综述

第13章　计算机专业英语

本章主要讲解计算机专业英语。

13.1　综述

英语能力是软件设计师的必备能力，因此，计算机英语是软件设计师考试　　。考试大纲要求"具有工程师所要求的英语阅读水平，理解本领域的英语术语".在软件设　　式题中，共75分，其中英语占5分（2007年以前英语占10分）。

1.软件设计师英语考试与其他英语考试的比较

近几年的软件设计师英语考试主要有如下几方面的特点。

难度略高于大学英语四级，相当于研究生入学考试。

题材限于计算机文化读物，不如其他英语考试广泛。

题型只限于短文填空（完型填空），题型单一。

2.复习与应试要点

根据考试试题的特点，软件设计师英语复习要点如下。

找一本研究生入学考试（或四级）英语复习资料，复习相关的固定搭配、短语、语法知识，重点复习其中的完型填空，掌握完型填空的考点及要求。

注意多读计算机报刊、杂志的时文，在了解这个领域最新信息的同时积累语言知识，训练阅读能力。在复习时看一些计算机英语材料，对这一领域的表达方式和词汇进行热身。本章我们精选了一些英语材料，供大家复习参考。

用近几年的软件设计师英语考试试题进行模拟测试，本章收集了近几年的试题。

由于软件设计师英语考试题型只限于短文填空（完型填空），因此，自己可以在考前作一些专项练习，结合复习总结出一些解题的技巧。一般可采用三步法，其要点如下。

粗略地看一遍全文，了解全文的信息。

以了解的信息作为基础，对全文进行精读，并进行完型填空。

从全局的角度，对答卷进行检查。

试卷分析

13.2　试卷分析

从1991年起，软件设计师计算机专业英语部分的考查方式为完型填空，延续至今。

从广度上看，考查的题材比较新颖，内容广泛，涵盖了计算机的几个主要领域。

涉及到网络的：2006年11月考网络访问控制，2006年5月考Cookies,2005年5月考DOM,2003年考网络协议，2003年考浏览器，2001年考数据包裹，2000年考VOIP（IP话音业务）解决方案；

涉及面向对象知识的：2008年12月考UML,2008年5月考面向对象分析，2007年11月考RUP;

涉及到程序语言及其基础理论的：2002年考强制型语言，1998年考字处理拼写检查，1996年考Java;

涉及到信息安全的：2005年11月考数字认证，2005年5月考邮件病毒，2002年考计算机系统攻击，2000年考防火墙；

涉及到计算机系统结构的：2001年考多指令多数据流系统，1992年考精简指令集对复杂指令集的争议；

涉及到数据库的：1997年考关系型数据库模型，1994年考面向对象的数据库管理系统；

涉及到操作系统的：1995年考微内核技术，1994年考Windows NT操作系统；

涉及到新的应用技术和领域的：2006年11月考虚拟化，1995年考可视化的开发工具，1993年考移动式计算机，1992年考人工智能硬件；

涉及到计算机文化和产业的：1999年考信息产业，1991年考计算机文化教育产业。

但从深度上看，对计算机知识考得都比较浅显，多是一些IT方面的时文摘要。考查的力度仅仅体现在"了解，知道"这个层次上，没有对考生做更深入的要求。

从考查的内容来看，不仅仅考查计算机知识。还考一些英语基础，如：查上下文和主题、词形的辨析、词义的辨析、词性的辨析、考时态、考语态和分词、介词及其固定搭配等。

因此在准备这部分考试内容的时候，不应该仅仅局限于计算机领域词汇的记忆。考试大纲要求"具有工程师所要求的英语阅读水平，理解本领域的英语术语".这要求考生在语言的基本功上应该有所积累，如大学英语四六级，研究生英语考试考纲要求的词汇，相关的固定搭配、短语，相关的语法知识，这些都是"正确阅读和理解文章"的本钱。除此之外，还应该注意多读计算机报刊、杂志的时文，在了解这个领域最新信息的同时积累语言知识，训练阅读能力。最好还能辅以一些英语短文完型填空的练习，自己总结出一些解题的技巧。从这几年来的试卷和考试大纲来看，主要考查两个方面。

考查的计算机专业词汇及相关知识，这方面，试卷既体现了"日新月异"的特点，又选择了相对稳定和主流的内容，没有刻意地求"新"和求"偏".笔者根据命题的趋势，从近年来较新的计算机文献中挑选了一些专业词汇和缩略语作为附录。既注意了"新",又兼顾了"全",剔除了过分陈旧和过分偏颇的词汇。在今后几年可能会考查到，或者作为文章的题材涉及到。亦可供平时查阅之用，免去考生自己收集整理之繁琐。希望对大家有所帮助。

考查"正确阅读和理解计算机领域的英文文献",本章中编者精选了一些最新英语素材，供考生复习时参考。

例题分析

13.3 例题分析

例题1（2011年5月试题71~75）

Ravi, like many project （71） ,had studied the waterfall model of software development as the primary software life-cycle （72） .He was all set to use it for an upcoming project, his first assignment.However, Ravi found that the waterfall model could not be used because the customer wanted the software delivered in stages, something that implied that the system had to be delivered and built in （73） and not as （74） .

The situation in many other projects is not very different. The real world rarely presents a problem in which a standard process, or the process used in a previous project, is the best choice.To be the most suitable, an existing process must be （75） to the new problem.A development process, even after tailoring, generally cannot handle change requests.To accommodate change requests without losing control of the project, you must supplement the development process with a requirement change management process.

（71）A.customers B.managers C.users D.administrators

（72）A.activity B.procedure C.process D.progress

（73）A.parts B.modules C.software D.a whole

（74）A.parts B.modules C.software D.a whole

（75）A.modified B.used C.suited D.tailored

例题分析：

文章大意如下：

Ravi就像很多研究过以瀑布模型为软件生命周期过程的软件开发项目经理一样，他被安排使用瀑布模型去开发一个即将启动的项目，而且这是他的第一个任务。然而，Ravi发现不能在项目中使用瀑布模型，因为客户想要该软件分阶段交付，而不是作为一个整体交付。

在很多其他的项目中也有类似的情况，现实生活中，本来就很少有能完全按标准来进行处理的问题，可能某标准处理前一个问题非常合适，但处理现在这个问题就不定适合了。最合适的方法就是对一个新的问题必须采用切合它自身的方法。为了适应变化的变更请求而不失去对项目的控制，你必须要支持项目的发展过程与一个需求变更管理过程。

例题答案：（71）B （72）C （73）A （74）D （75）D

13.4 阅读素材

这里结合考试内容的特点，精选了一些计算机英语的阅读素材，供考试复习参考。为培养考生的阅读习惯与能力，提高阅读水平，这里就不再提供译文，必要时读者可以借助词典。

（1）Unix was originally developed at Bell Laboratories as a reaction to the large, complex, non-portable operating systems of the early 1970s. AT&T did not see a significant opportunity in licensing or supporting an operating system; Instead, it provided Unix source code for a nominal license fee and small per-unit royalties. A number of academic efforts sprang up to take advantage of this opportunity and extend the original sources with new and interesting features. In the mid-1980s, every computer manufacturer either provided or planned to provide a Unix-based operating system for its computers. Each company had chosen a particular version of Unix to start with, and then added various proprietary features. Although all of these operating systems claimed to be Unix, software written on one version was often not portable to the other versions.

（2）Many believed this effort by AT&T was bad news for the software community. Among them was Richard Stallman, a programmer at MIT, who in 1984 founded the GNU Project（"GNU is Not Unix"）as an effort to rewrite Unix as "free" software-that is, software not controlled by any one person and available for anybody to use and modify as desired. This led to the creation in 1985 of the Free Software Foundation（FSF）, the main funding and advocacy organization for the GNU Project.The Free Software Foundation established the GNU General Public License（GPL）, the most important open source license in use today. While the GNU Project was getting started, other efforts began to at least standardize the definition of Unix to enhance software portability. One such effort was the formation of the IEEE 1003 standards committee, which would go on to produce the set of standards known as the Portable Operating System Interface（POSIX）。Another effort of note was the creation of the X/Open organization, a commercial venture oriented toward application portability, testing, and branding of Unix-compatible operating systems.

（3）The concept of a storage device has changed dramatically from the first magnetic disk drive introduced by the IBM RAMAC in 1956 to today's server rooms with detached and fully networked storage servers. Storage has expanded in both large and small directions-up to multi-terabyte server appliances and down to multi-gigabyte MP3 players that fit in a pocket. All use the same underlying technology-the rotating magnetic disk drive-but they quickly diverge from there.

（4）SAN relies on each server connecting to it to provide security and management of the portion of the storage pool assigned to it. Only recently are techniques being introduced that allow storage pools to be shared among servers, but this is usually only when the host servers are tightly clustered. The key is that the traditional SAN system is trying to present a view of direct-attached, dedicated disks to each server while pulling out

the reliability, backup, and disk management issues from single-server control. SAN is geared for raw performance and generally achieves better performance than direct-attached disks as a result of larger caches and buffering. In fact, early SANs had SCSI connections to the hosts, as well as to the back-end disks.

(5) The overwhelming majority of information in an enterprise is unstructured-that is, it is not resident in relational databases that tabulate the data and transactions occurring throughout the enterprise. This unstructured information exists in the form of HTML pages, documents in proprietary formats, and forms （e.g., paper and media objects ）。 Together with information in relational and proprietary databases, these documents constitute the enterprise information ecosystem.

(6) It is well known that some of the most valuable knowledge in an enterprise resides in the minds of its employees. Enterprises must combine digital information with the knowledge and experience of employees. An important distinction between the enterprise and the Internet is that while Internet users are anonymous for the most part, enterprise users are answerable and guided by specific controllable processes. Privacy issues are also very different in an enterprise, since people are usually engaged in enterprise-specific behavior and are being compensated for their engagement.

(7) In past decades, "Moore's law" has governed the revolution in microelectronics. Through continuous advancements in device and fabrication technology, the industry has maintained exponential progress rates in transistor miniaturization and integration density. As a result, microchips have become cheaper, faster, more complex, and more power efficient.

(8) Much of the performance gain in digital hardware can be traced to corresponding advances in integrated circuit technology. For example, over the past 30 years we have witnessed a steady decrease in transistor channel length by two orders of magnitude from 10?m in 1970 to less than 0.1?m today. While this reduction of feature size benefits analog and digital circuits alike, overall analog circuit performance is compromised by other trends such as reduced supply voltages.

(9) This is the new world order: software development on a global scale. As is almost always the case, this doesn't mean the entire technical arm of a corporation breaking off and floating across an ocean in one clean clump, but rather numerous small teams here and there "sharing the pain" across time zones. Nowadays it's not atypical to have multiple teams on multiple continents working on the same project. It's not just financial pressures that come to bear and break developers into teams. The sheer size-raw lines of code-of many of today's projects means they're just too big for one group to handle on its own. And whether teams are spread across a campus or across multiple continents, distributed development affects the way that we write code.

(10) Individuals from different cultures are also likely to be motivated differently. In

countries where individualism is valued, people seek material gain and personal recognition. Countries that emphasize the collective rather than the individual tend to value time for personal relations, family, and so forth, over material gain. For them the goal is to preserve social equilibrium, not to "rock the boat." And, for them the greatest punishment is ostracism. Team incentive systems should take these values into account, rewarding U.S. developers with money and French developers with time off.

（11）Systems such as Lotus Notes IM allow people to contribute to discussions remotely either synchronously or asynchronously. In cultures where meetings are the venues for confirming, rather than making, a decision, the actual content of the discussion is not visible to everyone concerned. In contrast, online databases show all the discussion, displaying various pros and cons, signed by the contributors. They are task-based rather than relationship-based in their underlying structure. It is easy to see that this kind of technology will be less favored by cultures that value "face" and relationships over task orientation.

（12）People are not well represented in today's software. With the exception of IM（instant messaging）clients, today's applications offer few clues that people are actually living beings. Static strings depict things associated with people like e-mail addresses, phone numbers, and home-page URLs. Applications also tend to show the same information about a person, no matter who is viewing it. This information does not change, at least not very rapidly. If your only exposure to people were through these strings, you would have little reason to believe that people actually move about in physical and virtual space and do things.

（13）It has been more than ten years since such "information appliances" as ATMs and grocery store UPC checkout counters were introduced. For the office environment, Mark Weiser began to articulate the notion of UbiComp（ubiquitous computing）and identified some of the salient features of the trends in 1991. Embedded computation is also becoming widespread.

（14）Microprocessors, for example, are finding themselves embedded into seemingly conventional pens that remember what they have written. Anti-lock brake systems in cars are controlled by fuzzy logic. And as a result of wireless computing, miniaturization, and new economies of scale, such technologies as PDAs（personal digital assistants）, IM（instant messaging）, and mobile access to the Internet are almost taken for granted.

（15）Ubiquitous computing is in some ways an everyday reality. However, cooperative ubiquitous computing is still in its infancy. New forms of interaction must be developed for this environment-interaction between two or more parties: people and people（both technologically mediated and not）, people and machines, and machines and machines. Implicit in this formulation is the importance of location. Previously, transactions took place where the computer was anchored. The location of the computer was not a design issue.

Now distance （both physical and social） and location are key considerations in understanding and designing systems.

（16）Containers, or wireless mobile devices, primarily serve as a mechanism for easily transporting data identifiers among terminals. Sample containers include PDAs, cell phones, bar-code readers, and Smart Cards （see figure 3）。 Some devices can be both a container and a terminal. These types of devices not only hold and transport an identifier, they can also allow some interaction with the associated data. For example, a PDA transporting an image identifier can also display and allow for machine manipulation of a version of the image itself. A container can also work in concert with a terminal, serving as an extension of the terminal's user interface. This is particularly useful when working with terminals that have limited input functionality.

（17）The hemispherical display surface provides the viewer with a greater sense of immersion than a typical flat-screen display. When viewing designs for the interior of cars, for example, this enhanced sense of immersion provides a better idea of what it would be like to actually sit inside the car. Furthermore, since this immersion is facilitated without encumbering stereoscopic hardware, subtle human body-language cues, such as eye gaze, are not obscured. Viewers' ability to interact with one another while using the terminal is thus uncompromised. However, easy interaction with the surface of the display itself is precluded by the size and shape of this terminal, and the fact that viewers should stand several feet away from the display to get maximum immersion. To counteract these factors, we provide an auxiliary 15-inch touch-screen display, mounted at waist height in front of the terminal, to serve as an interaction portal.

（18）To some degree, data access methods have been rooted in the metaphor of accessing files in a hierarchical filesystem. Technological developments such as wireless networks, mobile computing devices, and specialized display terminals can be used to present a different, and possibly more effective, user model for data access in a modern cooperative ubiquitous computing environment. We have proposed a user model called "sentient data access," which utilizes access context, location, and user information.

（19）In October of 2002, Sun introduced the Sun ONE Application Server version 7.0. This key ingredient of the Sun ONE platform is tightly integrated with the new Sun ONE Studio for Java 2 Enterprise Edition 4.1 set of web services tools. Sun also introduced a new business model that offers the core version （Platform Edition） free to enterprises and independent software vendors （ISVs） on all leading platforms. With its new modular architecture, the application server increases options, as well as return on investment （ROI）， in rapidly building and deploying Sun ONE Java web services.

（20）In addition to all the aforementioned strengths of J2EE, one of the key markets of tomorrow is the Wireless market, where Java technology-enabled devices from handset manufacturers such as Nokia, Ericsson, Motorola, Sharp among others, are being deployed

by key carriers such as Nextel, NTT Docomo and LG Telecom. More deployments and handset models will be unveiled shortly. This market is so huge that there are about 1 billion Java enabled handsets to be available in the next 2 or 3 years, and these devices#151;mobile phones, pagers, PDAs, automotive systems are all joining the traditional client, the personal computer, on the Net. They represent new platforms–always on, always with you–that are opening up exciting new opportunities for networked applications and services.

（21）If data is placed into the memory cache before being written permanently to the disk drives, then this memory must be protected to keep the data safe and recoverable after a power failure or other catastrophic error. Doing so allows the storage device to acknowledge a user write request as soon as the data enters the cache, rather than waiting for it to be permanently written. This greatly reduces latency, which is often the critical performance point in a system trying to improve fault tolerance.Implementing immediate acknowledge caches requires carefully designed hardware, software, and protocols for redundancy and error recovery in the various system components, such as multiple server processors, multiple paths to the cache, and potentially multiple batteries.This is further complicated if caching is performed at multiple layers in the storage system with each point immediately acknowledging the write. It quickly becomes difficult to prove strong statements about the overall reliability of the stored data.

计算机专业英语词汇及缩略语精选

13.5 计算机专业英语词汇及缩略语精选

说明：计算机领域内的很多词汇的形式尚无统一规定。为统一起见，这里列出的词汇尽量去掉了时态、分词形式、复数等。对于一词多义的，尽量列出其最主要的、常用的意思，以及在计算机领域内特定的意思。但对于某些词汇的分词、复数等形式在计算机领域中表达特定意义的，则做了保留。

常见计算机词汇

13.5.1 常见计算机词汇

Abstract	抽象的
Abstraction	抽象
Acceptance test	验收测试
Acceptor	接收器
Access control	访问控制
Activation	活跃期
Active object	主动对象
Activity diagram	活动图
Actor	参与者
Actuator	传动器
Adapter	适配器
Addressing	寻址
Agent	代理
Aggregation	聚合
Agile Methodologies	敏捷方法学
Algebra	代数学
Algorithm	算法
Allocation	分配
Alphabet	字母表
Alphabetize	按字母顺序
Amplify	放大
Animation	动画
Antenna	天线
Architecture	构架
Argument	引数
Aspect oriented	面向方面的
Assembler	汇编程序
Assertion	断言
Assessment	评估
Association rule	关联规则
Association	关联
Assumption	假设
Asymmetric key encryption	非对称密钥加密
Atomicity	原子性
Attack tree	攻击树
Attribute	特性

Authentication 认证

Automation 自动控制化

Backdoor 后门

Backup 备份

Barrier 隔离层、隔离物

Baseline 基线

Batch 批

Binary 二进制

Black box testing 黑盒测试

Bluetooth 蓝牙技术

Boolen algebra 布尔代数

Bottleneck 瓶颈

Breakpoint 断点

Bridge 网桥

Broadband 宽带

Buffer 缓冲区

Bug 缺陷

Bundle 捆绑

Business 业务、商业

Cable 电缆

Cache 高速缓冲存储器

Calculator 计算器

Call back 回调

Catalog 目录

Category 范畴

Certification 认证

Channel 信道

Class diagram 类图

Cleanroom software engineering 净室软件工程

Clipboard 剪贴板

Cohesion 内聚

Collaboration diagram 协作图

Collaboration 协作

Combinatory Mathematics 组合数学

Commerce 商务

Commit 提交

Compact 紧凑的

Compatibility 兼容性

Compile	编译
Compiler	编译器
Component	组件
Composite	复合
Computation	计算
Conceptual design	概念设计
Concurrent	并发的
Confidential	机密的
Configuration	配置
Congestion	拥挤、阻塞
Connection pool	连接池
Connector	连接件
Consistency	一致性
Console	控制台
Constrain	约束
Container	容器
Context	上下文
Coordinate	坐标
Copyright	著作权
Counter	计数器
Coupling	耦合
Cracker	骇客
Critical path	关键路径
Critical section	临界区
Crosscut	横切
Crystal	水晶、水晶方法
Data mining	数据挖掘
Data warehouse	数据仓库
Datagram	数据报
Debug	调试
Decision theory	决策理论
Decision tree	决策树
Decompile	反编译
Decryption	解密
Definition	定义
Delegate	代理、委托
Delegated administration	委托管理
Demo	样本

Demodulation 解调

Dependency 依赖

Deployment 部署

Derive 派生

Descriptor 描述符\描述器

Design by contract 契约式设计

Design pattern 设计模式

Diagnostics 诊断

Digital certificate 数字证书

Digital signature 数字签名

Disassemble 反汇编

Discrete mathematics 离散数学

Divergent 分歧

Dizzy 混乱的

Documentation 文档

Domain model 域模型

Domain-specific 领域相关的

Dot product 点积

Driver 驱动程序

Duplex system 双工系统

Duplex 双工

Durability 持久性

Dynamic 动态

Electronics 电子学

Element 元素

Embedded system 嵌入式系统

Emulation 仿真

Encapsulation 封装

Encryption 加密

Engine 引擎

Entity 实体

Ethernet 以太网

Euclidean space 欧氏空间

Even 偶数、偶校验

Evolutionary 进化的

Exception 异常

Executable 可执行的

Extension 扩展

Extract 提取

Extranet 外联网

Facsimile 传真

Fault tree 错误树

Fault-tolerant 容错

Feasibility 可行性

Feedback 反馈

Field 字段

Filter 过滤

Floppy disk 软盘

Flow chart 流程图

Flow control 流量控制

Foreign key 外键

Format 格式、格式化

Framework 框架

Frame 帧

Frequency 频率

Function overloading 函数重载

Function 功能、函数

Functional testing 功能测试

Fuzzy 模糊的

Game theory 对策论

Gantt chart 甘特图

Gateway 网关

Generative 再生的

Generic programming 泛型编程

Generic 泛型

Geometric 几何的

Global 全局的

Granularity 粒度

Graph theory 图论

Grey box testing 灰盒测试

Grid 网格

Guaranteed delivery 可靠性传输

Hacker 黑客

Handle 句柄

Handwriting recognition 手写识别

Harness 约束

Hashtable 哈希表

Heap 堆

Hierarchical 层次的、体系的

High availability 搞可用性

Hook 钩子

Human factors engineering 人因工程

Hybrid programming 混合编程

Hypermedia 超媒体

Hypertext 超文本

Hypothetical 假定的

Icon 图标

Identifier 标识符

Imagebase 基地址

Increment 增量

Incremental integration testing 组合测试

Infer 推理

Information hiding 信息隐藏

Infrastructure 下部构造、基础的

下部组织

Inheritance 继承

Initialize 初始化

Install 安装

Instance 实例

Instruction 指令

Integration testing 集成测试

Integration 集成

Intelligence 智能

Intensity 强调

Interceptor 拦截器

Intermediate 中间的

Internationalization 国际化

Interpret 解释

Intranet 内部网、企业网

Inversion 反转

Invoke 调用

Isolation 孤立性

Isomorphic 同构的

Iterative 迭代

Iterator	迭代器
Join point	连接点
Kernel	内核
Large-scale	大规模的
License	许可证
Life cycle	生命周期
Lifeline	生命线
Linear	线性的
Linearization	线性化
Linguistics	语言学
Liquid-crystal	液晶的
Load testing	负载测试
Load-balanced	负载均衡的
Location	定位
Log	日志
Logics	逻辑学
Macro	宏
Magnetic	磁性的
Maintenance	维护
Managed execution	托管执行
Manual	手册
Mapping	映射规则
Marshalling	编组、封送
Matrix	矩阵
Mechanism	机制
Mentor	导师
Merge	归并
Method	方法
Microwave	微波
Middleware	中间件
Migration	移植
Mirror	镜像、镜子
Modem	调制解调器
Modulation	调制
Module coupling	模块耦合
Module	模块、组件
Monitor	监视器
Motherboard	主板

Multiprogramming 多道程序设计

Multithreading 多线程

Mutation 变异

Namespace 名字空间

Natural language 自然语言

Navigation 定位、航行

Neural network 神经网络

Novice 初学者

Number theory 数论

Numerical computation 数值计算

Open source 开放源代码

Operator 操作符

Optical fiber 光纤

Optical 视力的、光学的

Optimization 优化

Orthogonal 正交

Outsourcing 外包

Over-engineering 过度设计

Overflow 溢出

Overload 重载

Override 覆盖

Package 包

Pair programming 结对编程

Panel 面板

Paradigm 泛例

Parameter 参数

Parity 奇偶

Pattern matching 模式匹配

Peer-to-peer computing 对等计算

Performance testing 性能测试

Performance 性能

Peripheral 外设

Persistence 持久性

Personalize 使个性化

Pipelining 流水线

Pixel 像素

Platform independent 平台无关

Platform invoke 平台调用

Platform 平台

Plug-in 插件

Pointer 指针

Policy 策略

Polymer 聚合体

Polymorphism 多态

Port 端口

Portability 可移植性

Portal 门户

Prepositional logic 命题逻辑

Preprocessor 预处理程

Primary key 主键

Priority 优先权

Probability theory 概率论

Procedure 过程

Process 处理

Processor 处理器

Production 产生式、成果、
产品

Profile 框架、轮廓

Projection 投影

Property 属性

Protocol 协议

Prototyping development approach型化开发方法

Proxy 代理

Pruning node 修剪结点

Pseudocode 伪代码

Quota 定额

Reactor 反应器

Real-time 实时的

Recovery testing 恢复测试

Redundancy 冗余

Refabricate 重构

Reference type 应用类型

Reference 引用

Referential integrity 参照完整性

Reflection 反射

Register 注册

Regular expression 正则表达式

Relational algebra 关系代数

Relational databases model 关系数据库模型

Release 发布

Remote 远程的

Repeater 中继器

Replication 复制

Repository 数据仓库、仓库

Resident 常驻的

Resolution 分辨率、决定的

Responsiveness 响应

Retrieve 检索

Reusability 复用性

Reverse engineering 逆向工程

Robot 机器人

Robust 健壮的

Rollback 回滚

Router 路由器

Sandbox 砂箱

Satellite 人造卫星

Scan 扫描

Scheduling 调度

Schema 模式、结构、方案

Scheme 方案、系统

Script 脚本

Search engine 搜索引擎

Security testing 安全测试

Security 安全性

Segment 段

Semantic 语义的

Semiconductor 半导体

Sensor 传感器

Sequential 顺序的

Serial 串行的

Serialize 串行化

Server cluster 服务器集群

Set theory 集合论

Set-top box 机顶盒

Shading 投影

Shareware 共享软件

Side effect 副作用

Signature 签名

Silicon 硅

Simplex 单工

Simulation 模拟

Simultaneous 同步的

Smart pointer 指针

Sockets layer 套接层

Software reuse 软件复用

Solution 解决方案

Sophisticated 高级的、复杂的

Speech recognition 语音识别

Speech synthesis ? 语音合成

Spiral model 螺旋模型

Spreadsheet 图表

Spyware 间谍软件

Stack 栈

Standardize 使标准化

Statistical 统计的

Stored procedure 存储过程

Strategy 策略

Stream 流

Stress testing 压力测试

String 串

Stub 存根

Subject 主体

Subnet 子网

Substantial 实质的

Supercomputer 超级计算机

Symbol 符号

Synchronize 使同步

Syntactic 语法的

System analyst 系统分析员

System testing 系统测试

Template 模板

Terminal 终端

Terminology 术语

Tertiary 第三方的

Test case 测试用例

Test Driven development 测试驱动开发

Thread pool 线程池

Thread 线程

Threshold 阈值

Throughput 吞吐量

Time-slicing 时间片

Token 令牌

Top-down programming 自顶向下程序设计

Topology 拓扑（结构）

Tow-way 双向的

Track 追踪

Transaction 事务

Transformation 转换

Transistor 晶体管

Trigger 触发器

Tuple space 元组空间

Unicode 国际双字节编码

Uninstall 卸载

Unit testing 单元测试

Unmarshalling 反编组、拆收

Upward compatible 向上兼容的

Use case 用例

User identity 用户身份认证

Utility 效用、工具

Vacuum tube 真空管

Value chain 价值链

Variable-length array 可变长数组

Variance 变动、协变

Vector 矢量

Velocity 速率

Vibration 震荡

View 视图

Violation 冲突

Virtual memery 虚拟内存

Virtual 虚拟的

Virus 病毒

Visual 可视化的

Wafer 晶片

Waterfall method 瀑布方法

Webservice Web服务

White box testing 白盒测试

Workflow 工作流

Workplace 工作区

Workstation 工作站

Worm 蠕虫病毒

常见计算机缩略语

13.5.2 常见计算机缩略语

3D（three dimension）： 三维

ACE（adaptive communication environment）： 可适配通信软件开发环境

ACM（association for computing machinery）：美国计算机学会

ADO（ActiveX data objects）： ActiveX数据对象

ADSL（asymmetrical digital subscriber line）： 非对称数字用户环路

AI（artificial intelligence）： 人工智能

AMI（asynchronous message invocation）： 异步消息

ANSI（American national standards institute）： 美国国家标准化协会

AOP（aspect oriented programming）： 面向方面编程

AP（application plan）： 应用程序规划

API（application programming interface）： 应用编程接口

ARP（address resolution protocol）： 地址解析协议

ASCII（American standard code for information interchange）：美国国家信息交换标准码

ASD（adaptive software development）：　自适应软件开发

ASP（active server page）：　动态服务器页技术

ATM（asynchronous transfer model）：　异步传输模式

B/S（browser/server）：　浏览器/服务器结构

B2B（business to business electronic commerce）：　企业对企业的电子商务

B2C（business to consumer electronic commerce）：　企业对客户的电子商务

C/S（client/server）：　客户端/服务器结构

CAD（computer aided design）：　计算机辅助设计

CASE（computer aided software engineering）：　计算机辅助软件工程

CDMA（code division multiple access）：　码分多址技术

CGA（color graphics adapter）：　彩色图形适配器

CIM（computer –integrated manufacturing）：　计算机集成制造技术

CISC（complex instruction set computer）：　复杂指令集计算机

CLI（common intermediate language）：　通用中间语言

CLR（common language runtime）：　公共语言运行环境

CLS（common language specification）：　公共语言规范

CMM（capability maturity model）：　能力成熟度模型

CMMI（capability maturity model integration）：　能力成熟度模型综合

CMP（container managed persistence）：　容器管理数据一致性

COM（component object model）：　组件对象模型

CORBA（common object request broker architecture）：公共对象请求代理体系结构

CRC（cyclic redundancy check）：　循环冗余校验

CRM（customer relationship management）：　客户关系管理

CSMA/CD（carrier sense multiple access collision detect）：载波侦听多路访问/冲突检测

DAO（data access object）：　数据访问对象

DBA（database administrator）：　数据库管理员

DBMS（database management system）：　数据库管理系统

DCE　（distributed computing environment）：　分布式计算机环境

DCOM　（distributed component object model）：　分布式组件对象模型

DFA　（deterministic finite automaton）：　确定有限状态自动机

DFD　（dataflow diagram）：　数据流图

DHTML（dynamic hypertext markup language）：　动态超文本标记语言

DLL　（dynamic-link library）：　动态链接库

DNS　（domain name System）：　域名系统

DoS（denial of service）：　拒绝服务攻击

DSDM（dynamic system development method）：　动态系统开发方法

DSS （decision support system）： 决策支持系统

ECC（error correction）： 纠错码

ECO（enterprise core object）： 企业核心对象

EDI（electronic data interchange）： 电子数据交换

EJB（enterprise javabean）： 企业Javabean

ERD（entity-relationship diagram）： 实体联系图

ERP（enterprise resource planning）： 企业资源计划

ES（expert system）： 专家系统

FAT（file allocation table）： 文件分配表

FDD（feature-driven development）： 特征驱动开发

FIFO（first-in first-out）： 先进先出

FTP（file transfer protocol）： 文件传输协议

GA（genetic algorithm）： 遗传算法

GC（garbage collection）： 内存垃圾收集

GIS（geographic information system）： 地理信息系统

GPS （global positioning system）： 全球定位系统

GSM（global system for mobile communication）：全球移动通信系统

GUI（graphics user interface）： 图形用户界面

HTML（hypertext markup language）： 超文本标记语言标准

HTTP （hypertext transfer protocol）： 超文本传输协议

IC（integrated circuit）： 集成电路

ICMP（Internet control message protocol）： 网际报文控制协议

IDE（integration development environment）： 集成开发环境

IDS（intrusion detection system）： 入侵检测系统

IEEE（institute for electrical and electronic engineers）：美国电气电子工程师学会

IGMP（Internet group multicast protocol）： 网际成组多路广播协议

IP（nternet protocol）： 网际协议

IPC（interprocess communication）： 进程间通信

IPS（intrusion prevention system）： 入侵防护系统

ISA（industry standard organization）： 工业标准化组织

ISDN（integrated services digital network）： 综合数字业务网

ISO（international organization for standardization）：国际标准化组织

ISP（Internet service provider）： 因特网服务提供商

J2EE（Java 2 enterprise edition）： Java 2企业版

J2ME（Java 2 micro edition）： Java 2袖珍版

J2SE（Java 2 sdk standard edition）： Java 2标准版

JDBC（Java database connectivity）： Java数据库连接

JDK（Java developer's Kit）： Java开发工具包

JDO（Java database object）： Java数据对象

JPEG（joint photo-graphic experts group）： 联合图像专家组（压缩标准）

JSP（Java server page）： Java服务器页面技术

JVM （Java virtual machine）： Java虚拟机

LAN （local-area network）： 局域网

MAC （media access control）： 介质访问控制

MAN （metropolitan-area network）： 城域网

MDA （model driven architecture）： 模型驱动构架

MFC （Microsoft foundation class）： 微软公司VC++类库名

MIMD （multiple instruction multiple data）： 多指令多数据

MIS （management information system）： 管理信息系统

MOF （managed object format）： 管理的对象格式

MPEG （moving picture experts group）： 运动图像专家组（标准）

MSDN （Microsoft developer network）： 微软开发者网络

MUD （multiple user dimension）： 多人文字角色扮演游戏

MVC （model-view-controller）： 文档-视图-控制模式

NFS （network filing system）： 网络文件系统

OA （office automation）： 办公自动化

OCL （object constraint language）： 对象约束语言

OCR （optical character recognition）： 光学字符识别

ODBC （open database connectivity）： 开放数据库连接

OEM （original equipment manufacture）： 原始设备制造商

OLAP （online analytical processing）： 联机分析处理

OLE （object linking and embedding）： 对象链接和嵌入

OMG （the object management group）： 对象管理组织

OMT （object modeling technique）： 对象建模技术

OO （object oriented）： 面向对象的

OOD （object oriented design）： 面向对象的设计

OOP （object oriented programming）： 面向对象的编程

ORB （object request broker）： 对象请求代理

OSI （open system interconnect reference model）：开放式系统互联参考模型

OWL （object window library）： 对象窗口库

PCI （peripheral component interconnect）：外部设备互联

PHP （PHP hypertext preprocessor）： PHP超文本处理器（语言名，递归定义）

POP3 （post office protocol, Version 3）：电子邮局协议，版本3

PSP （personal software process）： 个体软件过程

QA （quality assurance）： 质量保证

QoS （quality of service）： 服务质量

RAD （rapid application development）： 快速应用程序开发

RAM （random-access memory）： 随机存储器

RAP （Internet route access protocol）： 网际路由存取协议

RARP （reverse address resolution protocol）： 逆向地址解析协议

RDF （resource description framework）：资源描述框架

RIP （routing information protocol）： 路由信息协议

RISC （reduced instruction set computer）： 精简指令集计算机

RMI （remote method invocation）： 远程方法调用

ROM （read-only memory）： 只读存储器

RPC （remote procedure call protocol）： 远过程调用协议

RPG （role play games）： 角色扮演游戏

RUP （Rational unified process）： 瑞理公司软件统一开发过程

SCM （software configuration management）： 软件配置管理

SDK （software development Kit）： 软件开发工具包

SMP （symmetric multi processing）： 对称多处理系统

SMTP （simple mail transfer protocol）： 简单邮件传输协议

SNMP （simple network management protocol）：简单网络管理协议

SOAP （simple object access protocol）： 简单对象访问协议

SQL （structured query language）： 结构化查询语言

STL （standard template library）： 标准模板库

TCP （transmission control protocol）： 传输控制协议

TSP （team software process）： 团队软件过程

UDDI （universal description, discovery and integration）：统一描述、发现和集成协议

UDP （user datagram protocol）： 用户数据报协议

UI （user interface）： 用户界面

UML （the unified modeling language）： 统一建模语言

UP （unified process）： 统一软件开发过程

URL （uniform resource locators）： 通用资源定位符标准

USB （universal serial bus）： 通用串行总线

VAS （value-added serve）： 增值服务

VCD （video compact disc）： 视频光盘

VCL （visual component library）： 可视化构件库

VGA （video graphics adapter）： 视频图形适配器

VLAN （virtual local-area network）： 虚拟局域网

VOD （video on demand）： 视频点播系统

VPN （virtual private network）： 虚拟专用网络

VRML （virtual reality modeling language）： 虚拟现实建模语言

W3C （world wide web consortium）： 万维网联盟

WAN （wide-area network）： 广域网

WAP （wireless application protocol）： 无线应用协议

WCDMA（wideband code division multiple access）：多频码分多址技术

WLAN （wireless local-area network）： 无线局域网

WSDL （web service description language）：Web服务描述语言

WWW （world wide web）： 万维网

XAML （extensible application markup language）：可扩展应用程序标记语言

XML （extensible markup language）： 可扩展标记语言

XP （extreme programming）： 极限编程

上一节 本书简介 下一节

信息与信息化

第14章 信息化基础知识

本章主要介绍信息化的基础知识。

14.1 信息与信息化

本节主要介绍信息与信息化。

上一节 本书简介 下一节

信息的定义及其特性

14.1.1 信息的定义及其特性

信息是一种客观事物，它与材料、能源一样，都是社会的基础资源。但是，理性认识信息却只有几十年的历史。1948年美国科学家香农在对通信理论深入研究的基础上，提出了信息的概念，创立了信息理论。此后，人们对信息的研究迅速增加，形成了一个新的学科--信息论。至今，信息论已发展成为一个内涵非常丰富的学科，并且与控制论和系统论并称现代科学的"三论".计算机技术和网络技术的迅速发展和普及，更加重了"三论"在现代科学技术中的地位。

1.信息的定义

什么是信息？香农在《通信的数学理论》一文中对"信息"的理解是"不确定性的减少",由此引申出信息的一个定义：信息是系统有序程度的度量。同年，控制论的创始人维纳在《控制论》一书中指出，"信息就是信息，不是物质也不是能量".当然，人们还从不同的角度给信息下了定义，据统

计，目前信息的定义不下几十种。但是，被人们所普遍接受的大概还是香农的定义，因为香农不但给出了信息的定义，而且还给出了信息的定量描述，并确定信息量的单位为比特（bit）。1比特的信息量，在变异度为2的最简单情况下，就是能消除非此即彼的不确定性所需要的信息量。香农把热力学中的熵引入信息论。在热力学中，熵是系统无序程度的度量，而信息与熵正好相反，信息是系统有序程度的度量，因而，表现为负熵。它的计算公式如下：

$$H(x) = -\sum P(Xi)\log_2 P(Xi)$$

式中X_i代表n个状态中的第i个状态，$P(X_i)$代表出现第i个状态的概率，H（x）代表用以消除系统不确定性所需的信息量，即以比特为单位的负熵。

乌家培把信息的定义分解为3个层次：

语法或结构形式层次，反映信息的确定度；

语义或逻辑内容层次，反映信息的真实度；

语用或实用价值层次，反映信息的效用度。

对信息的量的研究，与第一个层次有关，构成经典信息论的内容；对信息的质的研究，与第二、第三两个层次有关，构成现代信息论的内容。

2.信息的特征

人们通过深入的研究，发现信息的特征有：

客观性。信息是客观事物在人脑中的反映。而反映的对象则有主观和客观的区别，因而，信息可分为主观信息和客观信息。主观信息，如决策、指令、计划等；客观信息，如国际形势、经济发展等信息。

普遍性。物质的普遍性决定了信息的普遍存在，因而信息无所不在。

无限性。客观世界是无限的，反映客观世界的信息自然也是无限的。

动态性。信息是随着时间的变化而变化的，因而是动态的。

依附性。信息是客观世界的反映，因而要依附于一定的载体而存在，需要有物质的承担者。信息不能完全脱离物质而独立存在。

变换性。信息通过处理可以实现变换或转换，使其形式和内容发生变化，以适应特定的需要。

传递性。信息在时间上的传递就是存储，在空间上的传递就是转移或扩散。

层次性。客观世界是分层次的，反映它的信息也是分层次的。

系统性。信息可以表示为一种集合，不同类别的信息可以形成不同的整体。因而，可以形成与现实世界相对应的信息系统。

转化性。信息的产生不能没有物质，信息的传递不能没有能量，但有效地使用信息可以把信息转化为物质或能量。