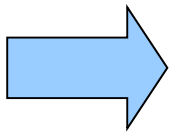
第二章 UNIX操作系统

本章主要内容



1.3 UNIX的发展历史和特点

1.4 UNIX操作系统的功能模块

第二章 UNIX操作系统

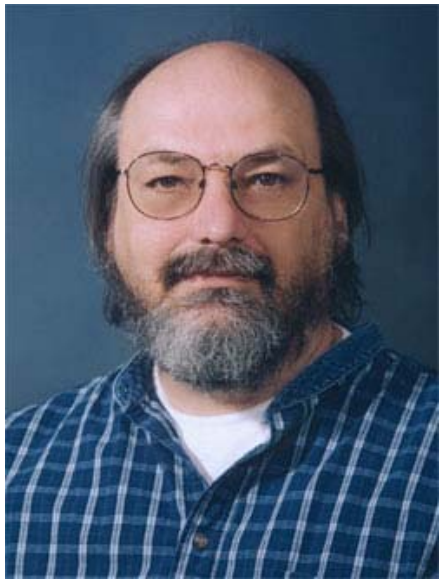
1.3 UNIX的发展历史和特点

- 1.3.1 UNIX的发展历史和特点
- 1.3.2 Linux的发展历史和特点

1.4 UNIX操作系统的功能模块

➢1.3 UNIX的发展历史和特点

- UNIX是一种分时操作系统，是当前工业主流的操作系统之一。
- Linux是一个优秀的类UNIX操作系统，现已成为一个先进和稳定的操作系统，丝毫不逊于商业版的UNIX。



UNIX之父
肯尼思·汤普森
K. Thompson

UNIX之父
丹尼斯·里奇
D. Ritchie

Linux之父
李纳斯·托沃兹
Linus Torvalds

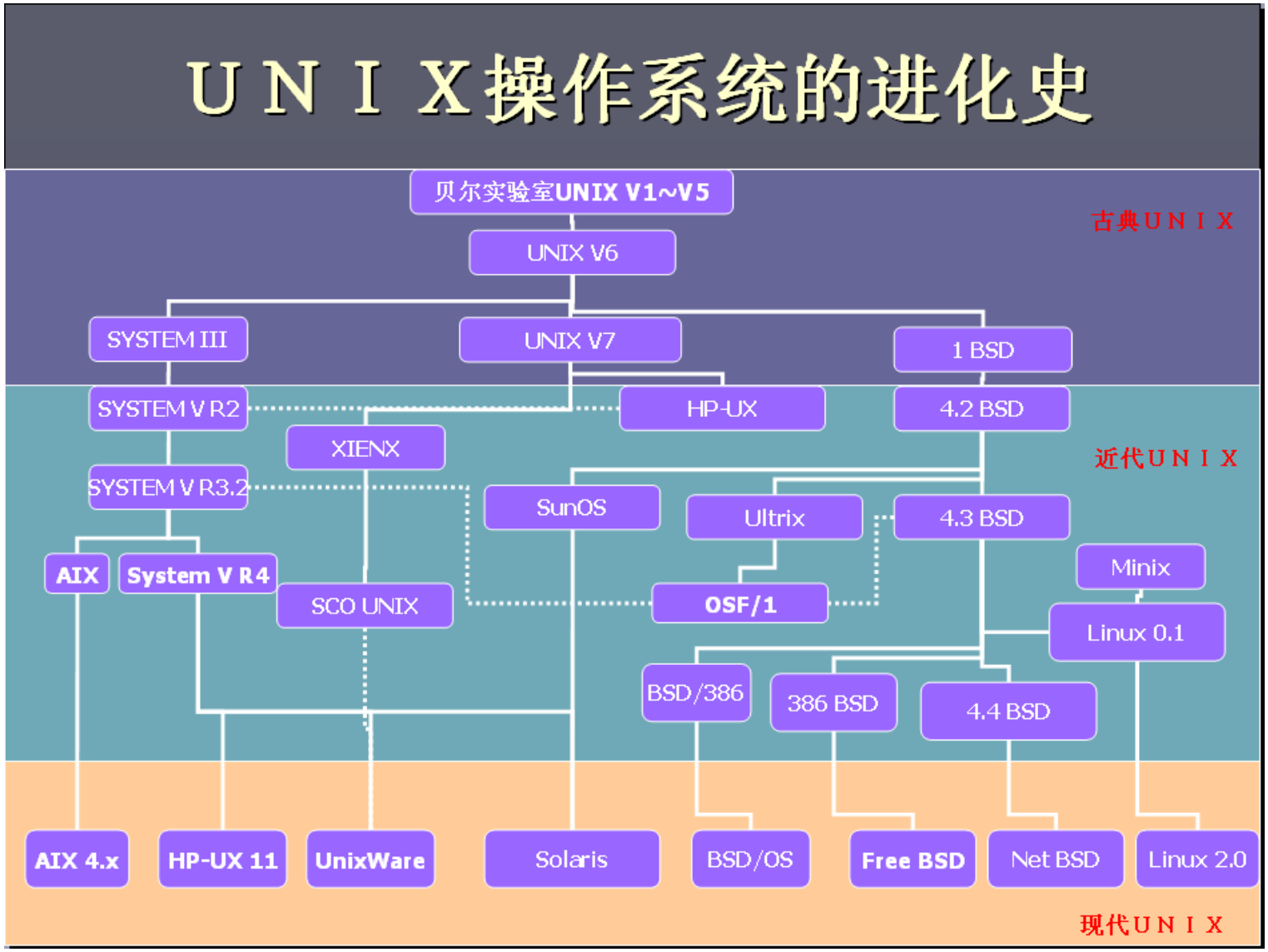
➢1.3.1 UNIX的发展历史和特点

- **1968~1969：**

  Bell实验室的Ken.Thompson & Dennis.Ritchie参与了分时操作系统Multics的设计

- **1970：** **Thompson在PDP-7上用汇编语言实现了UNIX**
- **1971：** **Thompson用B语言改写了UNIX**
- **1973：** **Ritchie用C语言改写了UNIX**
- **1981：** **两大主流版本，由AT&T推出的UNIX System V以及由Berkeley推出的Bsd4.x**
- **1991：** **芬兰赫尔辛基大学21岁的大学生Linus在网上发表了第一个在PC上的UNIX－Linux 0.02版本**
- **1994：** **Linux 1.0版本正式发布，并加入自由软件基金会(FSF) 的GNU工程。**

UNIX操作系统的进化史



➢1.3.1 UNIX的发展历史和特点

UNIX的特点

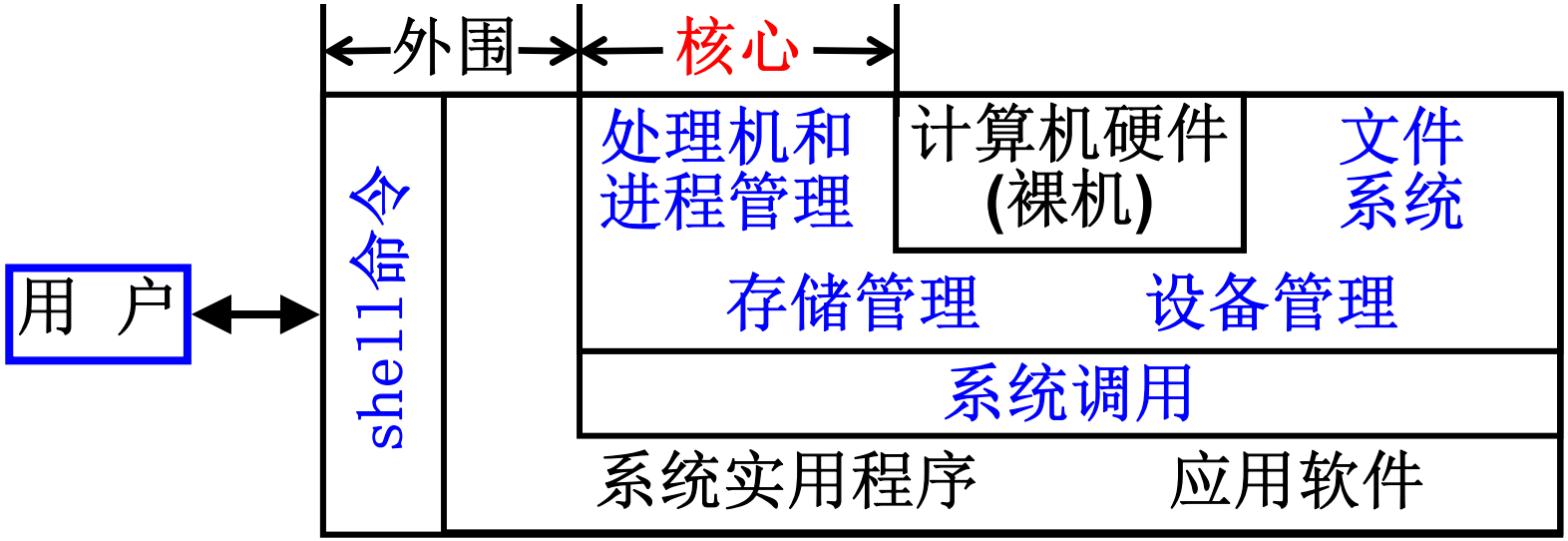
- (1) 在结构上分为核心(Kernel)和外围(Shell)两部分，有机结合。
- (2) 具有良好的用户界面，使用方便，功能齐全，清晰灵活，易扩充和修改，包括操作命令(shell命令)和系统调用。
- (3) 采用树状文件结构
- (4) 统一采用文件来管理普通文件、目录、设备等。
- (5) 具有丰富的语言处理程序，实用程序和工具性软件，是一种开放性的操作系统，提供完备的软件开发环境
- (6) 95%用C语言编程，5%用汇编语言编程，易于移植、理解、修改和扩充
- (7) 多任务系统，多用户、多进程，支持后台作业，能有效的管理CPU和内存资源，提供安全保护
- (8) 具有进程通讯功能

➢1.3.1 UNIX的发展历史和特点

UNIX的系统结构由核心与外围两个部分组成。

➢核心(kernel)

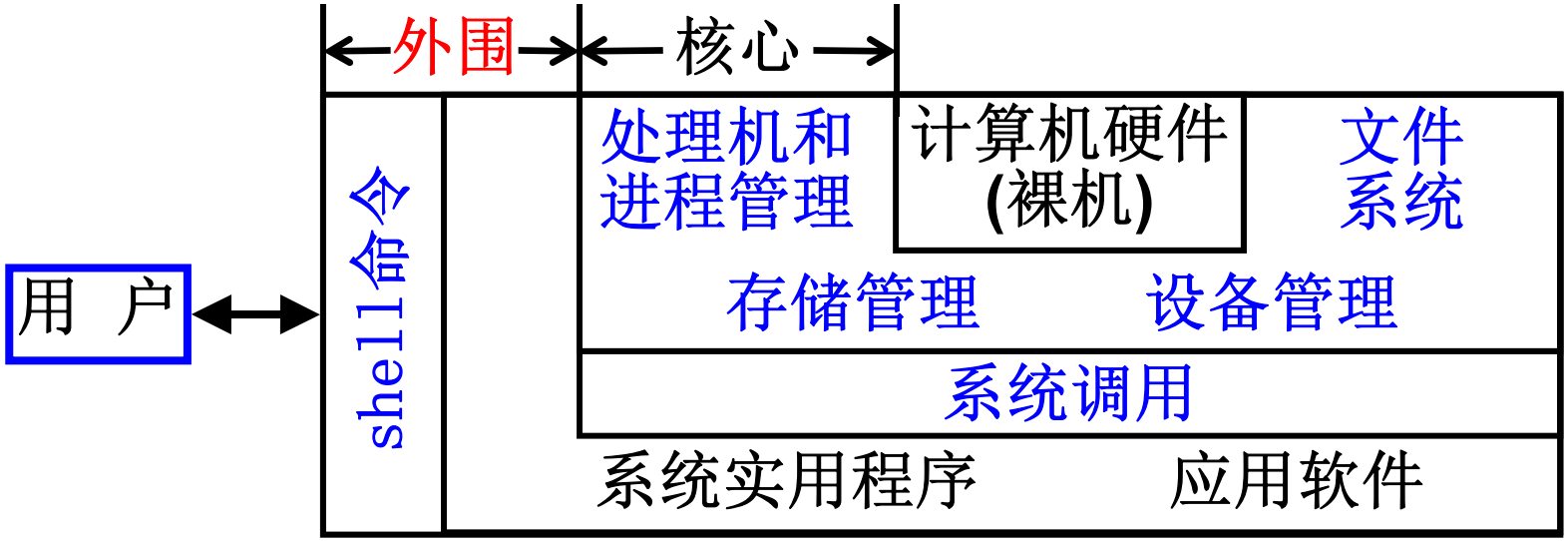
精心设计，简洁精干，只占很小空间，常驻内存。
核心包括处理机和进程管理、存储管理、设备管理和文件系统等四个模块。



➢1.3.1 UNIX的发展历史和特点

UNIX的系统结构由核心与外围两个部分组成。

➢外围(shell)：　用户界面是UNIX的外围。
把系统划分为核心与外围的UNIX结构，其功能模块成为对内和对外两类模块。对内而言，各种软件通过系统调用来使用核心，对外而言，用户可以通过shell来使用计算机。



➢1.3.2 Linux的发展历史和特点

➢Linux与GNU

➢Linux

1991年10月，芬兰赫尔辛基大学21岁的学生李纳斯-托沃兹(Linus Torvalds)在comp.os.minix新闻组上发表了如下消息：

“我正在设计一个(免费的)支持诸如386(486)AT计算机的操作系统(只是业余爱好而已，不会太大也没有GNU那么专业)。”

Torvalds的“业余兴趣”最终变成了现在的Linux操作系统。到90年代末，虽然Linux还只是一种发展不到10年的类UNIX操作系统，但它却引起了PC和因特网世界的革命。世界上数百万计算机用户在他们的家用PC和办公室工作站上使用Linux。

➢1.3.2 Linux的发展历史和特点

➢Linux与GNU

➢GNU

Linux与GNU关系密切。GNU工程由自由软件基金会(FSF)于1984年发起，旨在开发一个类似UNIX，且为自由软件的完整操作系统：GNU系统。

GNU是由“GNU's Not Unix”所递归定义出的首字母缩写，读为gun-noo。GNU认为，以Linux为内核的GNU操作系统应该更精确地被称为GNU/Linux系统。

➢1.3.2 Linux的发展历史和特点

Linux的特点

- (1) 抢占式(独占性)多任务
- (2) 多用户(多路性)
- (3) 设备无关性
  将所有外部设备统一视作文件来处理。
- (4) 开放性
  遵循世界标准规范，符合TCP/IP网络协议。很容易和其他系统互操作，也可以作为服务器。

➢1.3.2 Linux的发展历史和特点

Linux的特点

- (5) 高可扩展性、可维护性与开放源代码

- (6) 优异的网络性能

  Internet是在UNIX领域建立的。Linux通过免费提供大量具有强大网络功能的Internet网络软件，在通讯和网络功能方面明显优于其他操作系统。

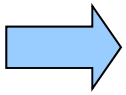
- (7) 可靠的系统安全

- (8) 良好的可移植性(portability)

  Linux是一种可移植的操作系统，尤其是可移植到计算机工作站以及PC机。

第二章 UNIX操作系统

1.3 UNIX的发展历史和特点



1.4 UNIX操作系统的功能模块

➢ 操作系统的功能模块

➢ 操作系统管理的内容

硬件资源：　　**CPU, Memory, STORAGE, I/O**

软件资源：　　**OS本身，系统实用程序及应用软件等，用户的程序及数据 (均以文件方式存放在软硬盘等介质之中)**

➢ 操作系统的功能模块

- ➢处理机和进程管理模块
- ➢存储器管理模块
- ➢设备管理模块
- ➢信息管理模块(又称文件管理,文件系统)
- ➢用户界面(又称用户接口)

第二章 UNIX操作系统

1.3 UNIX的发展历史和特点

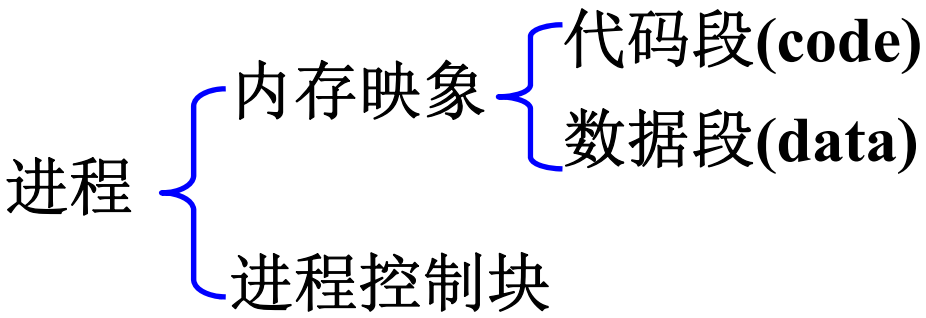
1.4 UNIX操作系统的功能模块

- 1.4.1 处理机和进程管理模块
- 1.4.2 存储管理模块
- 1.4.3 设备管理模块
- 1.4.4 文件系统
- 1.4.5 用户界面

➢1.4.1 处理机和进程管理模块

➢ 进程的构成

在Unix/Linux中，一个进程由内存映象和进程控制块(PCB，Process Control Block)组成。



➢1.4.1 处理机和进程管理模块

➢Unix进程调度

- ➢进程分类以及设定进程优先级
  系统进程—核心态(特权进程)
  用户进程—用户态(普通进程)
  系统进程比用户进程的优先级高

- ➢进程优先级的控制
  显示或改变进程的优先级： **nice**命令

- ➢动态优先的进程调度算法
  根据占有**CPU**的时间来动态改变优先级，可使各进程的响应时间较均匀。

➢进程的状态控制

➢支持前台进程和后台进程

在某一个时刻，用户在一个终端上只能运行一个前台进程，而可以运行多个后台进程。

➢产生后台进程

如果**a.out**是一个比较费时的进程，而没有作为后台进程运行。例如：

$ a.out 将**a.out**作为前台进程运行

此时将处于等待的状态，不管**a.out**是否产生显示结果，必须等到结束后才能运行其他命令。

假定需要把**a.out**作为后台进程运行，则应该在命令行后加**&**。例如：

$ a.out & 将**a.out**置为后台进程

[1] 26695 显示后台进程的进程号

$

此时将显示后台进程的进程号以及待命符，表示**a.out**进入后台可以启动其他命令。

➢进程的状态控制

➢将前台进程转入后台运行

如果运行**a.out**后，发现是一个比较费时的进程，一直在等待。如果希望在其结束之前启动其他命令，则首先应该键入控制符**^Z**，使其暂停，然后使它成为后台进程。例如：

$ a.out	将**a.out**作为前台进程运行
此时将处于等待的状态	
^Z	在**a.out**还没有结束时暂停该进程
$ bg	将其转为后台进程
[1] 26695	运行后台进程并且显示进程号
$	显示待命符，表示可启动其他命令

➢进程的状态控制

➢将后台进程转入前台运行

如果需要将处于后台的进程转为前台进程，则应该使用**fg**命令：

$ fg　　将后台进程转入前台运行

此时不会显示待命符，而转向执行**a.out**。在**a.out**结束运行之前，不能启动其他命令。

➢进程的状态控制

- ➢终止当前进程　<CTRL>C(或者 ^C)
- ➢暂停当前进程　<CTRL>Z(或者 ^Z)
- ➢显示进程信息

```
$ ps
PID(进程号)   TTY(终端名)  TIME    COMMAND
97          /dev/thy2     23:16   sh
126         /dev/thy2     10:04   a.out
```

- ➢终止进程命令kill

命令格式:

$ kill -9 *进程号*　　kill命令的参数-9表示强行中止进程

例如:

$ kill -9 126　表示强行终止a.out(进程号为126)的运行

➢1.4.2 存储管理模块

➢ 功能：

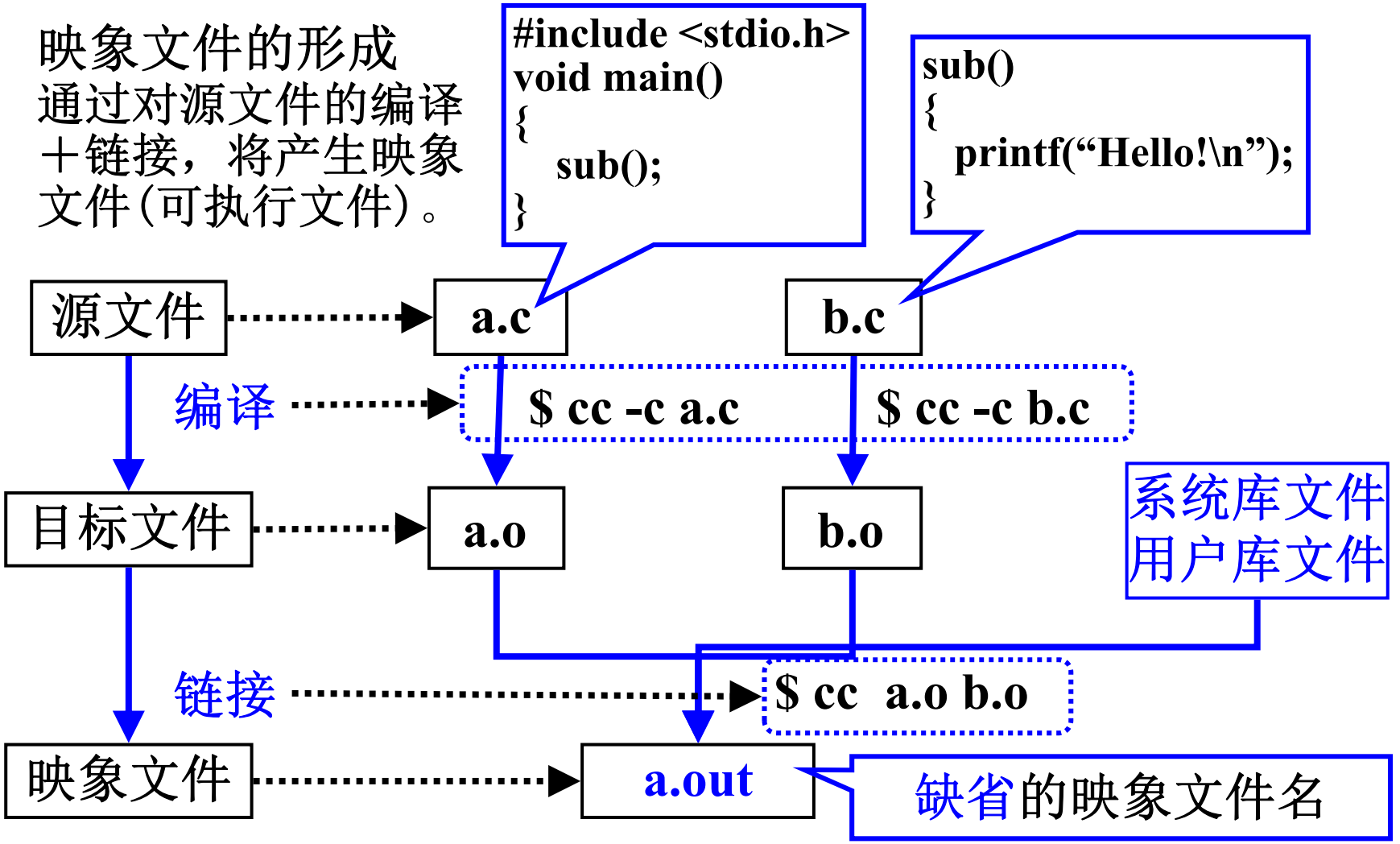
用于确保所有进程都能安全的共享主内存区，同时，支持虚拟内存管理方式，使得Linux支持进程使用比实际内存空间更多的内存容量。

存储管理模块的主要任务

<1>存储分配　　<2>地址映射　　<3>内存保护

➢1.4.2 存储管理模块

映象文件的形成
通过对源文件的编译＋链接，将产生映象文件(可执行文件)。



如果在链接时不指定映象文件名，例如：

$ cc a.o b.o　　生成缺省的映象文件a.out

➢存储分配

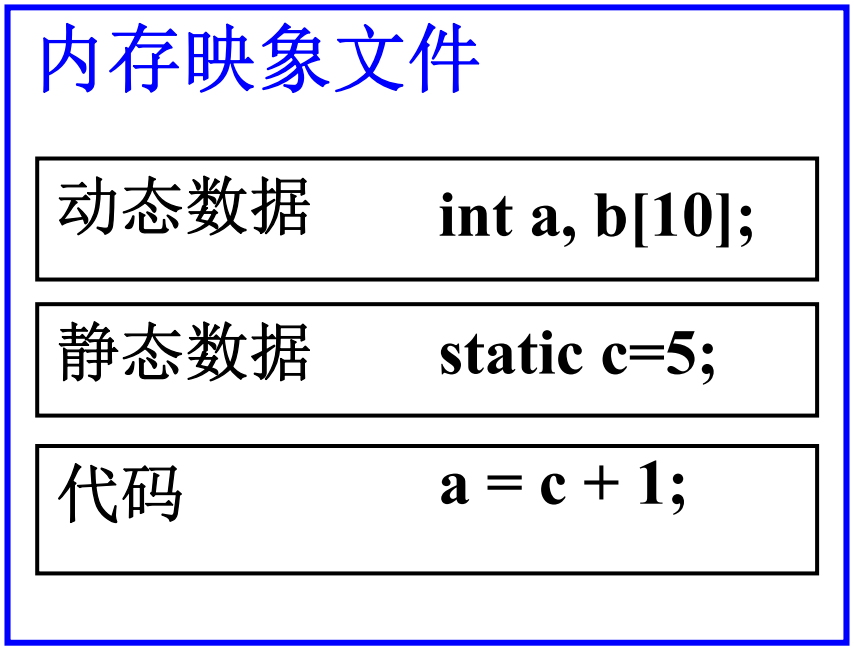
➢进程所需内存

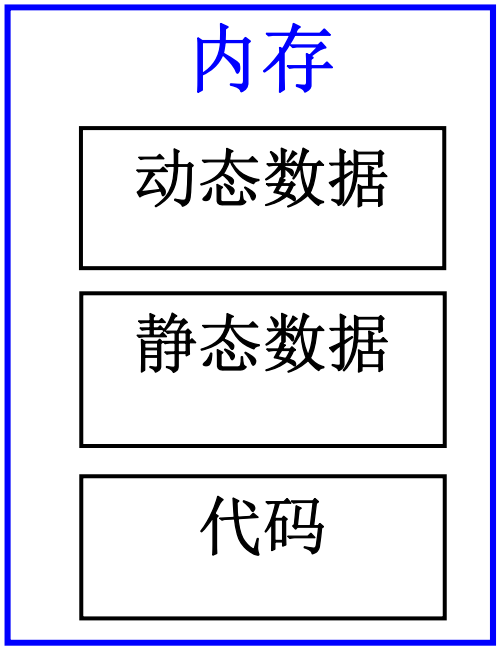
映象文件描述了进程所需的内存大小，但不等于内存的大小。
显示映象文件的大小：

$ ls -l a.out

显示所需内存的大小：

$ size a.out　　通常，映象文件比所需内存的空间要小

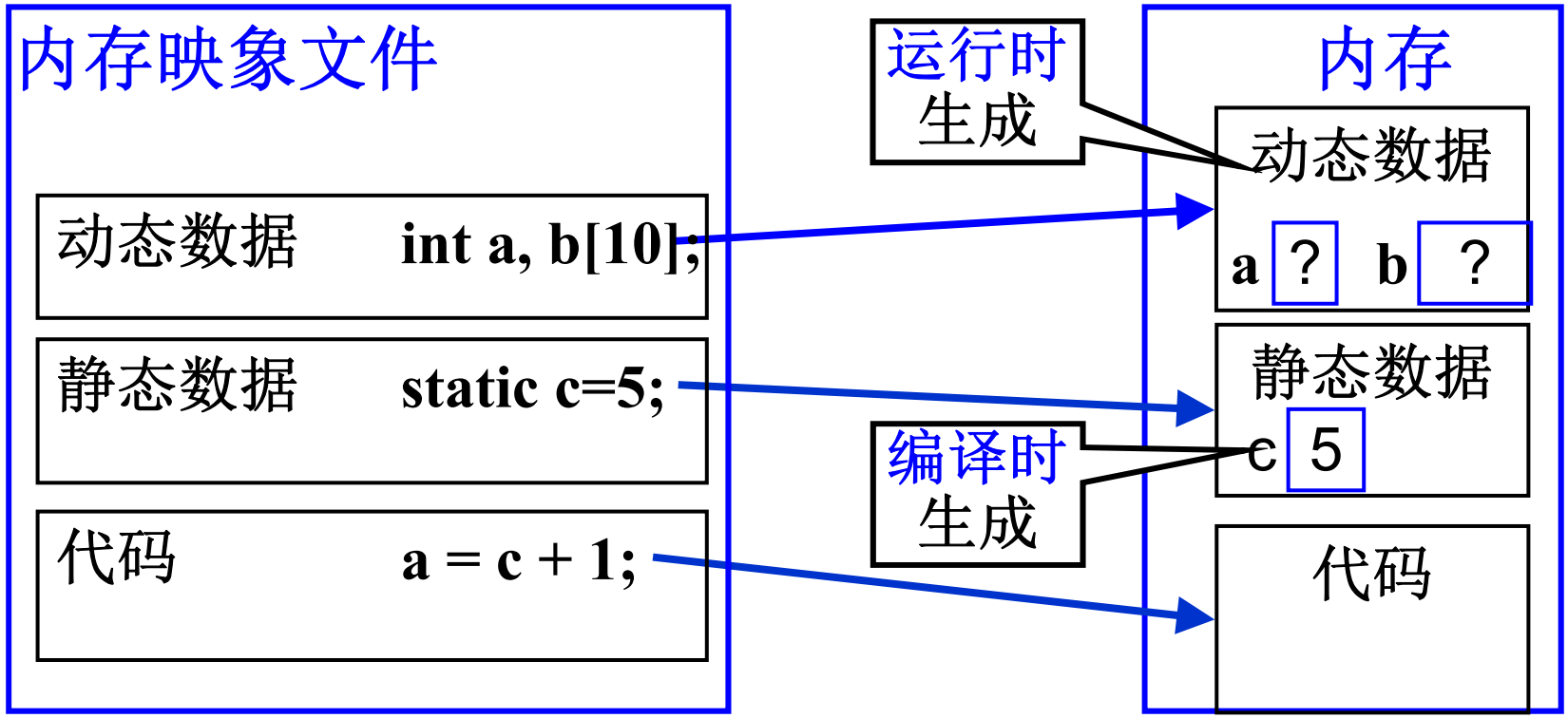




➢存储分配

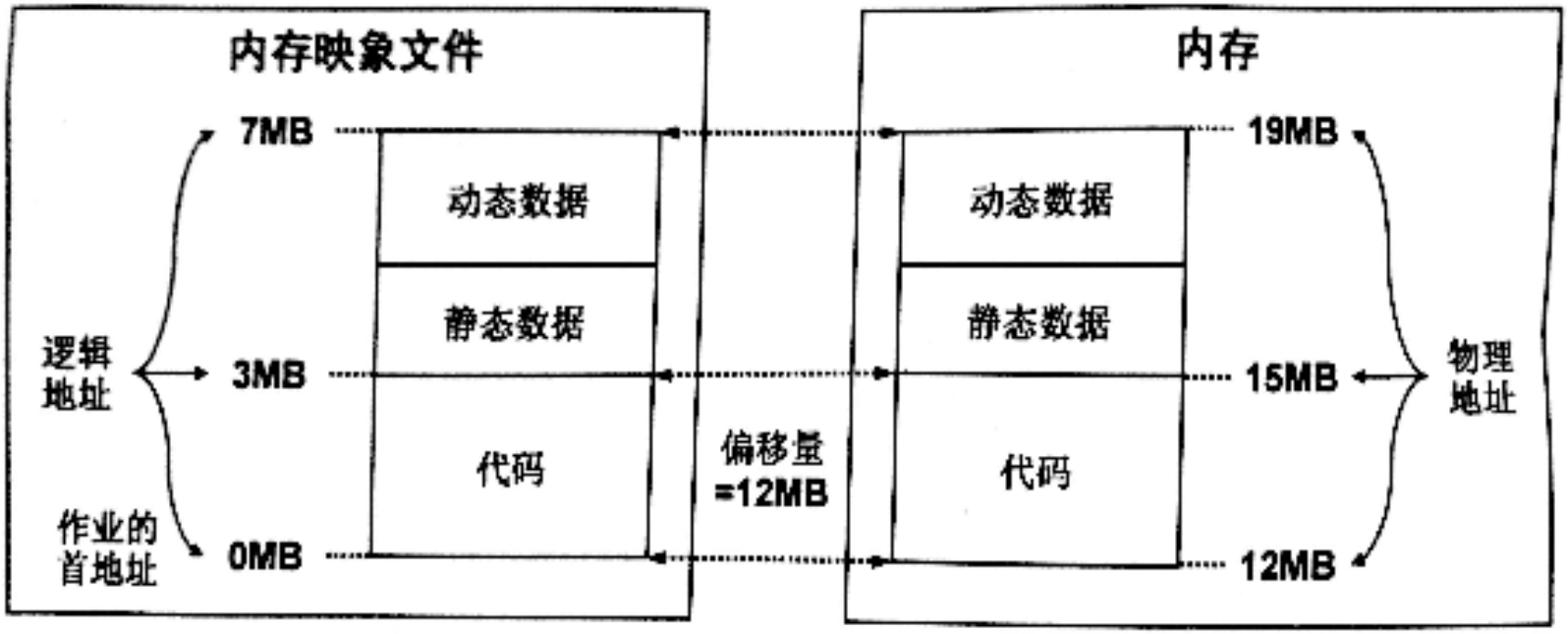
➢映象文件装载到内存的过程

- ➢首先确定内存中数据区的大小。
- ➢对于静态数据，不仅必须确定所需的空间，还必须完成静态数据的初始化，即在数据区中完成赋初值。
- ➢对于动态数据，计算所有动态数据可能需要的最大空间。也就是说，动态数据区将被各种动态数据共享。



➢地址映射

➢计算逻辑地址到物理地址的转换工作称为地址映射



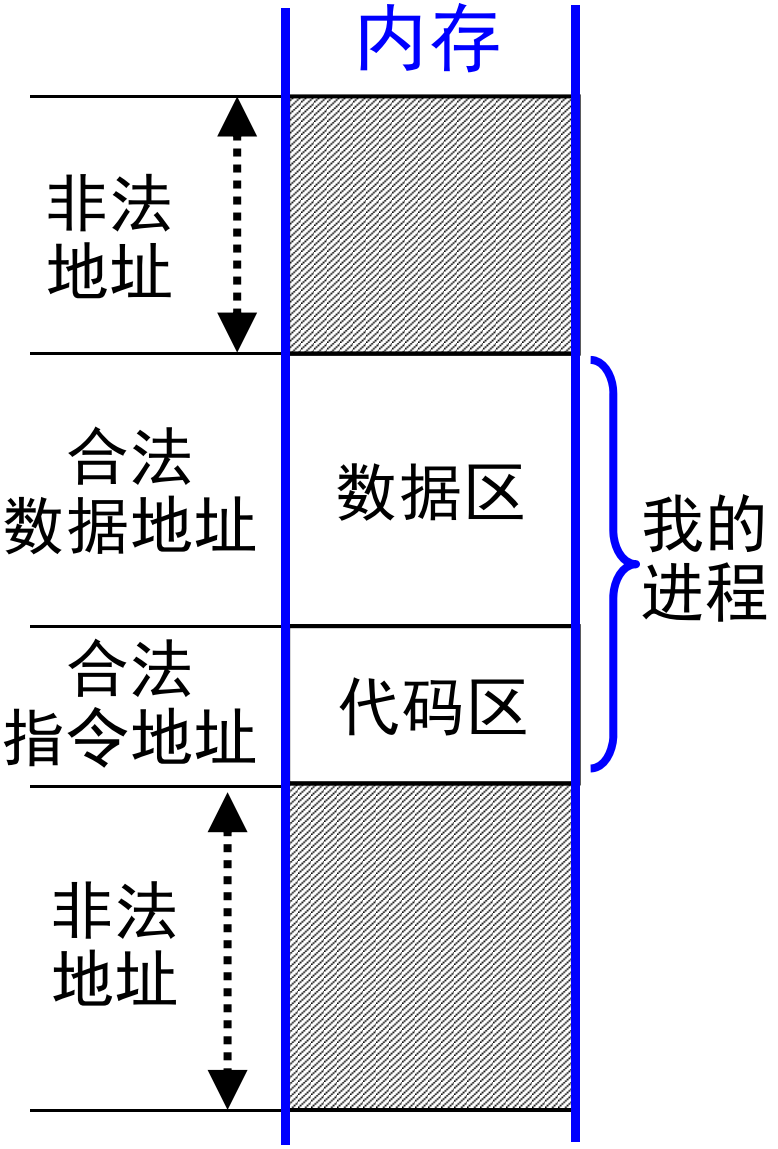
➢内存保护

➢禁止访问非法地址

➢每个进程在内存中的区域是该进程可以访问的合法地址。

➢每个进程可以执行指令的合法地址必须落在该进程的代码区内。

➢每个进程可以访问数据的合法地址必须落在该进程的数据区内。

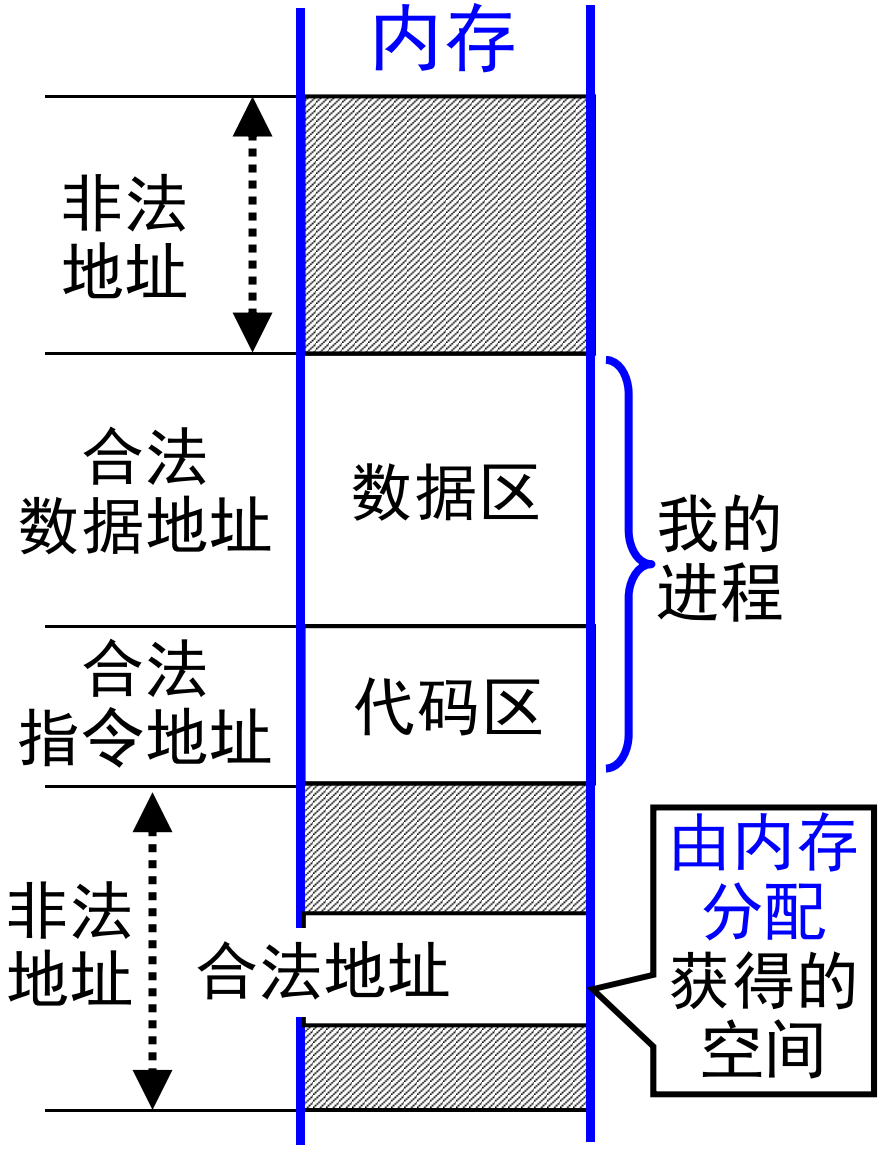


➢内存保护

➢动态申请的数据存储空间

使用动态内存分配所获得的数据空间，将位于进程的内存区域之外，这也是该进程可以访问的合法数据地址。

因此，每个进程可以访问数据的合法地址将包括该进程的数据区以及动态分配的数据空间。如果进程要求到这些区域存取数据的话，内存保护模块将认为是合法的。



➢1.4.3 设备管理模块

- 设备管理模块的管理对象
  - ➢文件的存储介质

    硬盘，软盘，磁带，光盘等
  - ➢I/O设备

    字符终端，图形终端，打印机，绘图仪等
  - ➢专用I/O设备

    数据采集仪，扫描仪，图像摄入装置，音频I/O设备等
- 设备管理模块的管理任务
  - 设计一定的分配策略，把相应的设备分配给提出请求的进程，并启动该设备，完成数据的传递和I/O操作

➢1.4.3 设备管理模块

设备分类

UNIX把设备分为块设备和字符设备两类。

➢块设备

- ➢用于存储信息，如通常的 硬盘，软盘，光盘，磁带等。
- ➢数据按块传输，速度快。
- ➢块设备称为共享设备，可供多个进程同时访问。
- ➢块设备可按随机存取方式(或称直接存取方式),则在块设备中将包含相同的目录结构。访问前要装配(mount命令)该设备，适用于硬盘，软盘，光盘等设备，也是最常用的情况。

$ mount /dev/hd1 /mnt **/dev/hd1是硬盘设备文件名**

$ umount /dev/hd1 **umount是卸装设备命令**

➢1.4.3 设备管理模块

设备分类

UNIX把设备分为块设备和字符设备两类。

➢字符设备

- ➢字符设备通常指用于数据的I/O设备，例如终端设备，打印机等。
- ➢数据按字符传输，速度慢。
- ➢字符设备称为独占性设备，不能有多个进程同时访问，而是按照先后顺序给予访问。例如，有两个进程都要求打印数据，则必然是顺序完成输出。同样，如果有两个进程都要求将数据在某个窗口显示，则也将获得顺序显示的效果。

➢1.4.4 文件系统

➢定义：

一组相关信息的集合称为文件

➢ 文件系统的功能

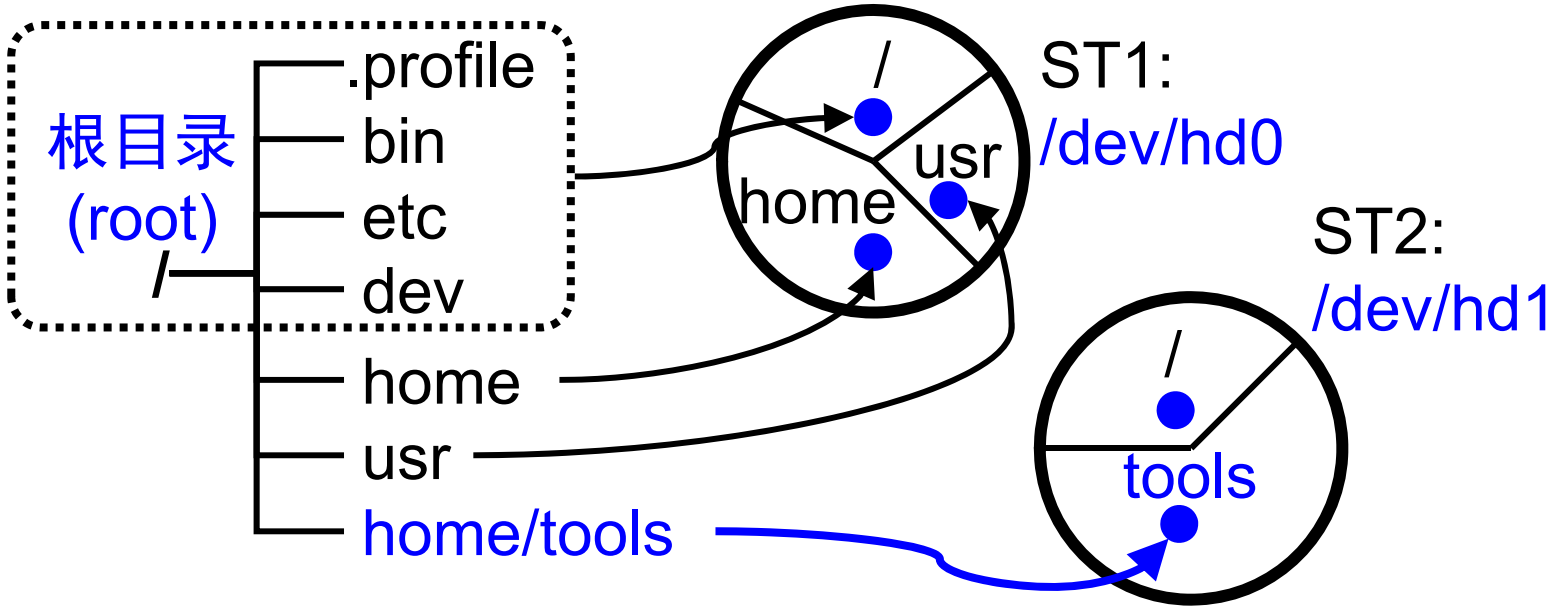
负责对软件资源的管理，又称为文件管理和文件系统。所有的软件资源都是以文件形式存放在介质中，以文件为单位，在计算机中传递信息。

➢任务

- ➢建立文件的存储检索
- ➢管理文件的空间分配
- ➢建立文件的共享，保护等机制
- ➢对用户提供文件操作的界面

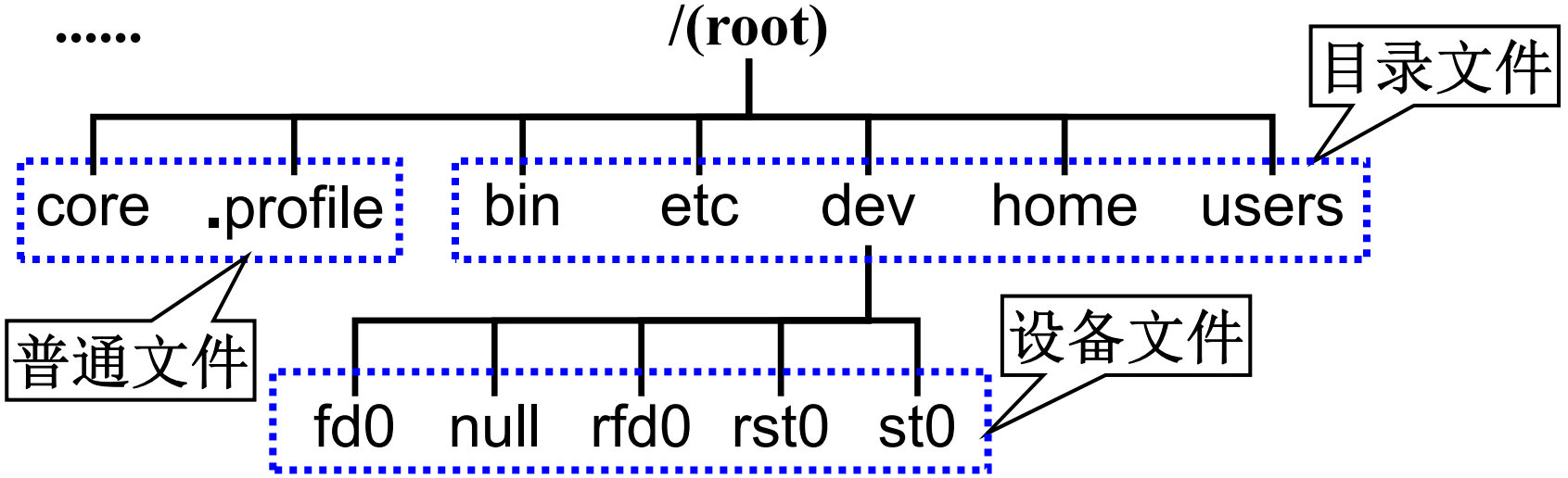
➢树状结构文件系统

UNIX提供了完整的树状结构文件系统。即在一个根目录“/”之下可以访问各级目录，不仅仅包括主盘中的目录，而且包括副盘中的目录，还可以访问网络上其他计算机(远程宿主机)的硬盘目录。



➢UNIX的标准目录结构

/bin 二进制(可执行文件)目录
/etc 系统杂类文件目录
/dev 设备文件目录
/lib 库函数目录
/tmp 临时文件目录
/usr 客户的系统文件目录
/usr/bin /usr/etc /usr/lib ... 客户系统文件的有关目录
/users 用户目录
/home 用户注册目录的父目录
.....



➢UNIX的文件管理机制

UNIX将普通文件、目录、设备等统一用文件来管理。

➢普通文件(file，或plain file)

普通文件，简称文件，存放ASCII或者Binary等各类数据，或者存放有格式或者无格式数据的各类文件。

例如：.profile，a.txt，a.out，/tmp/core，等等。

➢目录文件(directory)

目录也是一个文件，即目录文件。记录该目录下的文件信息，包括该目录下文件的名称、大小、所有者、访问权限、创建和修改时间、文件链接信息，以及该目录下文件的总数和总容量等等。

例如：/bin，/etc，/dev，/usr/bin，/usr/etc，等等

➢UNIX的文件管理机制

UNIX将普通文件、目录、设备等统一用文件来管理。

➢设备文件(device)

例如：/dev/fd0, /dev/mem, /dev/null(空设备,垃圾筒), 等等

其中，/dev/null称为空目录，如果将普通文件或者目录文件移到该目录下，相当于删除文件或者目录。例如：

```
$ mv a.out /dev/null    等价于
$ rm a.out
```

➢UNIX的文件管理机制

➢文件类型的特征显示

使用ls -l命令时，可看到在列表命令显示结果的最左一列是表示文件的保护模式的，而首字符用来表示文件的类型，例如：

-rwxr--r--　　“-”表示普通文件
drwxr--r--　　“d”表示目录(目录文件)
brwxr--r--　　“b”表示块设备文件(特殊文件)
crwxr--r--　　“c”表示字符设备文件(特殊文件)

➢隐含文件信息的显示

首字符为.的文件称为隐含文件，必须用ls -a显示其信息。假定当前目录下有文件.profile和a.txt，则有：

```
$ ls
a.txt                                   无法显示隐含文件
$ ls -a                                 要求显示隐含文件
.      ..      .profile      a.txt      .是当前目录，..是父目录
```

➢文件和路径

一个文件及其从根目录到所在目录的全称，称为路径(path)。
例如， /users/proj/liming/src/a.out和/usr/bin均称为路径。

其中，第一个斜杠“/”称为根目录(root)，此后的/是各级目录的分隔符，a.out是文件, 而users, proj, liming, src, usr和bin都是目录。

➢当前目录和注册目录

- 用户当前所在目录称为当前目录(current directory)，又称当前路径(current path)。
- 用户的注册目录(Home directory)是指用户登录(注册)操作系统成功后所在的目录，也可称为用户的根目录(Root directory)。

- 文件和路径
  - 当前目录和注册目录
    - 用**pwd**命令(**print current directory**)可以显示用户的当前目录。环境变量**HOME**表示用户的注册目录。假定用户的当前目录是**/users/proj**，用户的注册目录是**/home/tom**，可以操作如下：

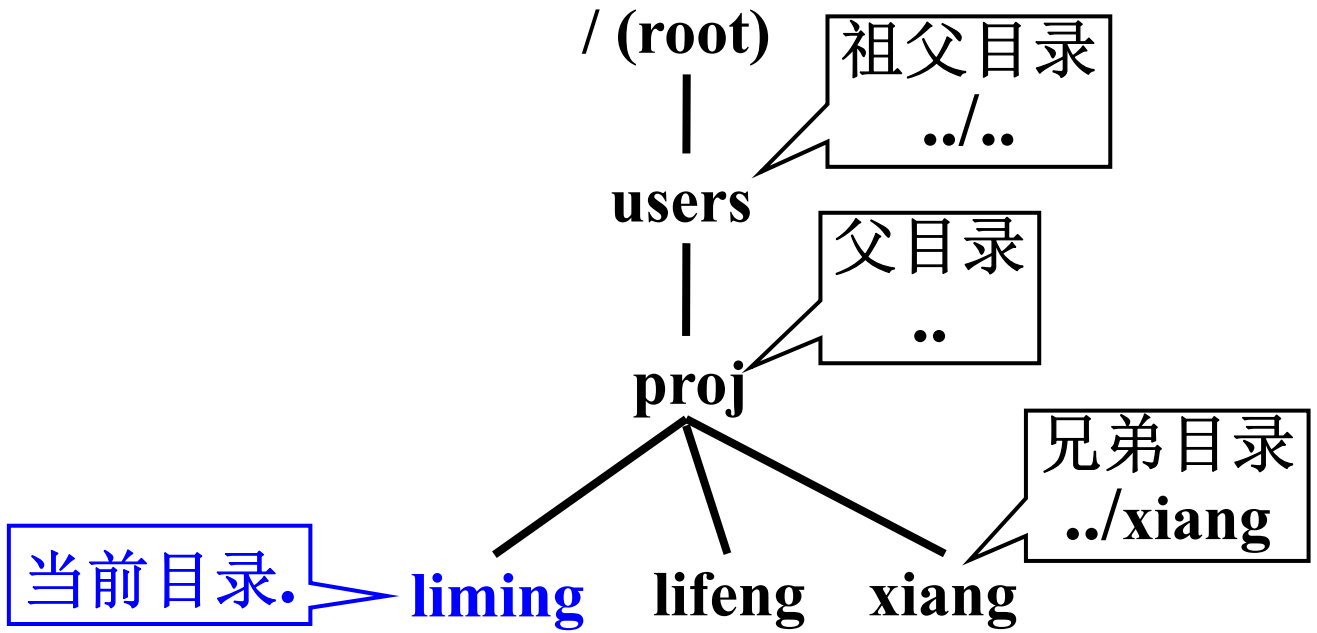
$ echo $HOME 显示注册目录(环境变量**HOME**的值)
/home/tom
$ pwd 显示当前目录
/users/proj
$ cd $HOME 改变当前目录到注册目录
$ pwd 显示当前目录
/home/tom 注册目录成为当前目录

- 文件和路径
  - 目录(路径)的相对表示

在UNIX中，采用当前目录的相对路径来表示其他目录(路径)。

假定当前目录在/users/proj/liming，可以用.表示。

- /users/proj是当前目录的父目录，可用..表示。
- /users是当前目录的祖父目录，可用../..表示。
- /users/proj/xiang是当前目录的兄弟目录，可用../xiang表示。



➢文件和路径

➢目录(路径)的相对表示

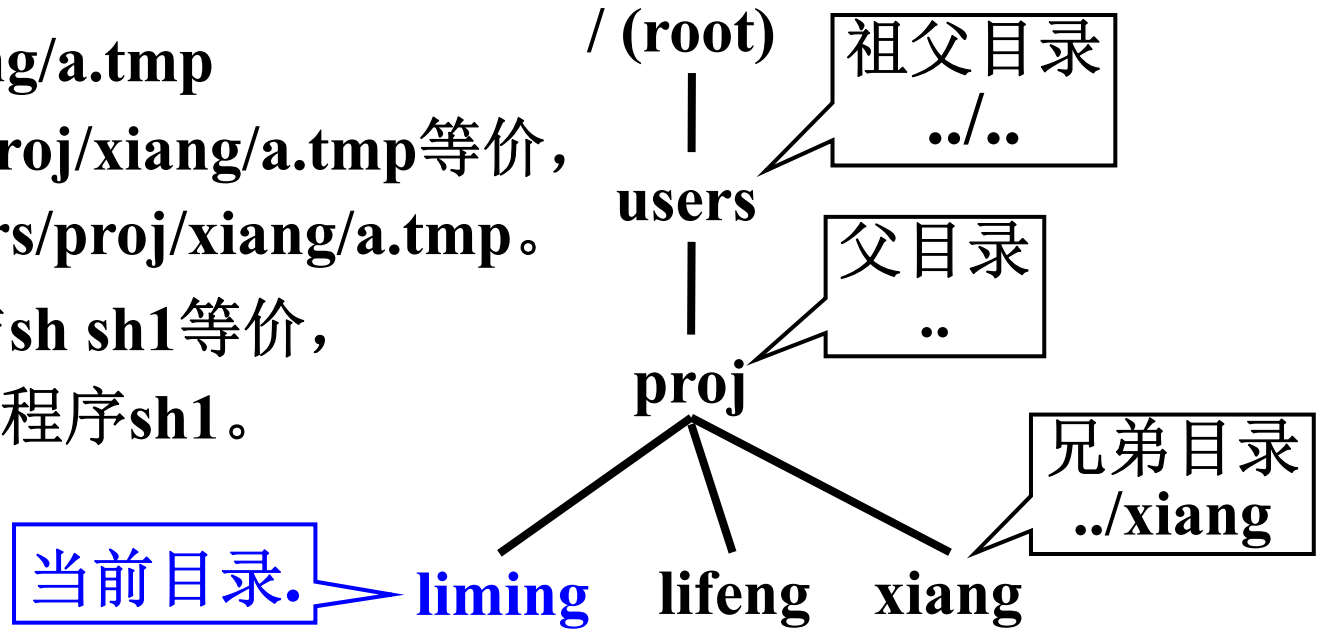
其他可以类推。例如：

1) 命令cat ../a.c与cat /users/proj/a.c等价，
表示显示文件/users/proj/a.c的内容。

2) 命令cd ../..与cd /users等价
表示将当前目录改为/users 。

3) 命令rm ../xiang/a.tmp
与rm /users/proj/xiang/a.tmp等价，
表示删除/users/proj/xiang/a.tmp。

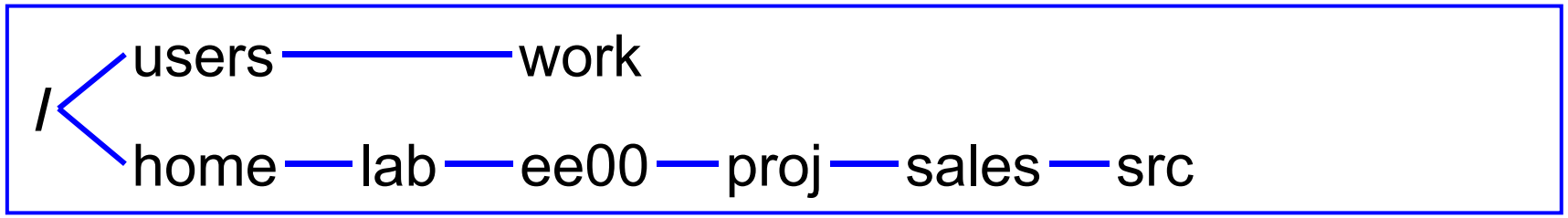
4) 命令sh ./sh1与sh sh1等价，
表示执行shell程序sh1。



➢文件系统的符号链接

在**UNIX**的树状文件系统结构中，允许通过符号链接把某个目录或者文件挂到另一个目录之下。

假定**/home/lab/ee00/proj/sales/src**是一个路径，**src**是一个比较深的目录。

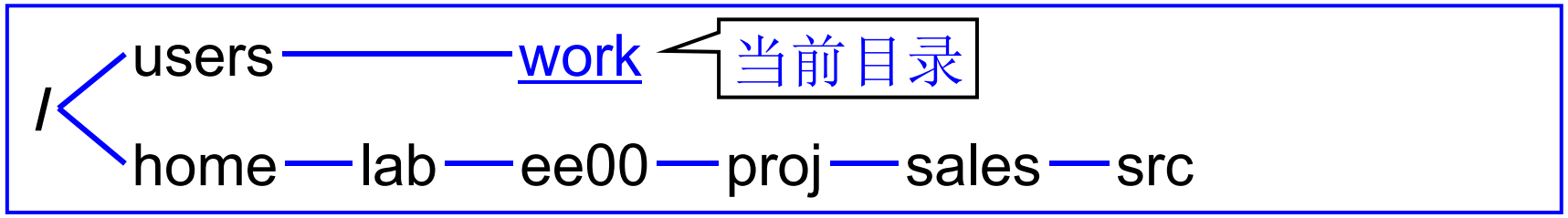


如果希望从一个经常使用的目录**/users/work**转到**src**，命令为：

$ cd /home/lab/ee00/proj/sales/src

显得操作比较麻烦，则可以通过符号链接事先把目录**src**挂到目录**work**之下，从而可以简化操作。例如：

$ cd /users/work　　改变目录到**/users/work**下**(**进入**work)**

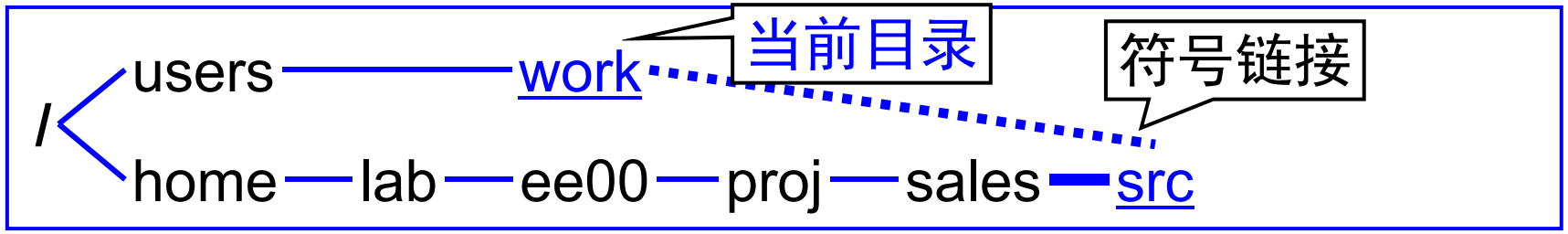


➢文件系统的符号链接

建立符号链接，将目录/home/lab/ee00/proj/sales/src链接到/users/work下，生成一个符号链接的目录/users/work/src。操作为：

$ ln -s /home/lab/ee00/proj/sales/src　　或者

$ ln -s /home/lab/ee00/proj/sales/src src



从目录src向上看，用实线相连的sales是它的父目录，用虚线相连的work是它的符号链接父目录。

```
$ pwd                                    显示当前目录
/users/work
$ ls -l                                  显示文件信息
lrw-r--r-- 1 ee00 users 154 Dec 26 1997 src -> /home/lab/
ee00/proj/sales/src
```

其中，l表示src为链接文件

通过符号链接，可以从work直接进入src，例如：

```
$ cd src                                 从work进入目录src
```

➢文件保护模式

➢用户(user)分类

u：	**user(owner)**	文件主人
g：	**group**	与文件主人同组的用户
o：	**others**	与文件主人不同组的其他用户

➢访问类型(access)分类

r：	**read**	表示可读
w：	**write**	表示可写
x：	**executable**	表示可执行
-：	**no access right**	表示无权访问

➢文件保护模式

➢对于文件

可读**readable**(文件可读)：　　取值：**r** (Y) 或者 **-** (N)

可写**writable**(文件可修改)：　　取值：**w** (Y) 或者 **-** (N)

可执行**executable**(文件可执行)：　取值：**x** (Y) 或者 **-** (N)

➢对于目录

可读**readable**(目录可列表)：　　取值：**r** (Y) 或者 **-** (N)

可写**writable**(文件可增删改名)：　取值：**w** (Y) 或者 **-** (N)

可执行**access able**(目录可进入)：　取值：**x** (Y) 或者 **-** (N)

➢**UNIX**中的超级用户**root**不受文件保护模式的限制

➢文件保护模式

➢保护模式的字符表示形式

对于三类用户，每一类都用三个字符来表示文件的保护模式，其中用r、w和x表示Y，用-表示N。

user:　　owner　group　others　全部称为all

access:　　r w x　r w x　r w x　共9个字符

例如：

$ ls -l myfile project

-rwxrwxrwx 1 eejm000 users 154 Dec 26 1997 myfile

drw-r- - - - - 1 eejm000 users 154 Dec 26 1997 project

表示所有用户都能够读、写和执行文件**myfile**。

对于目录**project**，**owner**可以读和写，不能执行；**group**可以读，不能写和执行，**others**不能读、写和执行。

➢文件保护模式

➢保护模式的内部八进制表示

文件系统的内部用三位八进制数来表示文件的保护模式，每一位为1表示Y，为0表示N。

例如：**rwxrwxrwx**的八进制值为**777**，

rw- r- - - - -的八进制值为**640**，**rw-rw-r--**的八进制值为**664**，

rwx- - - - - -的八进制值为**700**。

➢文件保护模式

- 修改保护模式命令**chmod(change mode)**

  命令格式：**chmod [*用户*] [=|+] [*模式*] *文件* ...**

  其中，*用户*的可取值为：

u：user(owner)	文件主人
g：group	与文件主人同组的用户
o：others	与文件主人不同组的其他用户
a：all	所有用户

➢文件保护模式

- 修改保护模式命令**chmod**(change mode)

  命令格式：**chmod [*用户*] [=|+] [*模式*] *文件* ...**

例如：

- **$ ls -l sh1**

  -rw-r--r-- 1 eejm000 users 154 Dec 26 1997 sh1

- **$ chmod 700 sh1**　　文件sh1禁止非owner用户访问
- **$ ls -l sh1**

  -rwx------ 1 eejm000 users 154 Dec 26 1997 sh1

- **$ chmod a+x sh1**　　或者
- **$ chmod +x sh1**　　允许所有用户可执行文件sh1
- **$ ls -l sh1**

  -rwx--x--x 1 eejm000 users 154 Dec 26 1997 sh1

➢文件保护模式

- ➢ 修改保护模式命令**chmod**(change mode)

  命令格式：**chmod** [*用户*] [=|+] [*模式*] *文件* ...

  注意在UNIX和Linux的使用上的差异，例如：

  - ➢ 在命令chmod中使用“+x”时，对于UNIX，

    chmod +x 等价于 **chmod a+x**

  - ➢ 而对于Linux，

    chmod +x 等价于 **chmod u+x**

➢1.4.5 用户界面

- ➢ 任务：向用户提供使用OS的手段
- ➢用户界面shell
  - ➢ 什么是shell

    shell是一个用C语言编写的解释性程序(解释器)。
    shell是用户使用UNIX/Linux的桥梁。
    shell既是一种命令语言，又是一种编程语言。
  - ➢ 作为命令语言

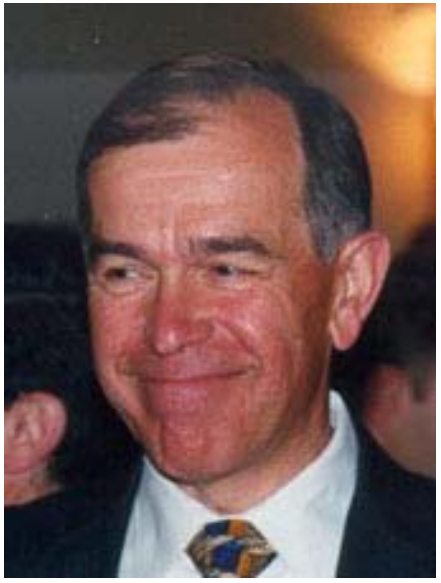
    以交互的方式解释和执行用户的输入命令。
  - ➢ 作为编程语言

    用shell语言编写的shell程序又可称为脚本，可定义各种变量和参数，并提供了许多在高级语言中才有的控制结构。

➢1.4.5 用户界面

➢shell的种类

B shell、C shell、K shell、Bash shell、Tc shell



B shell作者
史蒂夫•伯恩
Steven Bourne

C shell作者
比尔•乔伊
Bill Joy

K shell作者
大卫•科恩
Dave Korn

操作系统的功能模块

- 操作系统管理的内容

  硬件资源：　CPU, Memory, STORAGE, I/O

  软件资源：　OS本身，系统实用程序及应用软件等，用户的程序及数据 (均以文件方式存放在软硬盘等介质之中)

- 操作系统的功能模块
  - 处理机和进程管理模块
  - 存储器管理模块
  - 设备管理模块
  - 信息管理模块(又称文件管理,文件系统)
  - 用户界面(又称用户接口)

一些Unix/Linux命令

- bg与fg（ ^Z与^C ）
- ps与kill
- ls与size
- mount与umount
- mv与rm
- pwd
- cd
- cat
- more与less
- echo
- sh、csh、bash
- ln
- chmod

  文件保护模式：r w x -
- man

➢作业

- 习题 **E-1　1-5，1-6，1-7，1-12，1-13.1**
- 上机操作并巩固课堂上介绍的**Linux**命令。
  （要求见纸质课件）
- 上机习题(写在作业本上)

E-8：　　1-27(查看目录)，
在/bin目录下找到sh、csh、bash，完成　1-30.1