C language

重写UNIX

第一个高级语言OS

v.6 1975年

对外发表UNIX

大学和科研单位应用

v.7 1978年

第一个商业版本

我国开始深入研究应用的最早版本

System III

1981年

完全转向为社会提供的商品软件

System V

1983年

系统功能稳定完善

公布号: 1.0、2.0、2.3、3.5、4.0、4.2、4.3

现在最后版本为 System V Release 4 (SVR4)

2、主要分支和兼容版本

- **BSD** 加州大学伯克利分校
- **XENIX/OpenServer**
 Microsoft、SCO公司
- **HP-UX** **HP公司**
- **AIX** **IBM**
- **Solaris** **SUN公司**
- **IRIX** **SGI公司**
- **Ultrix** **DEC公司**
- **Linux** 开放源代码

3、基本功能特征

① 交互式分时多用户

- 人机间实时交互数据
- 多个用户可同时使用一台机器
- 每个用户可同时执行多个任务

② 软件复用

- 每个程序模块完成单一的功能
- 程序模块可按需任意组合
- 较高的系统和应用开发效率

③ 可移植性强

- 数千行汇编码，数十万行**C**语言代码

- ④ 配置灵活, 适应性强
 - 小内核, 参数灵活可调
 - 核外应用系统, 任意裁减
 - 限制规则很少
- ⑤ 界面方便高效
 - 内部: 系统调用丰富高效
 - 外部: **shell**命令灵活方便可编程
 - 应用: **GUI** 清晰直观功能强大
- ⑥ 安全机制完善
 - 口令、权限、加密等措施完善
 - 抗病毒结构
 - 误操作的局限和自动恢复功能

⑦ 多国语言支持

- 支持全世界现有的几十种主要语言

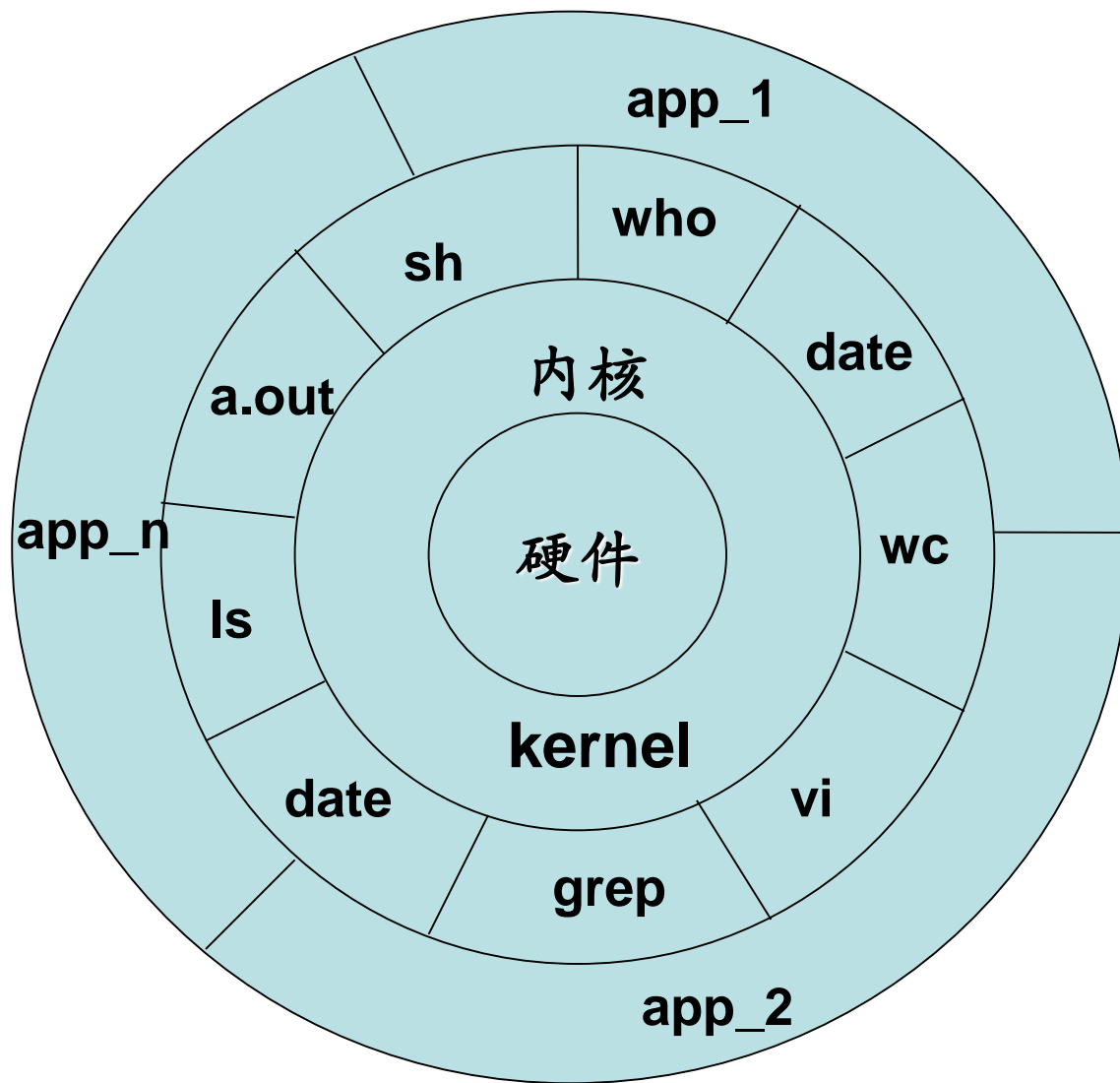
⑧ 网络和资源共亨

- 内部：多进程结构易于资源共享
- 外部：支持多种网络协议

说明：

- 1、其它操作系统可能包含部分上述**UNIX**的特征，但非全部（如**NT**就有部分多用户系统特征）
- 2、这些特征有些是核心直接实现的，有些是由核心提供实现这种特征的方便性和可能性，而由使用者来实现的。

- 1.2 系统结构



UNIX操作系统的整体结构

系统调用（**system call**）

以函数形式提供给核外的命令和上层应用系统使用的一组程序，涵盖操作系统的所有功能。是应用程序请求操作系统服务的唯一通道。

内核（**kernel**）

系统调用的集合及实现系统调用的内部算法就形成操作系统核心

• 1.3 用户看法

进程和文件是**UNIX**操作系统中最基本的两个概念（抽象）

进程：

所有处在运行期间的程序实例都是进程

一个进程就是处在运行期间的一个程序实例

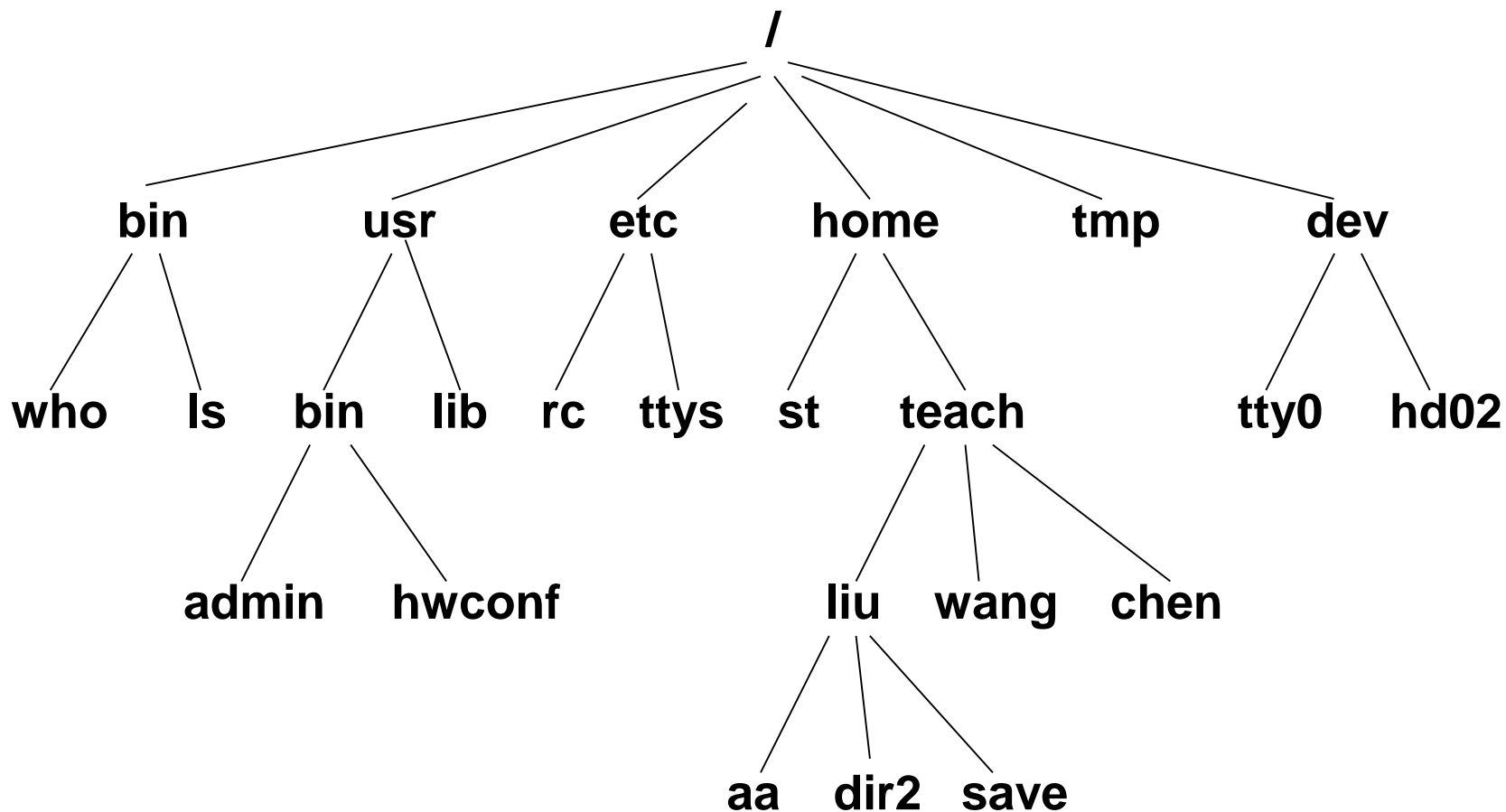
涵盖所有的动态概念

文件：

所有静态的无形数据和有形硬件设备

源程序、命令、图片、邮件、打印机、内存、磁盘等

1.3.1 文件系统



UNIX文件系统树示例

UNIX文件系统的特征:

1、树状层次结构

树根、树枝、树叶、路径

2、对文件数据的一致对待

文件为有序无格式的字节流，逻辑意义由使用者解释

3、文件管理

建立、删除、修改、备份、移动、替换 —— 上层操作

存储空间的分配和释放 —— 下层操作

4、文件的访问和保护

索引节点 (**inode**)、文件描述符(**fd**)

用户分组、权限划分

5、设备文件管理

统一各外部设备的访问模式

```

char buffer[2048];
main(int argc, char *argv[])
{
    int fdold, fdnew;
    if(argc != 3)
    {
        printf("Need 2 arguments for copy program\n");
        exit(1);
    }
    fdold = open(argv[1], O_RDONLY);
    if (fdold == -1)
    {
        printf("cannot open file %s\n", argv[1]);
        exit(1);
    }
    fdnew = creat(argv[2], 0666);
    if(fdnew == -1)
    {
        printf("cannot create file %s\n", argv[2]);
        exit(1);
    }
    copy(fdold, fdnew);
    exit(0);
}

copy(int old, int new)
{
    int count;
    while((count = read(old, buffer, sizeof(buffer))) > 0)
        write(new, buffer, count);
}

```

1.3.2 处理环境

程序：可执行的文件



文件头包括：

- 文件的幻数（**magic number**）
- 编译器的版本号
- 机器类型
- 数据段、正文段、工作变量的段大小
- 程序入口点

进程:

程序的一次执行实例

一个程序可同时有多个实例; 系统中可同时有多个进程

父进程:

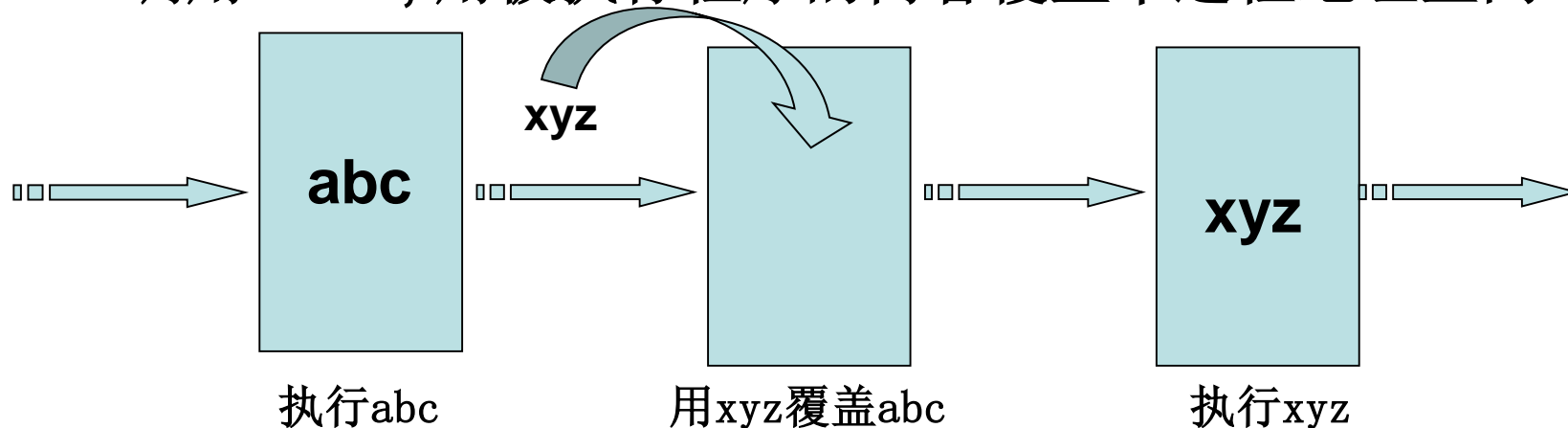
调用系统调用**fork**的进程

子进程:

由系统调用**fork**产生的新进程

执行程序:

调用**exec1**, 用被执行程序的内容覆盖本进程地址空间



例子：

执行可运行文件**copy**，其功能是拷贝文件，其运行格式为：
copy oldfile newfile

另一个名为**cpfile**的程序具体调用**copy**，**cpfile**的源程序如下：

```
main(int argc, char *argv[ ])
{
    if (fork() == 0)
        execl("copy", "copy", argv[1], argv[2]], 0);
    wait((int *)0);
    printf("copy done\n");
}
```

在用户环境下，程序的执行通常由命令解释器**shell**来完成，标准的命令格式为：

cmd [-options] [arguments]

shell可识别的命令类型有：

1、简单命令

cat file1

2、多条命令

who; date; ps

3、复合命令

ps -e | grep student2

(ls ; cat file3 ; pwd) > run_log

4、后台命令

ls -lR /home/teacher > tlist &

1.3.3 构件原语

“软件复用”和“模块组装”理念

程序内部：

简单功能划分；纯代码设计

程序外部：

使用构件原语进行功能重叠和组装

UNIX包含两种构件原语：

- ① 输入/输出重定向
- ② 管道

I/O重定向 (I/O redirect) :

一个进程通常(**default**)打开三个文件:

标准输入文件 (**fd=0**)

标准输出文件 (**fd=1**)

标准错误输出文件 (**fd=2**)

例如:

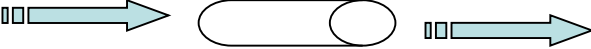
```
grep abc
```

```
grep abc < file1
```

```
grep abc < file1 > file2
```

```
grep abc < file1 > file2 2> file3
```

管道 (pipe) :

A进程的输出  B进程的输入

A进程将标准输出重新定向到管道中去;

B进程将标准输入重新定向从管道中来。

例如:

```
ps -e | grep student3 | wc -l
```

查看当前系统中与用户**student3**相关的进程有多少

- **1.4 操作系统服务**

UNIX操作系统提供五种主要的服务（也是**UNIX**核心的五个重要组成部分）：

- 1. 进程管理**

建立、终止、挂起、通信等

- 2. 时钟管理**

分时共享**cpu**，时间片，调度

- 3. 存储管理**

二级存贮器（内存和对换区），分配主存

- 4. 文件系统管理**

文件操作：读、写、更名、拷贝

二级存贮管理：分配和收回存贮区和索引节点

- 5. 设备管理**

对**I/O**设备进行有控制的存取（多进程系统的特征₃）

内核提供的服务的特点：

服务是透明的

①文件类型透明：

用户可不关心是普通文件还是外部设备，但O.S自己要关心文件类型！

②文件系统的透明：

文件系统类型、存放的物理位置。

③存贮方式透明：

文件的存放位置、存放方式、存放格式

④各用户进程能得到核心相同服务：

无论系统程序还是用户程序，平等对待，分时运行

• 1.5 硬件假设

(假设机器硬件只支持的运行状态)

UNIX系统上进程的执行分成两种状态:

用户态、核心态

用户态:

进程正在执行用户代码时的状态

核心态:

进程正在执行系统代码（系统调用）时的状态

用户态和核心态的区别:

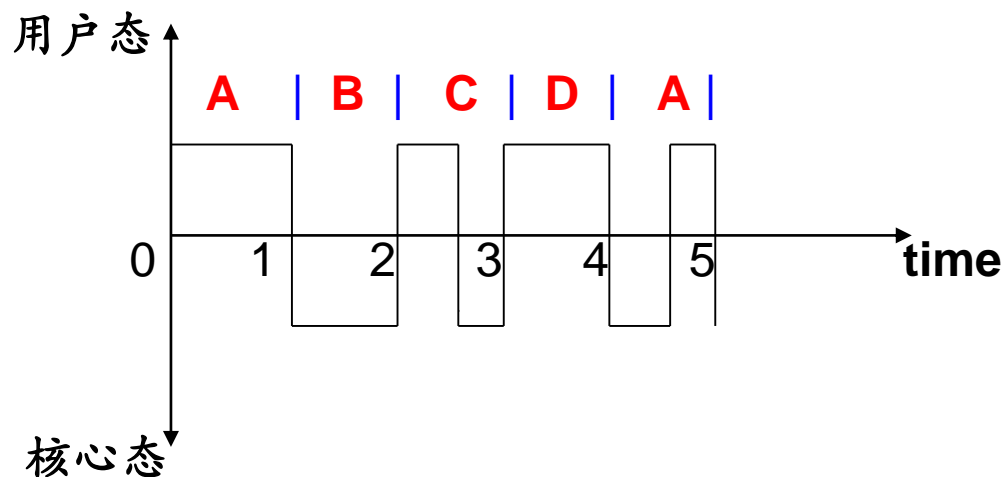
①用户态: 进程只能存取自己的地址空间

核心态: 进程可存取核心和用户地址空间

②用户态: 不能存取特权指令，只能存取自己的指令和数据

核心态: 除了能存取自己的指令和数据外，还可存取特权指令

一个进程在运行时必须处在，而且只能处在或者核心态或者用户态下：



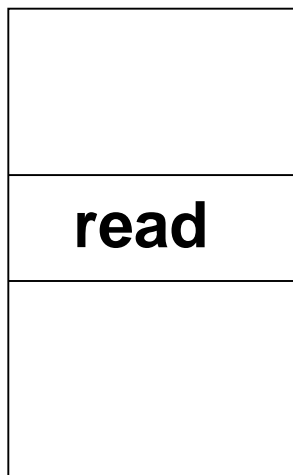
核心态的进程不是与用户进程平行运行的孤立的进程集合，而是每个用户进程的一部分。

“核心分配资源”：

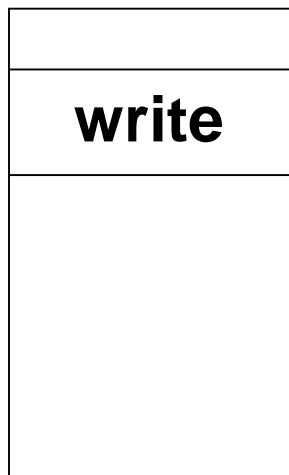
一个在核心状态下执行的进程分配资源。

一个进程某时在“用户态”下运行，另一时刻又在“核心态”下运行，在其生命周期内可能在这两种状态间切换多次

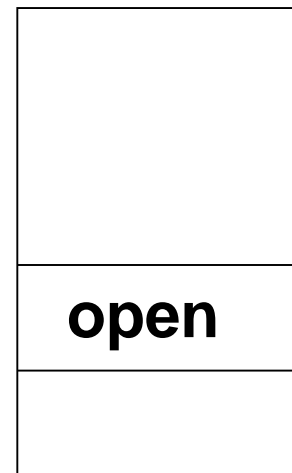
核心——处在核心态下的进程的相应部分的集合



A 进程



B 进程



C 进程

硬件是按核心态和用户态来执行操作的，但对这两种状态下正在执行程序的用户是相同对待的。

1.5.1 中断与例外


- 中断（要保存上下文）：

来自进程之外的事件（外设、时钟等）引起的，发生在两条指令执行之间，中断服务完毕后从下一条指令继续执行。
(中断服务是由核心中特殊的函数，而不是特殊的进程来执行的)

- 例外（不保存上下文）：

来自进程内部的非期望事件（地址越界，除数为0等），发生在一条指令执行过程中，例外事件处理完后重新执行该指令。

1.5.2 处理机执行级

中断事件	中断级别
硬件故障	 高
时钟	
硬盘	
网络	
终端	
软件中断	
	低

用一组特权指令给处理机设置一个执行级，以屏蔽同级和低级的中断，最大限度地减少其它事件的干扰，使当前任务顺利执行并尽快完成；但开放更高级的中断，以响应更紧迫的请求。

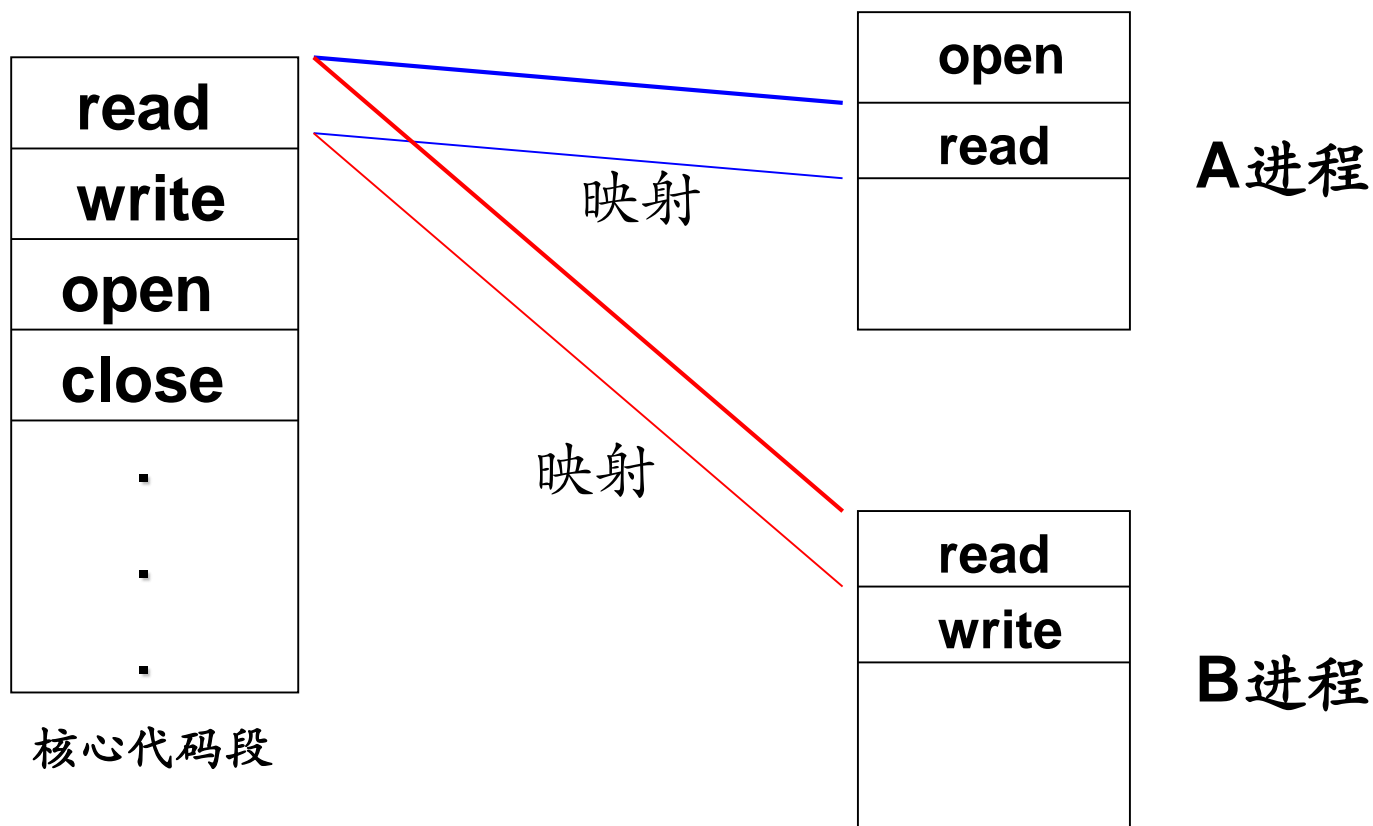
1.5.3 存储管理

UNIX系统中的存储管理原则(或特点):

1. 当前正在执行的进程（全部或部分）驻留在主存中；
2. 核心是永远驻留在主存中的（是永远活动的！）；
3. 编译程序产生的指令地址是虚地址（逻辑地址）；
4. 程序运行时核心与硬件（存储管理部件**MMU**）一起建立虚地址到物理地址的映射。

核心永远是活跃的

普通进程具有特定的生命周期（除非人为设定为无限循环）



只是用户进程中的核心态下运行的代码段常驻内存，而非整个用户进程常驻内存。这些代码段是“可再入段”（或纯代码段、可共享代码段），被各用户进程段共享，为提高运行速度，避免频频访问磁盘，故常驻内存，这些代码段的集合就是**OS**的内核。