

# 高级操作系统

## Advanced Operating System

---

### Distributed Systems

### Concepts and design

朱 青 Qing Zhu,

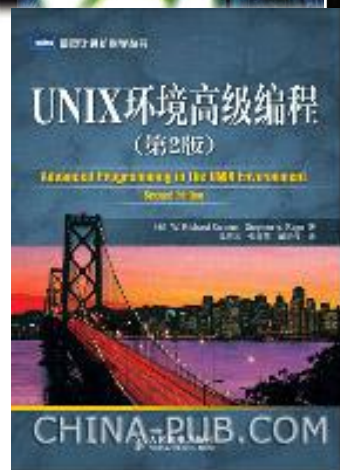
Department of Computer Science, Renmin University of China

[zqruc2012@aliyun.com](mailto:zqruc2012@aliyun.com)

# Chapter 1

## Characterization of Distributed Systems

### 分布式系统特征



朱 青

信息学院计算机系，  
中国人民大学，

[zqruc2012@aliyun.com](mailto:zqruc2012@aliyun.com)

# Textbook



⌘ **Distributed Systems – Concepts and Design, George Coulouris, Jean Dollimore, and Tim Kindberg**

⌘ **(第 2, 3, 4, 5 版)**

⌘ **The lecture is based on this textbook.**

# 第1章：分布式系统特征

---

- ⌘ 1.1 简介 (Introduction)
- ⌘ 1.2 分布式系统实例 (Distributed Systems)
- ⌘ 1.3 教学内容
- ⌘ 1.4 考核要求
- ⌘ 1.5 教材与参考文献
- ⌘ 1.6 分布式系统的趋势
- ⌘ 1.7 资源共享和web (Resource Sharing and Web)
- ⌘ 1.8 挑战 (Challenges)
- ⌘ 1.9 操作系统支持

# 1.8 挑战 Challenges of Distributed Systems

---

**When constructing a distributed system the designer encounters challenges that are specific to distributed systems.**

**☒ The size of each challenge depends on the requirements of the specific system.**

Those challenges can be assigned to several classes:

- |                             |        |
|-----------------------------|--------|
| 1. Heterogeneity            | (异构性)  |
| 2. Openness                 | (开放性)  |
| 3. Security                 | (安全性)  |
| 4. Scalability              | (可伸缩性) |
| 5. Failure handling         | (故障处理) |
| 6. Concurrency              | (并发性)  |
| 7. Transparency             | (透明性)  |
| 8. Quality of Service (QoS) | (服务质量) |

# Challenge: Heterogeneity (异构性)

---

- ⌘ Networks: **Ethernet** (以太网), **WLAN** (无线局域网), **GSM** (数字通), **UMTS** (? 军事训练), **ATM**, ...
- ⌘ Data types: **integer types**, **floating point types**, ...
- ⌘ Computer hardware: **Processor**, **printer**, **storage**, ...
- ⌘ Operating systems: **various types of Windows and UNIX**, ...
- ⌘ Programming languages: **C++**, **Java**, ...
- ⌘ **Application implementation**

# Challenge: Heterogeneity (异构性)

---

**Common interfaces are needed: TCP/IP for the Internet, ...**

**Middleware**中间件: **A software layer that separates the application from the system software and hardware layer**

一个软件层，提供了一个编程抽象以及对低层网络、硬件、**OS**和编程语言异构性的屏蔽。（4，5，17章）

☒ **It provides a programming abstraction.**

☒ **Examples: CORBA, JavaRMI, ...**

**Mobile code**移动代码 **can be sent from one computer to another and run at the destination: Java byte code**

⌘ **虚拟机方法：提供了一种使代码可在任何硬件上运行的方法。**

# Challenge: Openness（开放性）

---

**In an open system new services can be added or existing services can be re-implemented.**

开放性决定系统是否能以各种不同的方式扩展和重现。

**⌘ Key software interfaces of the components of an open system must be standardized or at least published.**（开放系统的特征：主要的接口是对外发布的）

**☒ Request For Comments 征求意见稿(RFC) for Internet protocols ([www.ietf.org](http://www.ietf.org)), 发布的手段。**

**☒ Technical documents for CORBA ([www.omg.org](http://www.omg.org))**



# Challenge: Openness（开放性）

---

- ⌘ **Open distributed systems are based upon a uniform communication mechanism（一致的通信机制）.**
- ⌘ **Open distributed systems are independent from individual vendors**
- ⌘ 开放的分布式系统是独立与经销商.
  - ☑ **The conformance of a component to the published standard must be tested and verified.**
- ⌘ 每个组件与发布的标准之间的一致性要测试和验证。

# Challenge: Security (安全性)

---

**It is more difficult to achieve security in a distributed system than in a centralized system due to the vulnerability of messages.**

## **Components of security:**

- ⌘ Confidentiality机密性: Protection against disclosure (泄露) to unauthorized individuals or applications**
- ⌘ 防止泄露给未授权的个人**

# Challenge: Security (安全性)

---

⌘ Integrity完整性: **Protection against alteration or corruption**

☑ 防止改变或讹误。

⌘ Availability可用性: **Protection against interference with the means to access the resources**

☑ 防止对访问资源的手段的干扰。

⌘ Authentication鉴定: **Reliable and correct identification of a remote user or another agent.**

☑ 远程用户或代理可靠性和正确性鉴定。

# Challenge: Security (安全性)

---

## Important present security challenges:

Denial of service拒绝服务攻击: **Disruption of a service by another user.**

用户可能因为某些原因中断服务。

Mobile code: **This code may include malicious (恶意的) components.**

移动代码的安全 (7章)

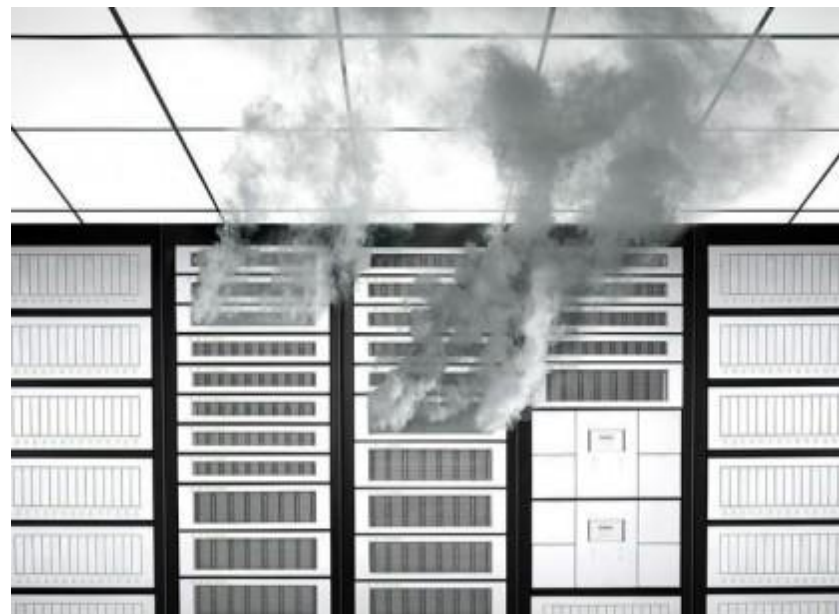
# 2014年最具代表性的七大网络入侵事件

---

- ⌘ 土耳其石油管道事件
- ⌘ 伊朗黑客瞄准航空公司系统
- ⌘ **JP**摩根被黑事件
- ⌘ 索尼影业被黑事件
- ⌘ 社区医疗系统被黑
- ⌘ 大型零售商家被黑
- ⌘ 心脏流血（**Heartbleed**）和**Shellshock**。

# JP摩根被黑事件

⌘ 就网络安全而言，**JP**摩根一直都是行业内的佼佼者，因为该公司每年都会投入**2.5**亿美元资金用于打造安全的网络系统。但在今年**10**月，**JP**摩根电脑系统的数据在三个月内出现了第二次泄露，此次泄露涉及约**8300**万客户（其中包括**7600**万家庭和**700**万小企业客户的姓名、地址、电话号码、电邮信息），成为史上规模最大的客户数据泄露案之一。



---

⌘ 市场研究机构**IDC**指出，如今开源软件的触角几乎已经深入了我们生活的方方面面，包括国际空间站、股票交易所等重要机构、设施中都会使用这类软件，而且这类软件的使用比例甚至超过了**95%**。但是，一种名为心脏流血（**Heartbleed**）和**Shellshock**全新攻击形式的出现却彻底改变了人们对于这类软件的看法。

# 心脏流血（Heartbleed）和Shellshock

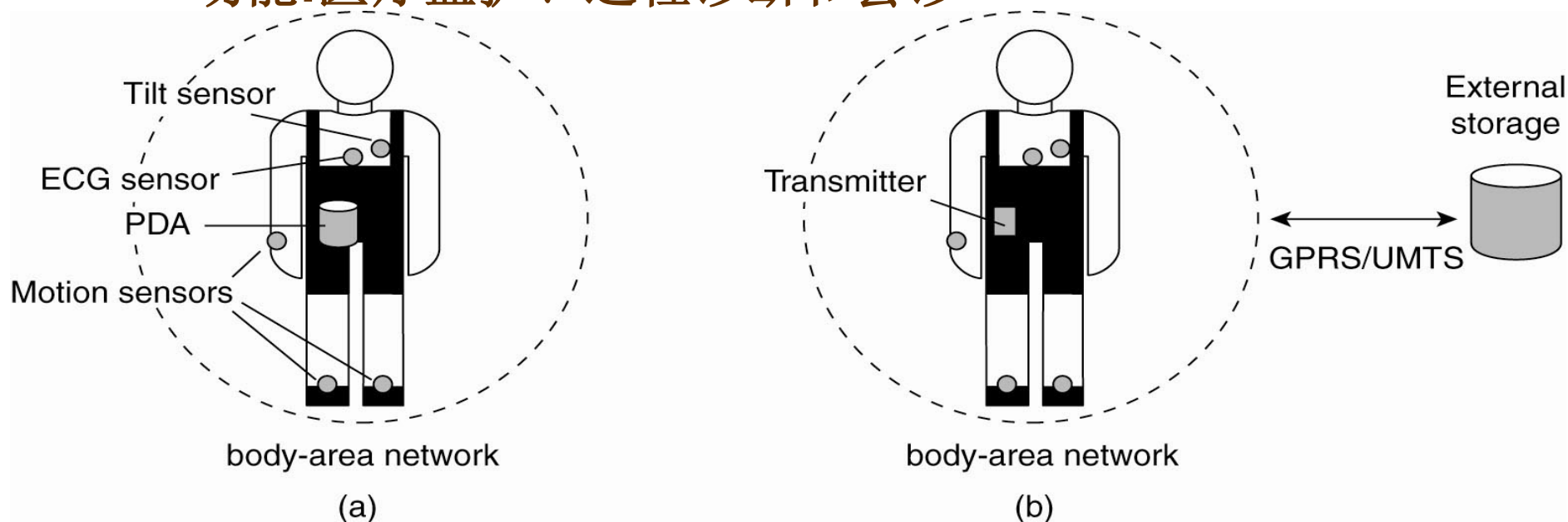
✂ 举例来说，心脏出血是一个出现在开源加密库**OpenSSL**的程序漏洞。通过这一漏洞，黑客可以读取到包括用户名、密码和信用卡号等隐私信息在内的敏感数据，并已经波及了大量互联网公司，受影响的服务器数量可能多达几十万，其中已被确认受影响的网站包括**Imgur**、**OKCupid**、**Eventbrite** 以及**FBI**网站等。同时，在这一漏洞遭到披露后，有不少美国科技企业均表示自己会在今后进一步加大对于系统安全方面的投入。





# 电子保健系统与安全私密性

- 配有各种传感器，构成身体局域网
- 功能: 医疗监护、远程诊断和会诊



- **(a)** 使用本地网络集线器
  - **(b)** 使用无线连接

# 电子保健系统与安全私密性

---

电子保健系统要解决的问题:

- ⌘ 监视数据如何保存?
- ⌘ 如何防止关键数据的丢失?
- ⌘ 生成和广播警告信息需要什么基础设施?
- ⌘ 医生如何提供在线反馈?
- ⌘ 如何实现监视系统的健壮性?
- ⌘ 会有哪些安全问题? 如何采取正确的策略?

# Challenge: Scalability (可伸缩性)

---

**A system is scalable if it remains efficient although there is significant increase in the number of resources and the number of users.**

如果资源数量和用户数量激增，系统仍能保持有效，该系统可被描述成可伸缩性。

## **Challenges to scalability:**


⌘ Controlling the cost of physical resources

**The cost for adding  $n$  users should be at most  $O(n)$ .**

⌘ 控制物理资源的开销： $n$ 个用户的物理资源代价保持 $O(n)$ 。

# Challenge: Scalability (可伸缩性)

---

- ⌘ Controlling the performance loss 控制性能丢失  
**The computational complexity of methods to manage  $n$  users should be  $O(\log n)$ .**
- ⌘ Preventing software resources from running out  
防止软件资源的耗尽  
**IP V4 limits the number of addresses to  $2^{32}$ .**  
 **Conversion to IP V6 is required to maintain scalability.**
- ⌘ Avoiding performance bottlenecks  
避免性能瓶颈
- ⌘ **Algorithms in a scalable systems should be decentralized.**
- ⌘ 通常算法应该分散，避免性能瓶颈

# Computers in the Internet

---

<i>Date</i>	<i>Computers</i>	<i>Web servers</i>
1979, Dec.	188	0
1989, July	130,000	0
1999, July	56,218,000	5,560,866
2003, Jan.	171,638,297	35,424,956

Figure 1.5 (第四版) 图表

Figure 1.6  
Growth of the Internet (computers and web servers)

<i>Date</i>	<i>Computers</i>	<i>Web servers</i>	<i>Percentage</i>
1993, July	1,776,000	130	0.008
1995, July	6,642,000	23,500	0.4
1997, July	19,540,000	1,203,096	6
1999, July	56,218,000	6,598,697	12
2001, July	125,888,197	31,299,592	25
2003, July	~200,000,000	42,298,371	21
2005, July	353,284,187	67,571,581	19

# 光棍节 呈现

## 全场5折

### 11月11号 约定你啦

30000个  
热门品牌  
汇聚一网

淘宝商城

www.tmall.com

网购品牌正品,从这里开始

淘宝商城

品质保证



淘宝商城

Q 搜索什么商品?

按商品

按店铺



## 全场5折!

11月11日 仅此一天

全场包邮

分享好

网购狂欢节  
11.11

再一次,让世界沸腾!

全场5折! 全国包邮!

仅此1天 开抢

敬请期待 1111.tmall.com

Tmall.com

商品 店铺

搜索 淘宝网 商品

搜索

所有商品分类

服装、内衣、配件

女装 男装 内衣 家居服 配件  
毛衣 卫衣 羽绒服 夹克 裤衣

鞋、箱包

女鞋 男鞋  
箱包 女包 男包 拉杆箱 背包

珠宝首饰、手表眼镜

饰品 项链 珠宝 钻石 手表

化妆品

护肤 彩妆 香水 男士 精油 美容

运动、户外

运动鞋 运动服 运动用品 户外

手机、数码

手机 笔记本 相机 平板电脑  
配件 电脑硬件 移动存储

家用电器

首页

品牌导航

品牌特卖

淘品牌

港台&海外馆

综合卖场

会员中心

积分兑换

# 11.11

星期五 农历十月十六

宜 5折  
包邮  
血拼

忌 伤心  
错过  
淡定

## 物价回到十年前

服装 | 鞋包 | 护肤 | 居家

5折

### 品牌特卖

#### 双11狂欢

51元红包  
快来抢

### 狂欢11.11

#### 家电排行

超值精选商品  
低至2折

### 网购狂欢

#### 全场5折

全场商品 全场包邮

一年一遇 一起期待



# 双11淘宝的销售数据

- ⌘ 淘宝“双11”交易额突破**350**亿元 创下新纪录
- ⌘ 截止**12**日零时整，**11·11**购物狂欢节实时交易总额达**350**亿。





# 历年双11的交易额的对比

---

- ⌘ **2013年11月11日，注定会被载入互联网的历史2011年双十一支付宝交易额52亿元。**
- ⌘ **2012年支付宝交易额191亿元，而今年则达到了350亿元。而今年则首次完成了对线下零售额的超越。**

# 历年双11的交易额的对比

## ⌘ 历年双11线上，网络零售额



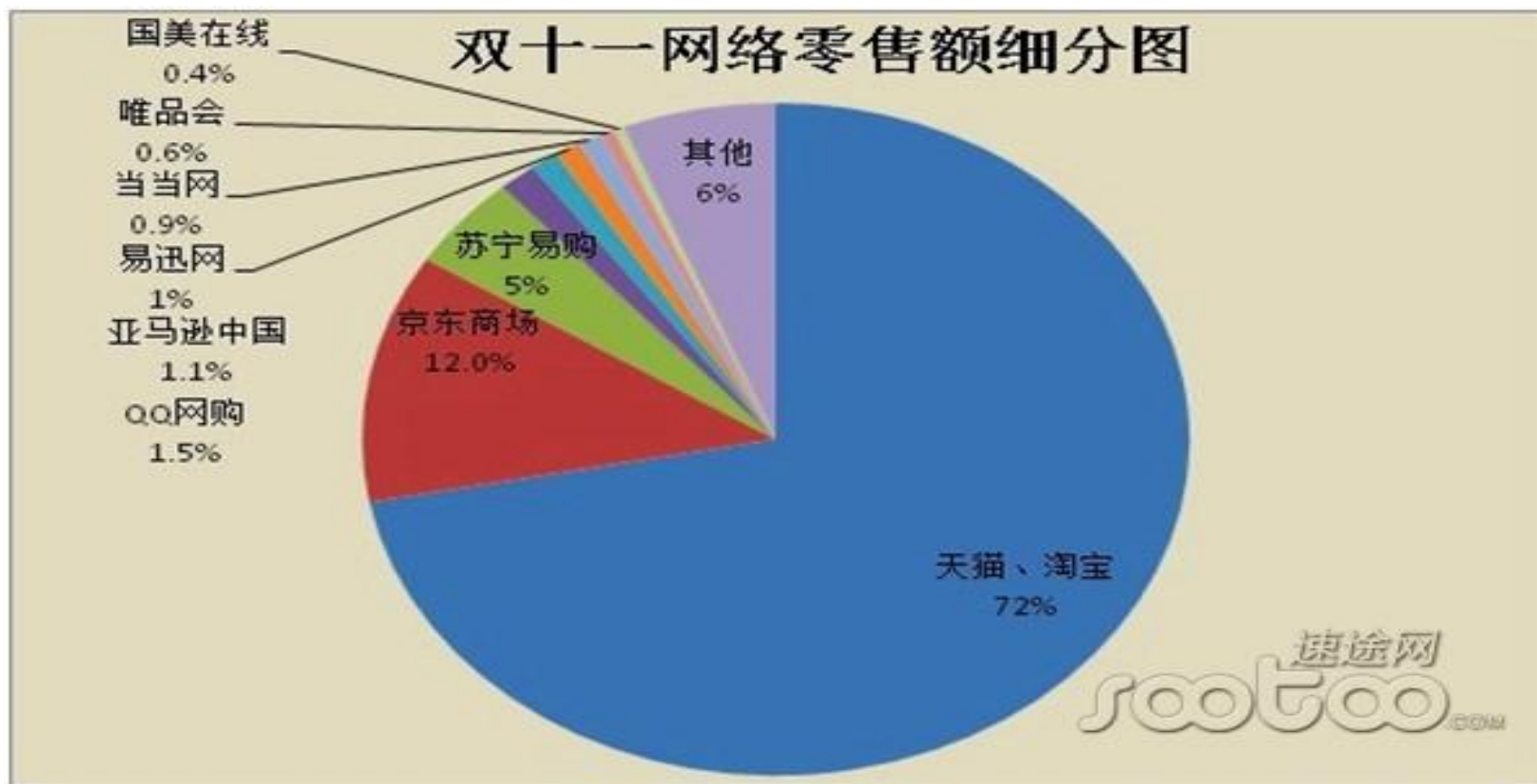
# 交易额比例

⌘ 2013年双十一服装鞋帽为第一销售额类目



# 双11天猫、淘宝网络零售额

⌘ 双十一网络零售额天猫、淘宝占比达72%



# 双11当日网络交易额最多的省份

⌘ 双11当日网络交易额最多的省份为浙江省



# 系统峰值的处理

---

## ⌘ 从双11看电子商务系统的发展与挑战

### ⌘——鏖战双11，电商架构大起底

⌘ 自2009年11月11日，淘宝商城（现名天猫）拉开网购狂欢节的序幕，各大电商的促销浪潮此起彼伏。

⌘ 此时的电商大战不仅是价格之争，更是技术的较量。

⌘ 如何设计电商峰值系统来更好地满足用户蜂拥而至的访问

⌘ 如何在海量数据处理中实时发现有效信息并转化为商机

⌘ 成为众多电商企业密切关注的问题。

⌘

# 鏖战双11，电商架构大起底

---

- ⌘ 自**2009年11月11日**，淘宝商城（现名天猫）拉开网购狂欢节的序幕，各大电商的促销浪潮此起彼伏。
- ⌘ 此时的电商大战不仅是价格之争，更是技术的较量。
- ⌘ 如何设计电商峰值系统来更好地满足用户蜂拥而至的访问
- ⌘ 如何在海量数据处理中实时发现有效信息并转化为商机
- ⌘ 成为众多电商企业密切关注的问题。

# 快稳炫：电商峰值系统架构三字诀

---

- ⌘ 双11这样的大型促销活动带来的是流量暴涨，在高访问量的冲击下，电商系统会受到以下挑战：
- ⌘ 瞬间访问量可能是平时的几十倍；
- ⌘ 网络带宽被占满，用户响应很慢；
- ⌘ 机器负载高甚至宕机；
- ⌘ 数据库压力过大导致服务不可用。
- ⌘ 此时，如何设计电商峰值系统来更好地满足用户蜂拥而至的访问，如何在海量数据处理中实时发现有效信息并转化为商机，成为众多电商架构师都在认真思考的问题。
- ⌘ 解决这一问题的核心思路：采用分而治之的思想，大系统小做，小系统大做。浓缩一下就是三个字：快、稳、炫。



# Challenge: Failure Handling故障处理 (1)

---

**Failures in a distributed system are partial – not all components fail.**

分布式系统的故障是部分的。

☑ **The functioning system can still be used.**

☑ **The handling of failures is particularly difficult.**

⌘ **Failure detection and localization故障的诊断和定位**

☑ **Detection of a corrupted data transmission by a checksum** 出错数据传输的诊断

☑ **A localization of a faulty network component may be difficult.** 错误网络组件的定位困难

# Challenge: Failure Handling

---

- ⌘ Masking failures: **Hiding a failure from a user or an application program.**
  - ⌘ 故障覆盖：用户或应用程序隐藏错误
  - ⌘ **Automatic retransmission of missing messages**
  - ⌘ 丢失消息的自动重发
    - ☑ **Use of RAID to store files**
- Those methods do not work in a (rare) worst case.**

# Challenge: Failure Handling (2)

---

- ⌘ **Tolerating failures容错: The client does not fail if there are failures that cannot be detected or hidden.**
  - ☒ **After several unsuccessful retransmission attempts the system gives up and informs the client. The client continues with other work and retries the transmission later.**
- ⌘ **Recovery from failures故障恢复: A failure does not cause permanent damage.**
  - ☒ **After a remote invocation has been stopped by a server crash the original state before the invocation is restored.**
  - ☒ **A faulty component is integrated into the system after its repair.**

# Challenge: Failure Handling

---

- ⌘ **Redundancy冗余** **Failure hiding can often be achieved by use of redundant components.**
- ⌘ 服务可以利用冗余的组件实现容错。
  - ☑ **Different routes between two routers in the Internet**
  - ☑ 两个路由器之间，至少存在两个路由。
  - ☑ 在域名系统中，每个名字至少被复制到两个服务器上。
  - ☑ 数据库可以被复制到几个服务器上。
  - ☑ **Additional components and increased costs**

**Availability: The proportion of time a system is available for use**

# Challenge: Concurrency 并发性

---

**Resources are shared by several clients in a distributed system.**

分布式系统中服务和应用两者均提供可被用户共享的资源。

☒ **A shared resource may be accessed at the same time by more than one client.** 一个共享资源同时可以被多个服务器访问。

☒ **Handling all client requests sequentially limits the throughput of the server.** 处理用户需求序列方法限制了吞吐量。

**Many services support concurrent processing of multiple client requests.**

分布式系统中代表共享资源的任何一个对象必须负责确保他在并发环境中的操作正确。

☒ **Concurrent requests must not interfere with each other.**

☒ **Services and objects must be safe in a concurrent environment.**

中国人最难登的岛是

12306







中秋来了，国庆来了，12306也来了。一个想回家给外公过大寿的网友讲述了他的烦恼：用12306订火车票，非常不靠谱。他说，感受不到铁道部在升级12306网站，新增的车票预约和排队出票功能都很坑爹。他质疑，铁道部是“有钱拍宣传片，没钱搞服务”。





17号晚上：接到爸爸电话决定回家

17号晚上：登陆12306，系统提示：预约时间已过，预约时间为2012年09月10日，11:00至20:00。”

18号早上6点：登陆12306，系统提示还在维护中，提示7点-23点可以登录

18号早上7点：刷新，29号的票还没开始卖

18号早上7点05：刷新，还没开始卖

7点20…7点30…7点55…8点：刷新，12306提示，：当前访问用户过多，请稍后重试！”

8点03分：终于登录上去了，按照流程，点了提交订单，系统显示：今日该车次和席别已有超过415人次先于您提交订单需求，至处理您提交请求时可能已无票，是否继续？

系统提示：排队等待中，您还需等待大于30分钟

排队等待中，您还需等待25分33秒

排队等待中，您还需等待18分21秒

排队等待中，您还需等待09分44秒

排队等待中，您还需等待00分05秒





出票失败。很抱歉！当前提交订单用户过多，请您稍后重试。”

继续奋斗。。。。。

刷新，登陆，一查居然还有余票！

新一轮有开始了

出票失败。很抱歉！当前提交订单用户过多，请您稍后重试。”

18号奋斗结束。。。。。





19号早上，鉴于18号的惨痛经验，同时在电脑上打开了火狐浏览器和搜狗浏览器，并启用了搜狗的自动登录功能。我知道，一个人用两个ID，从概率上来说，是提高了自己的命中率，但等于也把别人回家的希望降低了一些。不过这个时候，已经顾不得什么活雷锋精神，占着茅坑不拉屎就占吧，简直恨不得用3个ID来登了！朋友说，也试试在95105105之前加上保定或者唐山的区号，提高一下电话的成功率。我觉得，我们准备工作做的这么足，应该可以订的上吧？

结果，结果就没有结果了。



# 各种声音

春运时就发生过的网站堵塞、登录失败、排队等候、购票失败等等问题，现在依旧没有解决，想让一个斯文的人变得粗俗、满口脏话吗？那就请他去12306买票吧……想让一个淑女瞬间变泼妇吗？那就请她去12306买票吧……铁道部啊，找几个靠谱的产品经理和技术这么费劲么？

7, 我选择抢到了，不容易，话说抢到票还是在万能的淘宝上买了一个火车票刷票外挂！擦，你没看错，是刷票外挂！尼玛买火车票都要用外挂了，挂机刷票啊，刷到还要火速购买啊。

政府的民心工程，应该把买票难的事放在第一位。



# Challenge: Transparency透明性 (1)

---

**Transparency is the concealment (隐藏) from the user and the application programmer of the separation of components in a distributed system.**

☑ **The system is perceived as a single computer.**

☑ 透明性被定义成对用户和应用程序员屏蔽分布式系统的组件的分散性，系统被认为是一个整体，而不是独立组件的集合。

⌘ **Access transparency: Local and remote resources are accessed by using identical methods**

☑ **File access in a distributed file system without requiring explicit methods like ftp.**

# Challenge: Transparency<sub>透明性</sub>

---

- ⌘ Location transparency（位置的透明性）：**Resources can be accessed without knowledge of their locations.**
- ⌘ 不需要知道资源的位置就能访问它们。
  - ☑ **A user may want to address a resource at a specific location, like a printer:** Location aware computing.
  - ☒ **The location of a resource must be available on request.**

**Access and location transparency together are also called network transparency.**

## Challenge: Transparency (2)

---

- ⌘ **Concurrency transparency (并发透明性) : Several processes can operate concurrently on shared resources without interference among them.**
- ⌘ **几个进程并发的对共享资源进行操作而互不干扰。**
- ⌘ **see the concurrency challenge.**
  - ☒ **The system needs to prevent undesired (不希望的) concurrent activities.**
  - ☒ **Two users cannot modify the same file concurrently.**
  - ☒ **Use of locks or semaphores.**



# Challenge: Transparency

---

- ⌘ Replication transparency（复制透明性）：**Multiple instances of resources can be used to increase reliability and performance without knowledge of the replicas by users or application programmers.**
- ⌘ 使用资源的多个实例增加可靠性和性能，而用户和应用程序员无需了解副本的存在。
  - ☑ **Replicas of file servers in some distributed file systems.**
- ⌘ Failure transparency（故障透明性）：**Faults specific to distributed systems are concealed from users and application programs, 分布式系统的故障时被用户和应用程序隐藏。**
  - ☑ **see the failure handling challenge.**

## Challenge: Transparency (3)

---

- ⌘ **Mobility (migration) transparency (移动透明性) :**  
**Resources and clients can move in the system without affecting the operation of users or programs.**
  - ☑ **Names do not change when resources are moved.**
  - ☑ **Example: Mobile phones**
- ⌘ **Performance transparency (性能透明性) :** **The system can be reconfigured to improve performance without requiring user activity.**
  - ☑ **The load of the system is balanced by automatically moving some services.**
  - ☑ **当负载变化时，容许系统重构以提高性能。**

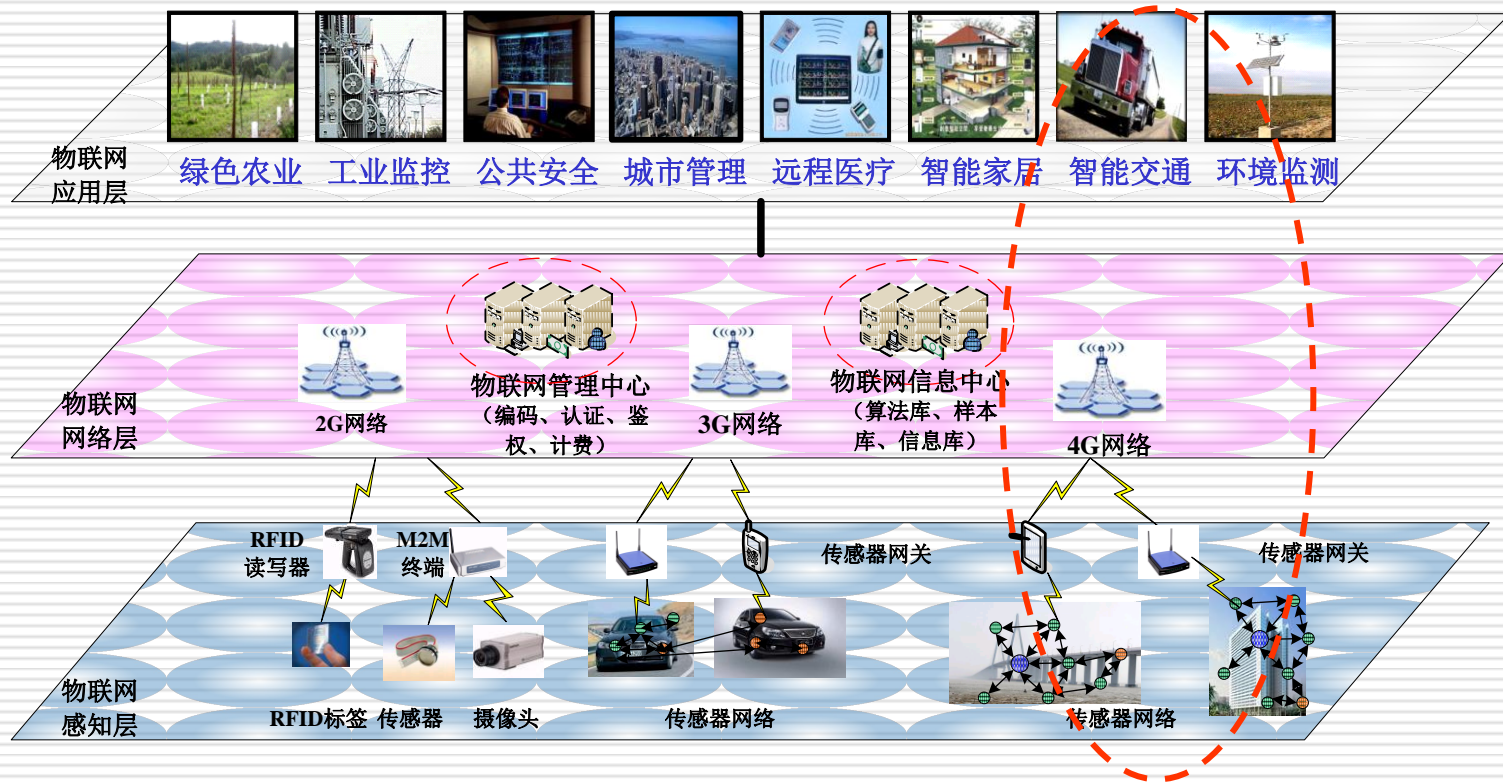
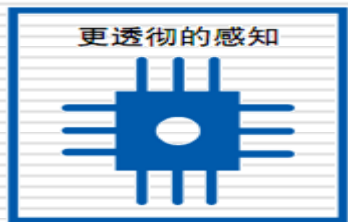
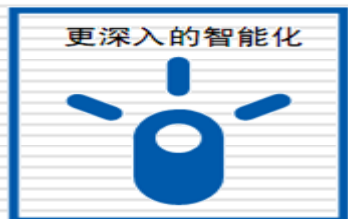


# Challenge: Transparency

---

- ⌘ **Scaling transparency（伸缩透明性）：The system and the scale of its applications can grow without requiring a change in the system structure or the application algorithms, see the scalability challenge.**
- ⌘ 在不改变系统结构或应用算法的前提下，允许系统和应用的扩展。

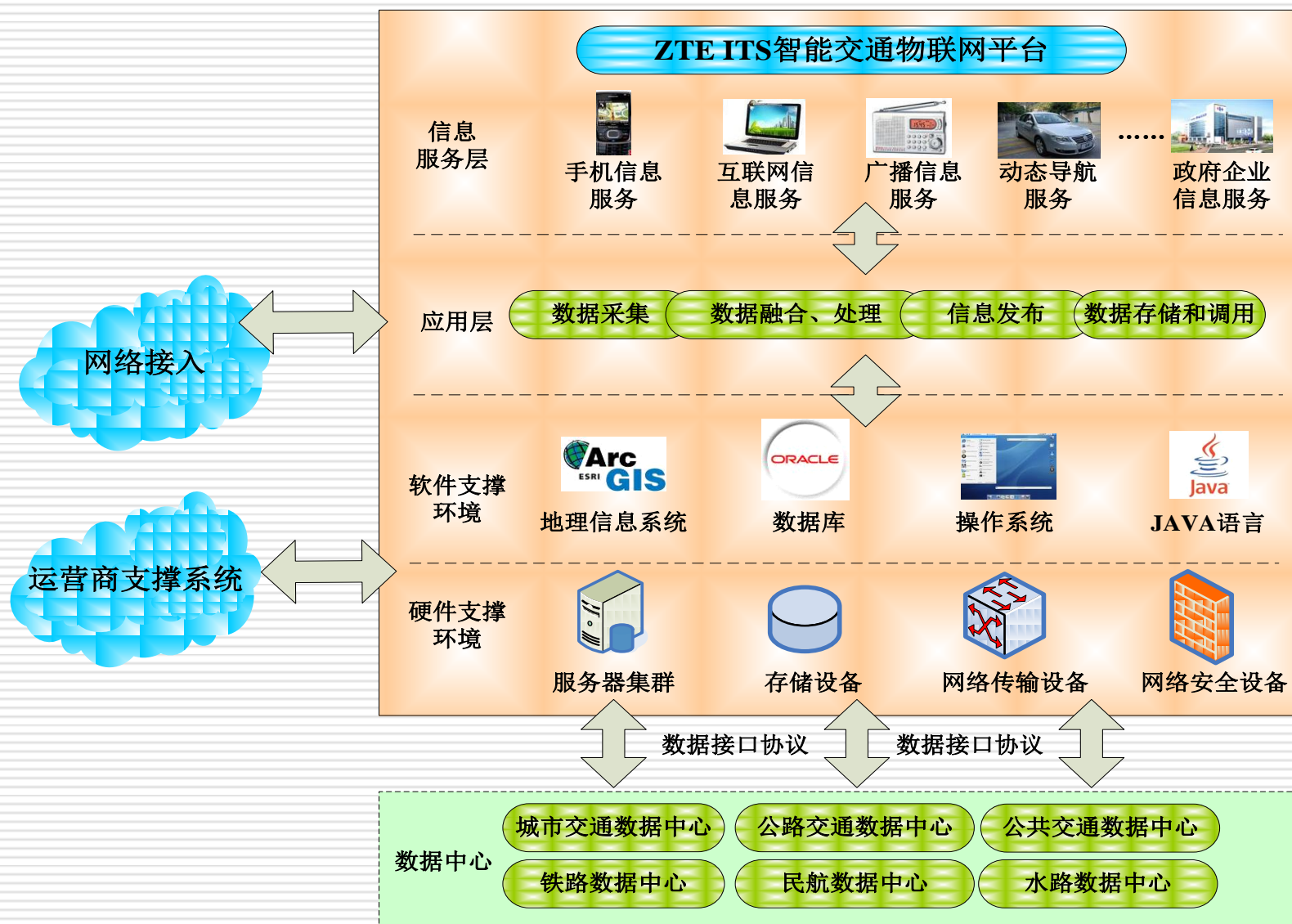
# 物联网的应用架构



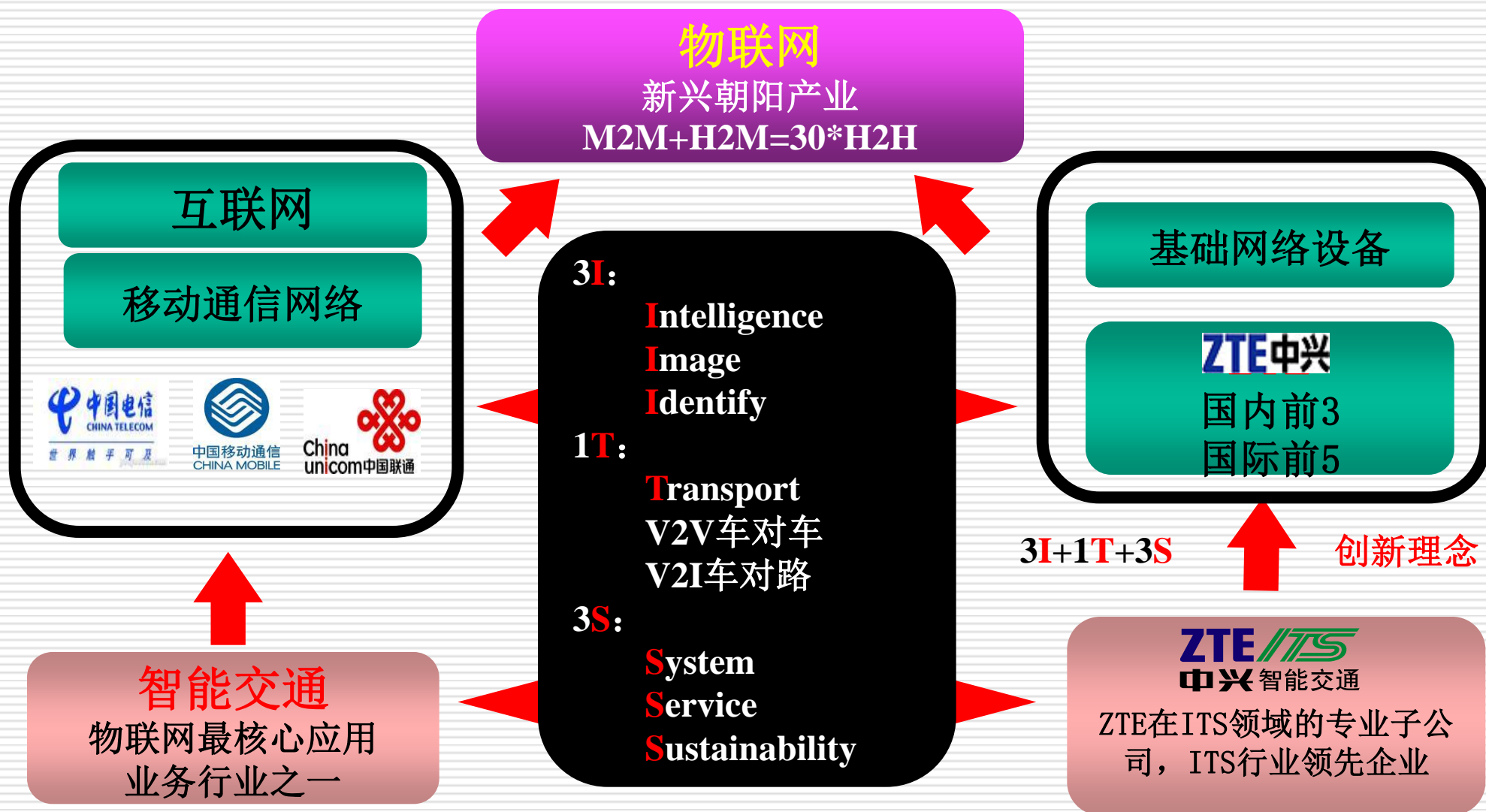
# 智能交通物联网：物联网产业化最能够率先成功的核心行业



# 智能交通物联网平台建设



# 智能交通物联网





# 未来中国ITS物联网“模型”



## Section 1.5.7 Transparencies

**Access transparency:** enables local and remote resources to be accessed using identical operations.

**Location transparency:** enables resources to be accessed without knowledge of their physical or network location (for example, which building or IP address).

**Concurrency transparency:** enables several processes to operate concurrently using shared resources without interference between them.

**Replication transparency:** enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers.

**Failure transparency:** enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.

**Mobility transparency:** allows the movement of resources and clients within a system without affecting the operation of users or programs.

**Performance transparency:** allows the system to be reconfigured to improve performance as loads vary.

**Scaling transparency:** allows the system and applications to expand in scale without change to the system structure or the application algorithms.

# Challenge: QoS 服务质量

---

⌘ **QoS** (Quality of Service) 服务质量

⌘ 关键指标主要包括:

⌘ 可靠性

⌘ 安全性

⌘ 系统性能

⌘ 可用性、吞吐量、时延、时延变化(抖动等)和丢失

⌘ 可信性

⌘ 用户响应

⌘ ~ ~ ~



# 第1章：分布式系统特征

---

- ⌘ 1.1 简介 (Introduction)
- ⌘ 1.2 分布式系统实例 (Distributed Systems)
- ⌘ 1.3 教学内容
- ⌘ 1.4 考核要求
- ⌘ 1.5 教材与参考文献
- ⌘ 1.6 分布式系统的趋势
- ⌘ 1.7 资源共享和web (Resource Sharing and Web)
- ⌘ 1.8 挑战 (Challenges)
- ⌘ 1.9 操作系统支持

# 1.9 操作系统支持 Operating System Support

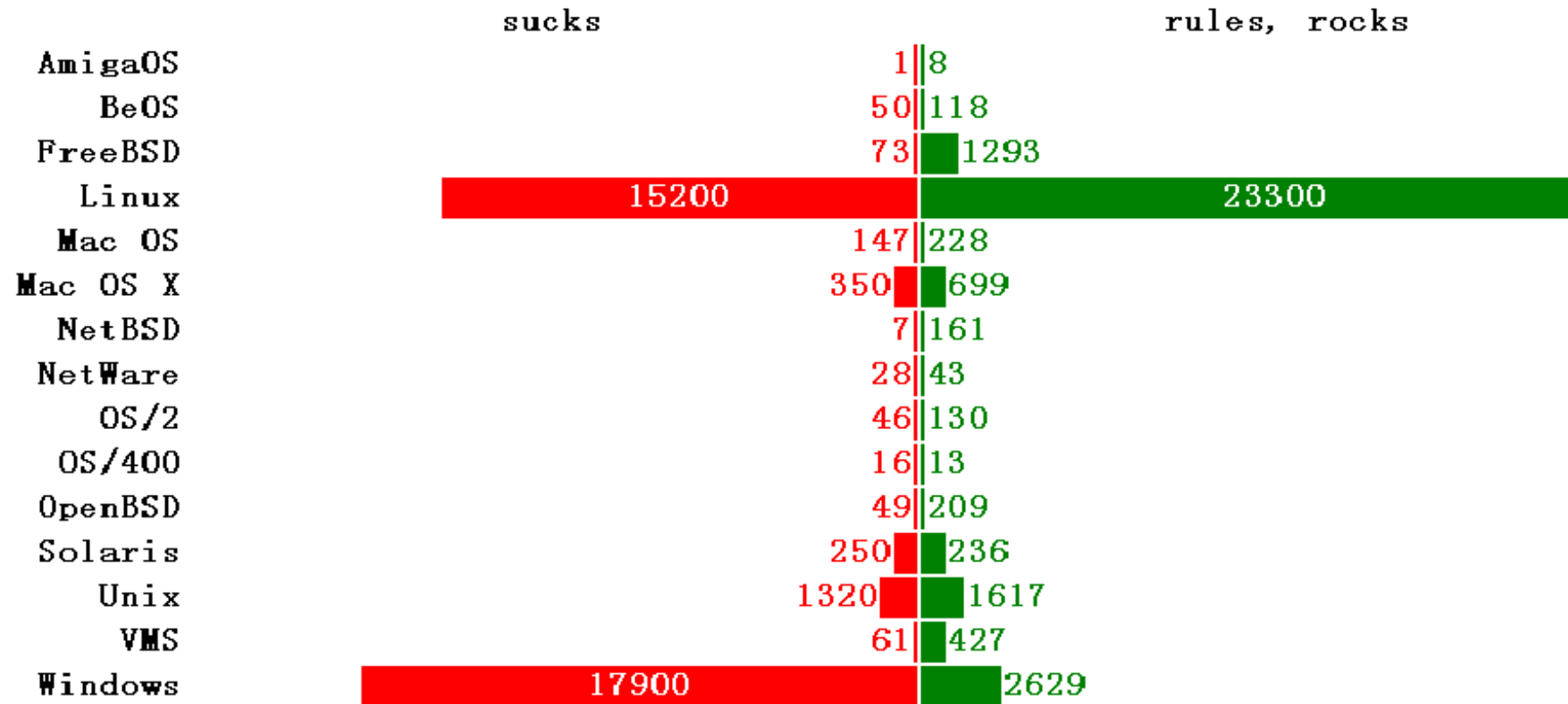
---

- ⌘ Who has used UNIX/LINUX?
- ⌘ Who was programmed in C/C++?
- ⌘ Who has programmed under UNIX/LINUX?
- ⌘ Do you have departmental accounts?

# People Love UNIX

---

## Operating System Sucks-Rules-0-Meter



Updated Sun Aug 22 08:03:43 2004

<http://srom.zgp.org/>

# Web Servers: Linux vs Windows 2003

---

Platform		
Ease of Use	▶▶▶▶▶▶▶▶▶▶	▶▶▶▶▶▶▶▶▶▶
Reliability	▶▶▶▶▶▶▶▶▶▶	▶▶▶▶▶▶▶▶▶▶
Speed	▶▶▶▶▶▶▶▶▶▶	▶▶▶▶▶▶▶▶▶▶
Functionality	▶▶▶▶▶▶▶▶▶▶	▶▶▶▶▶▶▶▶▶▶
Price	▶▶▶▶▶▶▶▶▶▶	▶▶▶▶▶▶▶▶
Integrated with both Solutions	▶▶▶▶▶	▶▶▶▶▶▶▶▶▶▶
Free / Open Source Software	▶▶▶▶▶▶▶▶▶▶	▶▶

# Why people like it?

---

- ⌘ powerful and robust
- ⌘ hard to use.
- ⌘ often free (Linux).
- ⌘ open.
- ⌘ and addictive.

# Getting Started With UNIX

## Section A: Objectives

---

After studying this lesson, you should be able to:

- ☒ Define operating systems in general
- ☒ Describe Linux as it relates to UNIX
- ☒ Explain the function of UNIX shells
- ☒ Describe the options for connecting to a UNIX system

# Getting Started With UNIX

## Section A: Objectives

---

After studying this lesson, you should be able to:

- ☒ Use the **date**, **who**, **man**, etc. commands
- ☒ Enter multiple commands on a single command line
- ☒ Recall a command from the command history
- ☒ Log on to and log out of UNIX

# Understanding Operating Systems

---

- ⌘ An operating system (OS) is the most important program that runs on a computer
- ⌘ Operating systems enable you to store information, process raw data, use application software, compile your own programs, and access all hardware attached to a computer



# Operating System Model

---

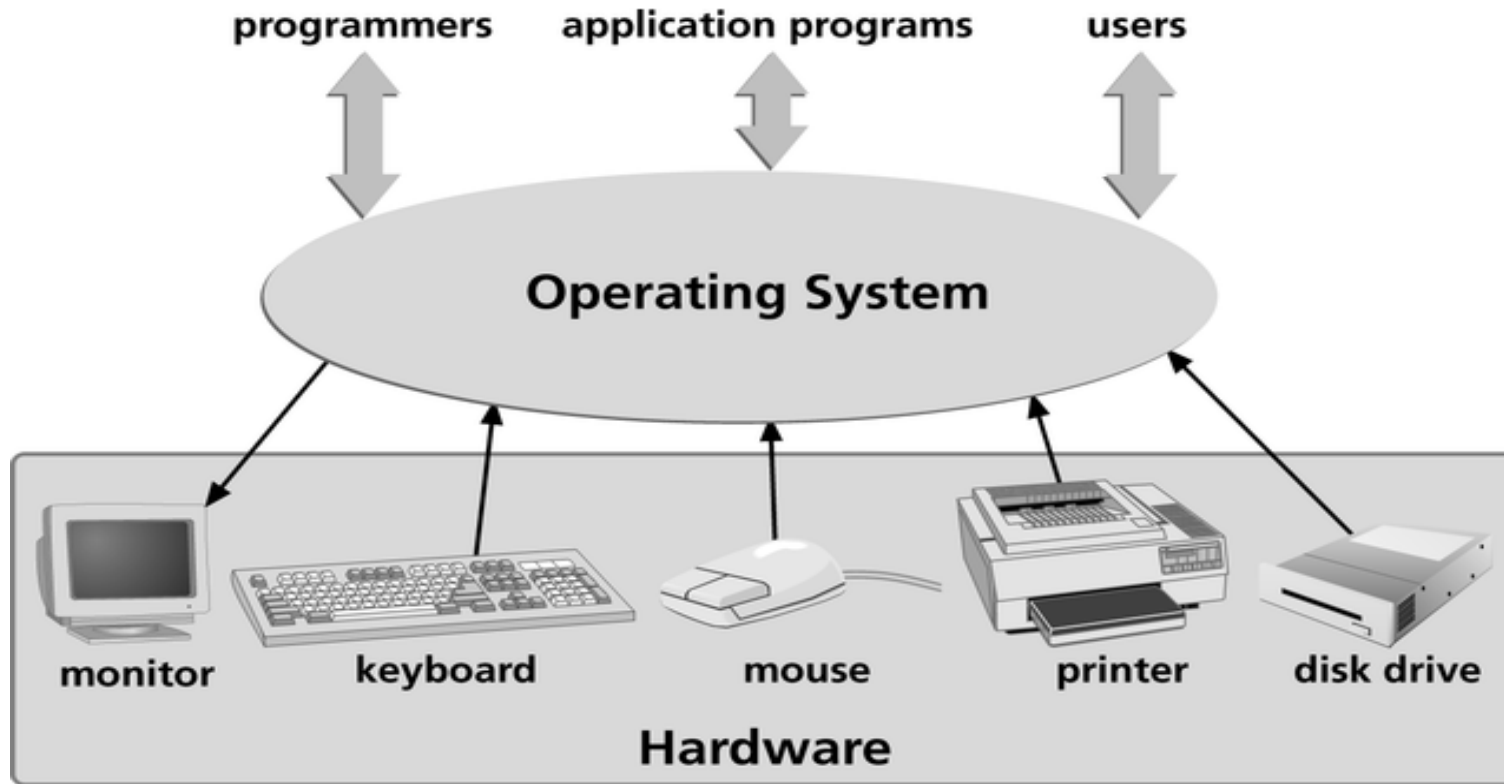


Figure 1-1: Operating system model

# Where can we find OS?

---

Computers



Phones



Game Boxes



Cars



- PC OSes
- Network OSes
- Distributed OSes
- Real-time OSes
- Embedded OSes

# Common PC Operating Systems

---

- ⌘ A **personal computer (PC) OS** conducts all I/O, processing, and storage operations on a stand-alone computer.



Figure 1-2: Common PC operating systems

# Common Mainframe Operating Systems

---

- ⌘ **A mainframe OS** controls a large computer system with multiple processors for I/O, processing, and storage operations for many users



Figure 1-3: Common mainframe operating systems

# Relationship of Servers and Clients on a Network

---

- ⌘ A **network OS** controls the operations of a server computer (host), which accepts requests from user programs running on other computers (clients)

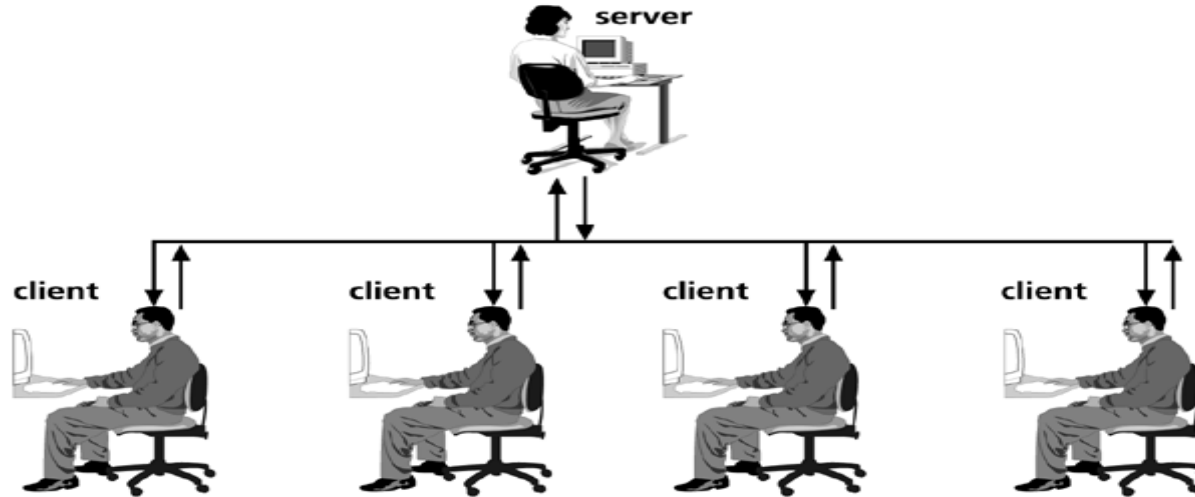


Figure 1-4: Relationship of servers and clients on a network

# Introducing the UNIX Operating System

---

- ⌘ UNIX is a multi-user, multitasking operating system with built-in networking functions
- ⌘ Single-user, single-task:  
historic, but rare (only a few PDAs use this)
- ⌘ Single-user, multi-tasking:  
Some PCs and workstations may be configured like this. e.g., Windows 95
- ⌘ Multi-user, multi-tasking:  
used on large, mainframes and PCs, workstations and servers today. e.g. UNIX, Windows XP, Linux

# Introducing the UNIX Operating System

---

- ⌘ UNIX is a **multi-user system**, which lets many people simultaneously access and share the resources of a server computer
- ⌘ A **multitasking system** lets one user execute more than one program at a time
- ⌘ UNIX is also a **portable** operating system

# Introducing the UNIX Operating System

---

- ⌘ Its **portability** means it can be used in a variety of computing environments.
  - ☒ Traditionally, most operating systems were written in **Assembler**, for a specific architecture.
  - ☒ VERY painful - if at all possible - to `port' the operating system to other architectures.
- ⌘ UNIX is mostly written in the **C language**. This alone allows UNIX to be portable to many architectures.
- ⌘ UNIX runs on a wider variety of computers than any other operating system

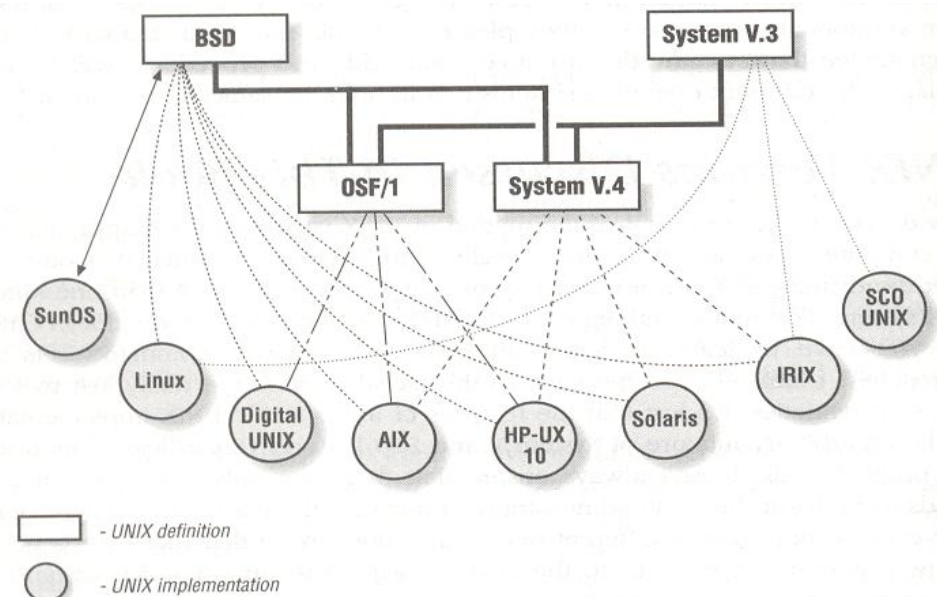
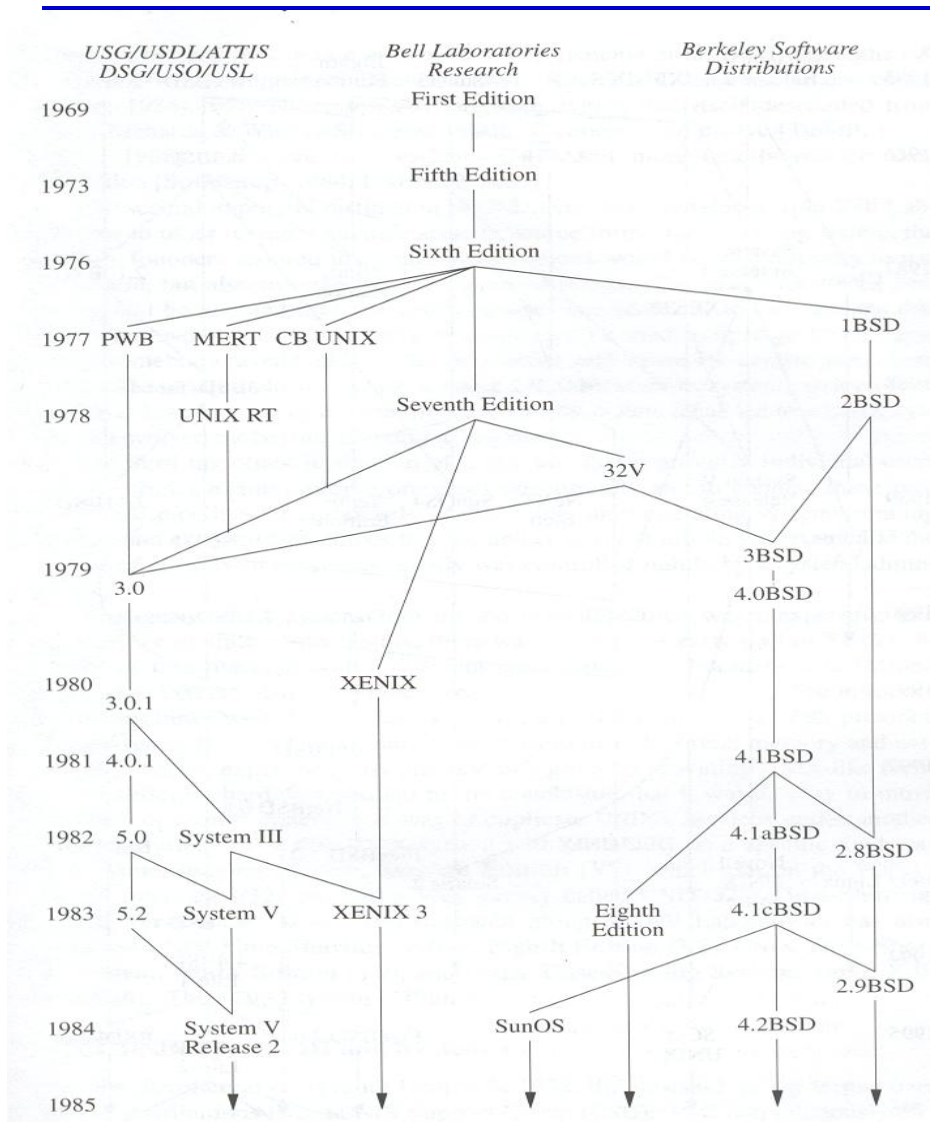


# A Brief History of UNIX

---

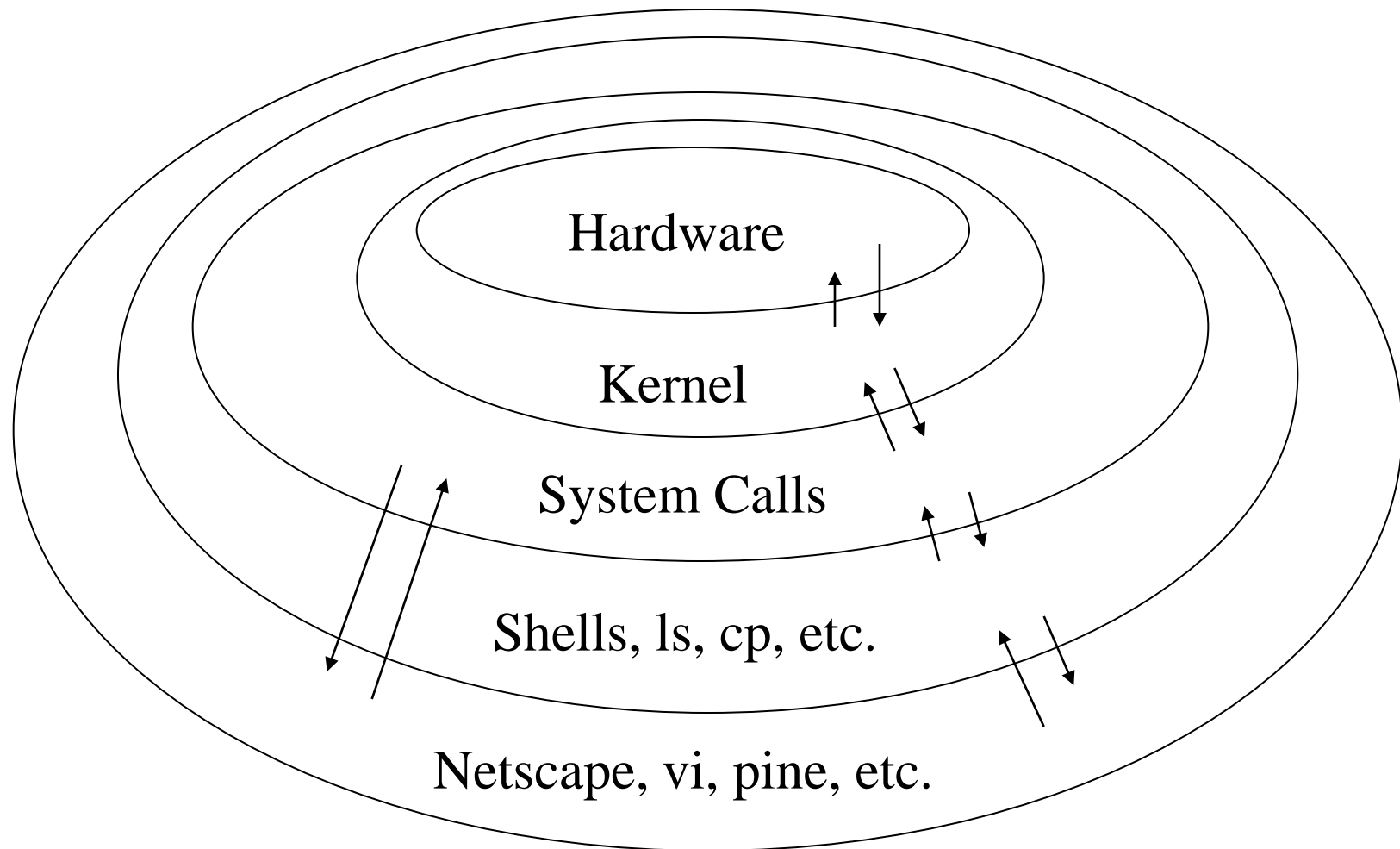
- ⌘ A group of programmers at Bell Labs originally developed UNIX in the early 1970s
- ⌘ Bell Labs distributed UNIX in its source code form, so anyone who used UNIX could customize it as needed
- ⌘ Eventually, two standard version of UNIX evolved:
  - ⏏ AT&T produced System V (1983)
  - ⏏ University of California at Berkeley developed BSD (Berkeley System Distribution, 2BSD, 1978)

# A Brief History of UNIX



# Layers of a UNIX System

---



# UNIX Concepts

---

- ⌘ Microsoft DOS and Microsoft Windows adopted original UNIX design concepts, such as the idea of a **shell**--an interface between the user and the operating system--and the **hierarchical structure** of directories and subdirectories
- ⌘ The **kernel** is the base operating system, which interacts directly with the hardware and services the user programs

# UNIX Concepts

---

- ⌘ The kernel is only accessible through **kernel mode**, which is reserved for the system administrator
- ⌘ This prevents unauthorized commands from invading the **foundation layer** or the hardware that supports the entire UNIX structure
- ⌘ **User mode** provides access to higher layers where all application software resides

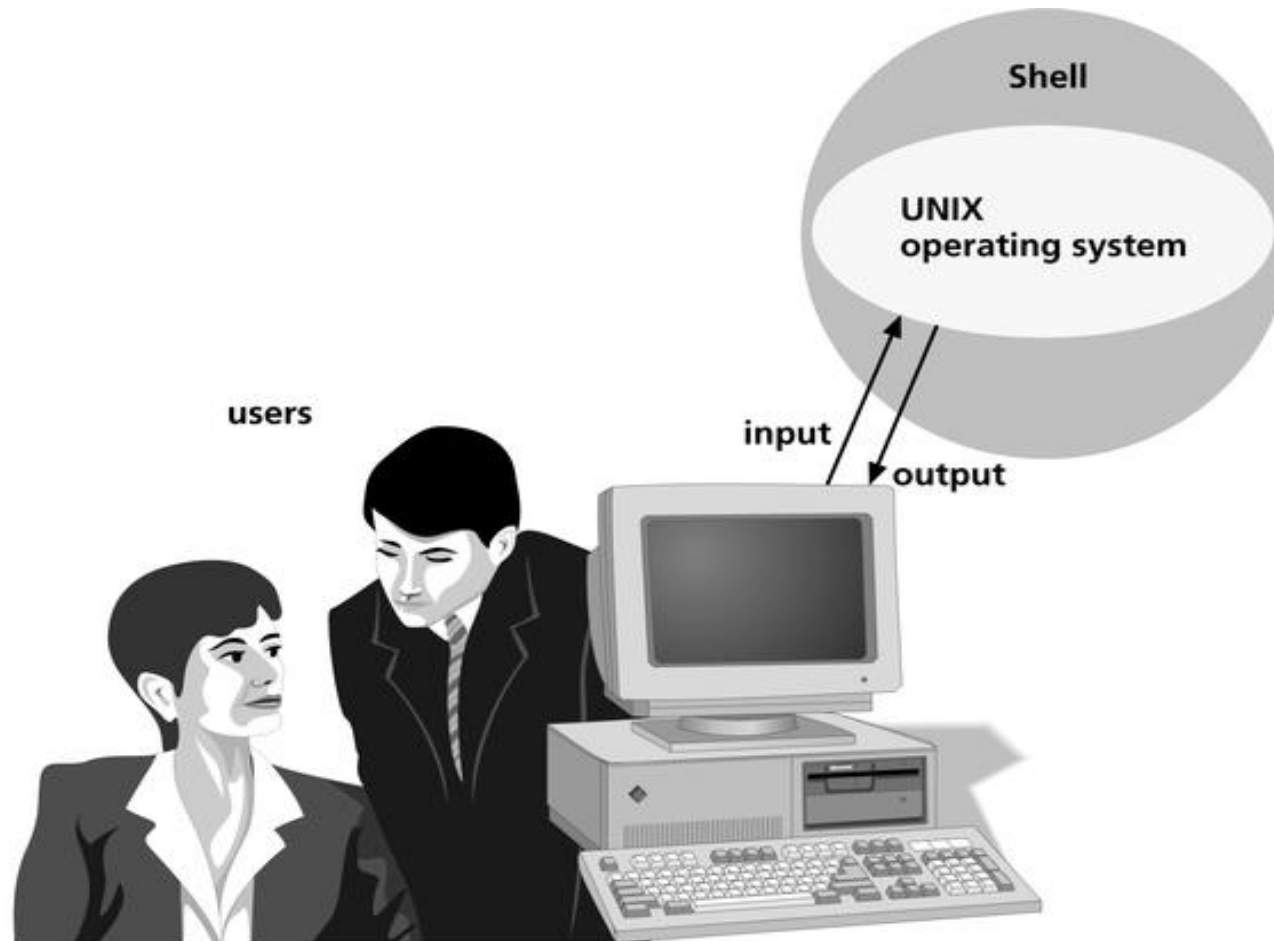
# Introducing UNIX Shells

---

- ⌘ The **shell** is a UNIX program that interprets the commands you enter from the keyboard
- ⌘ UNIX provides several shells, including the Bourne shell, the Korn shell, and the C shell
- ⌘ Linux uses the freeware Bash shell as its default command interpreter

# Shell's Relationship to the User and the Hardware

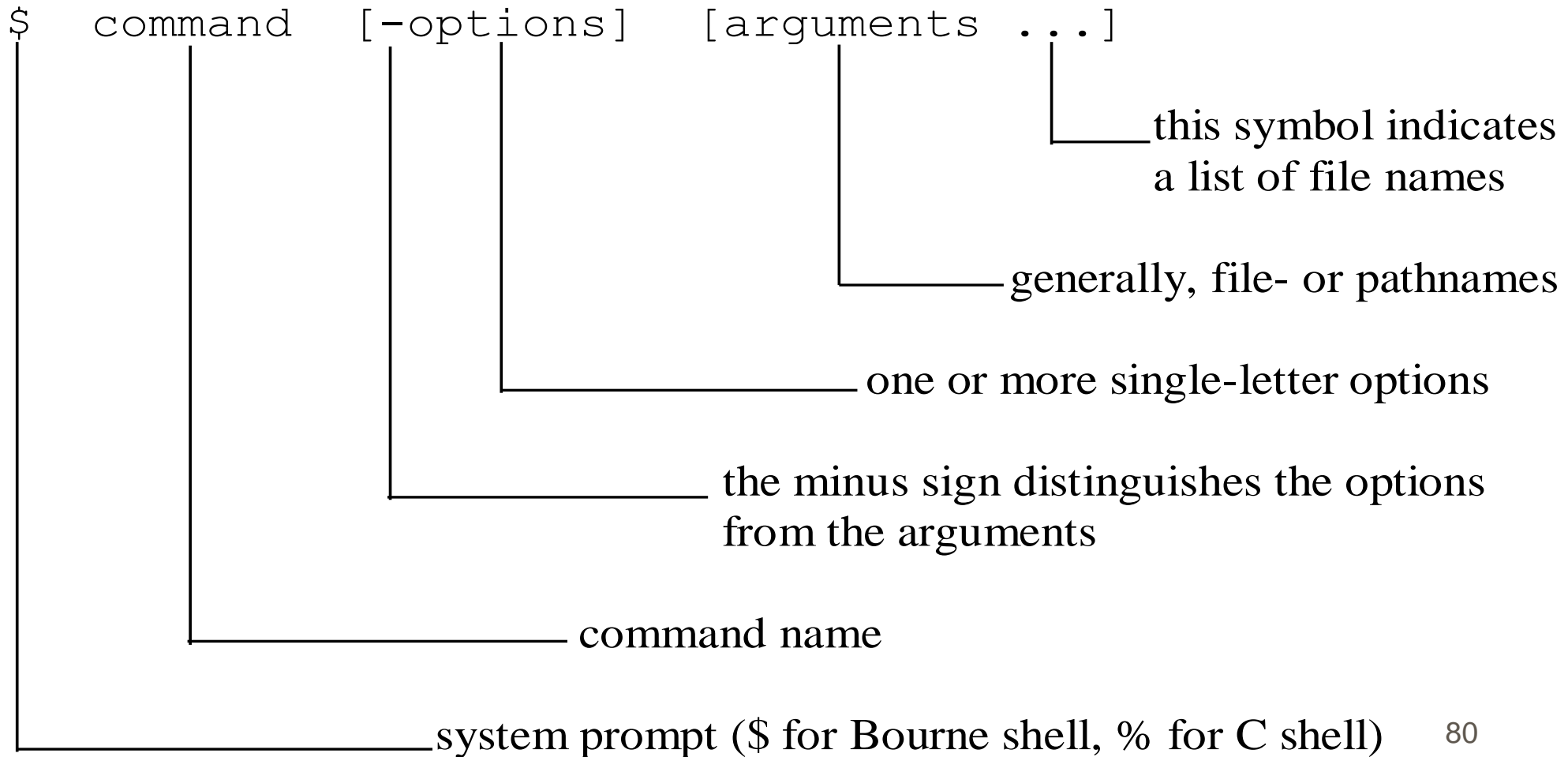
---



**Figure 1-6:** Shell's relationship to the user and the hardware

# Typical Command Format

---





# UNIX/Linux常用命令

---

- |           |          |
|-----------|----------|
| ⌘ 1.cd    | 2. ls    |
| ⌘ 3.cp    | 4.mv     |
| ⌘ 5.rm    | 6.mkdir  |
| ⌘ 7.rmdir | 8.man    |
| ⌘ 9.more  | 10.cat   |
| ⌘ 11.echo | 12.mount |

# 1. cd

---

- ⌘ 功能：改变当前目录
- ⌘ 语法：**cd <dir>**
- ⌘ 说明：**dir** 是要切换的目标记录
- ⌘ 示例：
  - 📁 **cd /user**
  - 📁 **cd ..**      //向上一层目录

## 2. ls

---

- ⌘ 功能：列出文件或目录的信息。
- ⌘ 语法：**ls -s1 ...-sN <file1>...<fileN>**
- ⌘ 说明：**s1**到**sN**为命令开关。
- ⌘ 示例：
  - ☒ **ls -t** //按时间顺序列出文件，近在前
  - ☒ **ls -l** //长列表，包括较多信息：**ls -lt**
  - ☒ **ls -u** //按最后使用的时间次序列出文件
  - ☒ **ls -r** //以逆序列出文件 如：**-rt**

### 3. cp

---

- ⌘ 功能：拷贝文件到另一个文件或目录
- ⌘ 语法：**cp <file1>...<fileN> <target>**
- ⌘ 说明： **<file1>...<fileN>**是所拷贝的文件名称。
- ⌘ 示例：
  - ☞ **cp /home/jerry/paper project**
  - ☞ **cp try.c /mnt/floppy**
  - ☞ **cp try1.\* /mnt/floppy**

## 4. mv

---

- ⌘ 功能：移动文件到另一个文件或目录
- ⌘ 语法： **mv <file1>...<fileN> <target>**
- ⌘ 说明：该命令可以为文件重命名。
- ⌘ 示例：
  - 📄 **mv /home/jerry/paper project2**
  - 📄 **mv try.c /mnt/floppy**

## 5. rm

---

- ⌘ 功能：删除文件
- ⌘ 语法： **rm <file1>...<fileN>**
- ⌘ 说明：**Linux**下文件删除后不可恢复。
- ⌘ 示例：
  - 📄 **rm /home/jerry/paper**
  - 📄 **rm example.\***

## 6.mkdir

---

- ⌘ 功能：创建新目录
- ⌘ 语法： **mkdir <file1>...<fileN>**
- ⌘ 说明：可以一次创建多个新目录
- ⌘ 示例：
  - 📄 **mkdir /home/jerry/c\_dir**
  - 📄 **mkdir user**

## 7. rmdir

---

- ⌘ 功能：删除空目录
- ⌘ 语法：**rmdir <dir>**
- ⌘ 说明：被删除的目录必须是空的，不容许删除当前工作目录
- ⌘ 示例：
  - 📄 **rmdir /home/jerry/paper**
  - 📄 **rmdir user**



## 8. man

---

- ⌘ 功能：查阅制定命令或资源的联机手册。
- ⌘ 语法：**man <command>**
- ⌘ 说明：**command**是要查看有关帮助信息的命令名称。
- ⌘ 示例：
  - ⏏ **man ls**
  - ⏏ **man mkdir**
  - ⏏ 退出 **:q**

## 9. more

---

- ⌘ 功能：显示指定文件的内容，每次一屏幕。
- ⌘ 语法： **more <file1>...<fileN>**
- ⌘ 说明： **<file1>...<fileN>**是所要显示的文件
- ⌘ 示例：
  - ⌘ **more /home/jerry/paper**
  - ⌘ **more /usr/program.c**

## 10. cat

---

⌘ 功能：通常用于连接显示几个文件，或一次显示整个文件。

⌘ 语法： **cat <file1>...<fileN>**

⌘ 说明： **<file1>...<fileN>**是所要连接或显示的文件

⌘ 示例：

☞ **cat /home/jerry/paper**

☞ **cat a1 a2**

# 11. echo

---

- ⌘ 功能：回显所给参数
- ⌘ 语法：**echo <arg1>..<<argN>**
- ⌘ 说明：<arg1>..<<argN>要回显的参数。
- ⌘ 示例：  
    📦 **echo “Hello the world”**

## 12. mount (Linux常用)

---

- ⌘ **mount**功能：挂装文件系统（U盘）
- ⌘ 示例：**mount -t vfat /dev/sdb1 /mnt/sub**
- ⌘ 功能：挂装**USB**设备/文件系统。
- ⌘ 装载使用**mount**命令，U盘是**fat**的系统，所以在类型参数中，选择**vfat**。新挂装文件系统的名字：**/mnt/sub**
- ⌘ **umount**功能：拆除文件系统（U盘）
- ⌘ 示例：**umount /mnt/sub**
- ⌘ 卸载成功之后，**/mnt/usb**目录下不再有任何文件。

# Linux / UNIX下的 C语言程序设计

---

- ⌘ 标准C函数库
- ⌘ **ANSI C**不仅定义了C语言的语法和语义，而且定义了其标准库
- ⌘ 常用的：
  - ☒ `<ctype.h>`, `<errno.h>`, `<float.h>`, `<limits.h>`, `<math.h>`, `<signal.h>`, `<stddef.h>`, `<stdio.h>`, `<stdarg.h>`, `<stdlib.h>`, `<string.h>`, `<time.h>`等

# Linux / UNIX下的C

---

- ⌘ 1998年，IEEE制定“UNIX的标准接口和环境”POSIX1.0实现源代码式的兼容。
- ⌘ UNIX版本 system V(AT&T) , 4.3+BSD 几乎含ANSI C的大部分头文件并依从POSIX
- ⌘ 有头文件<sys / types.h>,  
⌘ <sys / times.h>,<sys/ wait.h>等
- ⌘ 进行UNIX的系统调用

# 系统调用与库函数

---

## ⌘ 系统调用：

☑ 所有**OS**都提供多种服务的入口点，由此程序向内核请求服务，经过入口点进入内核，这些入口点被称为系统调用（**system call**）。

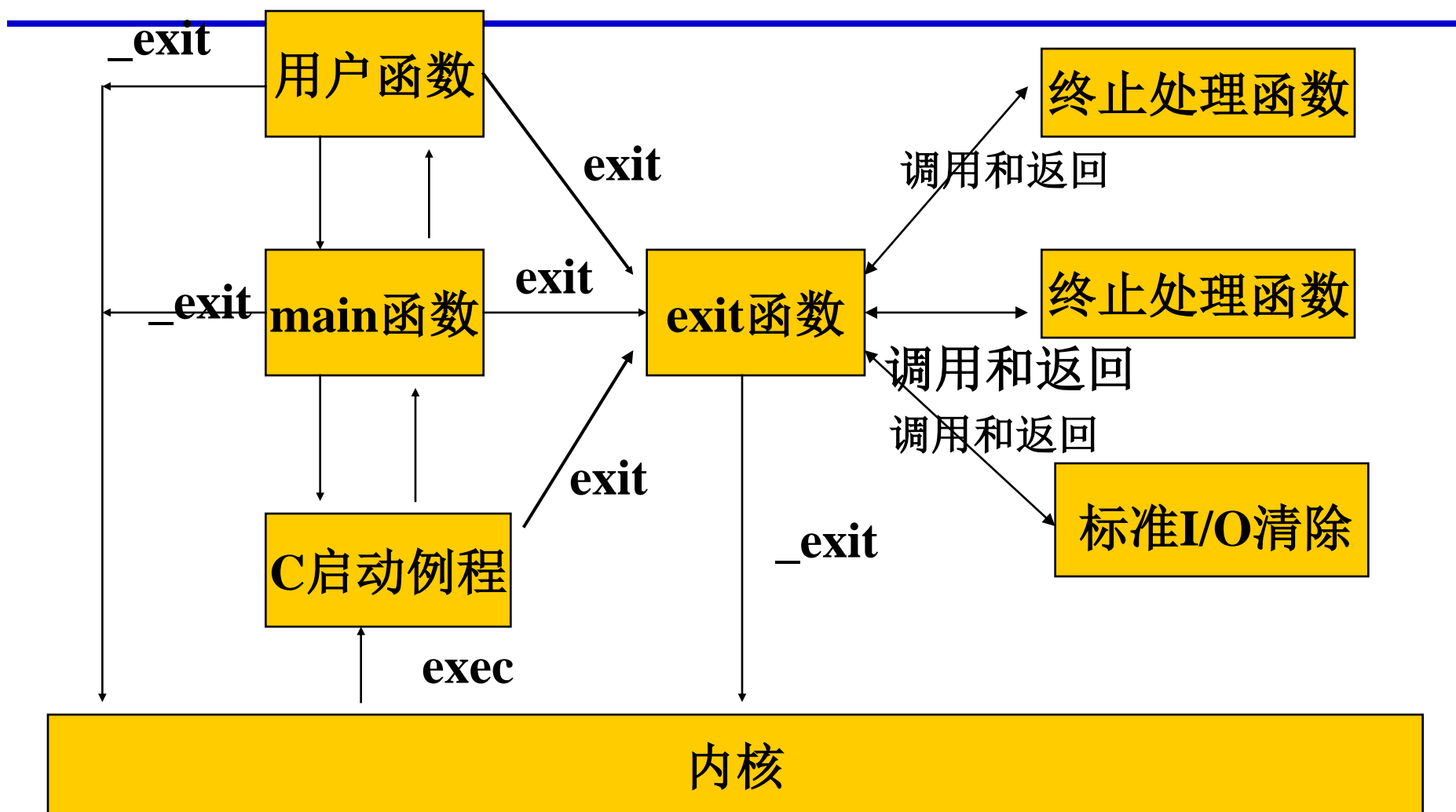
⌘ 每个系统调用在标准**C**库中具有同样名字的函数。  
（ **C**程序→函数→内核服务。）

⌘ 通用函数 ----- 一个或多个内核的系统调用。

⌘ 例： **malloc**函数-----**sbrk**系统调用



## C语言是如何启动和终止的图示



## C语言是如何启动和终止的说明

---

- ⌘ 说明: `_exit()` 立即进入内核
- ⌘ `exit()` 先执行一些清除处理, 然后进入内核。
- ⌘ 通常: C程序中使用 `return(0)` 或者 `exit(0)` 向执行此程序的进程访回终止状态0。
- ⌘ 在一些经典C程序中隐式返回。
- ⌘ 进程也可以非自愿地有一个信号使其终止。

# Linux / UNIX下的C语言程序（续）

---

- ⌘ 输入与运行标准C语言程序
- ⌘ 1. **vi**（或其他编辑软件）输入C文件
- ⌘ 2. **gcc 文件名.c** //编译C语言程序
- ⌘ 3. **./ a.out** //运行正确的C程序
- ⌘ （或）
- ⌘ 2. **gcc -o 可执行文件 文件名.c** //编译
- ⌘ 3. **./ 可执行文件** //运行
- ⌘ 如果使用C++语言编程工具，可以用**g++**编译。

# Linux / UNIX下的C语言程序（续）

---

## ⌘ 1. vi try.c //编辑文件try.c

⏏ #include <sys/types.h>

⏏ #include <stdio.h>

⏏ int main()

⏏ {

⏏ printf("try c .....\\n");

⏏ }

## Linux / UNIX下的C语言程序（续）

---

- ⌘ 2. `gcc try1.c` //编译try1.c语言程序
- ⌘ 3. `./a.out` //运行正确的try1的执行程序
- ⌘ （或）
- ⌘ 2. `gcc -o try1 try1.c` //编译
- ⌘ 3. `./try1` //执行文件
- ⌘ 结果：
  - ☐ try c .....

# 输入和输出

---

- ⌘ 1. 文件描述符：非负正数，内核用以标识一个特定进程正在访问的文件。打开或创建一个文件时，返回一个文件描述符。读写文件时可访问。
- ⌘ 2. 标准输入、输出
  - ☒ `fgets, getc, scanf`
  - ☒ `fputs, putc, printf`

# 输入和输出

---

## ⌘ 3.不用缓存的输入、输出

- ☒ **open, read, write, lseek, close**

- ☒ **例如：程序段**

- ☒ **while ((n=read(STDIN\_FILENO,buf,BUFSIZE))>0)**

- ☒ **if (write(STDOUT\_FILENO,buf,n)!=n)**

- ☒ **err\_sys("write error");**

- ☒ **if (n<0) err\_sys("read error");**

- ☒ **exit(0);**

- ☒ **// 头文件<unistd.h>**

- ☒ **//#define BUFSIZE 8190    int n ;**

# 程序和进程

---

- ⌘ 程序:存放在磁盘文件中的可执行文件。
- ⌘ 进程: 程序的执行。任务可表示正被执行的程序。
- ⌘ 进程 ID: 每个**UNIX**进程都有唯一的 进程 ID号。
  - ☒ **gepid()** 可打印进程ID号。
- ⌘ 程序与进程的区别
  - ☒ 程序: 静态。
  - ☒ 进程: 动态。程序+数据+**PCB**（动态标识）



# 程序和进程

---

```
⌘ #include <unistd.h> // 打印进程 ID
⌘ int main()
⌘ {
⌘ printf("hello world from process ID%d\n",getpid());
⌘     exit(0);
⌘ }
⌘ 结果:
    ☒ $a.out
    ☒ hello world from process ID 851
    ☒ $a.out
    ☒ hello world from process ID 854
```

# 用户标识

---

## ⌘ 用户ID

- ☑ 标识各个不同的用户。用户不能更改其用户ID
- ☑ ID=0 为根 (root) 或超级用户 (superuser)
- ☑ `getuid()` `getgid()` 返回用户ID和组ID

## ⌘ 组ID

- ☑ 口令文件登录项也包含用户的组ID。UNIX中组被用于将若干用户集合到项目和部门中去。

## 用户标识（续）

---

⌘ //打印用户ID和组ID

⌘ #include "ourhdr.h"

⌘ Int main( )

⌘ {

⌘   printf("uid=%id,gid=%d\n",getuid( ),getgid( ));

⌘ exit(0);

⌘ }

⌘ \$./a.out

⌘ uid=224 ,gid=20