外教社

牛津英汉双解百科分类词典系列

Oxford DICTIONARY OF COMPUTING WITH CHINESE TRANSLATION

牛津英汉双解计算机词典

(附汉英术语对照表)

上海外语教育出版社

外教社 SHANGHAI FOREIGN LANGUAGE EDUCATION PRESS

牛津英汉双解百科分类词典系列

Oxford Dictionary of Computing with Chinese Translation

牛津英汉双解计算机词典

Valerie Illingworth 等　原编

张季东　编译

上海外语教育出版社
外教社 SHANGHAI FOREIGN LANGUAGE EDUCATION PRESS

图书在版编目(CIP)数据

牛津英汉双解计算机词典 / 张季东编译. —上海：
上海外语教育出版社，2007
（牛津英语百科分类词典英汉双解版）
ISBN 978-7-5446-0546-5

Ⅰ.牛… Ⅱ.张… Ⅲ.计算机-双解词典-英、汉
Ⅳ.TP3-61

中国版本图书馆 CIP 数据核字(2007)第 118436 号

图字：09-2005-385
出版发行：上海外语教育出版社
（上海外国语大学内） 邮编：200083
电　　话：021-65425300（总机）
电子邮箱：bookinfo@sflep.com.cn
网　　址：http://www.sflep.com.cn　http://www.sflep.com
责任编辑：章骏德

印　　刷：上海信老印刷厂
经　　销：新华书店上海发行所
开　　本：787×960 1/32　印张 36.125　字数 1280 千字
版　　次：2007 年 9 月第 1 版　2007 年 9 月第 1 次印刷
印　　数：3 500 册

书　　号：ISBN978-7-5446-0546-5 / H・0229
定　　价：63.00 元

《牛津英汉双解计算机词典》
编辑出版人员

英语原版编者： Valerie Illingworth 等

英汉双解版编译者： 张季东

策划编辑： 张春明

责任编辑： 章骏德

封面设计： 王中维

版式设计： 章骏德

出 版 说 明

我社自1999年开始陆续出版牛津百科分类词典英语版，迄今已出版近40种。这批百科词典深受广大专业人员、英语学习者的欢迎。同时，部分读者要求我们出版该套词典的英汉双解版，以更好地满足读者学习、翻译的需要。为此，我社经过充分调研和论证，并同牛津大学出版社协商，从该系列中挑选出9种，组织有关专业人员编译成英汉双解版。双解版的9种分别是经济学、商务、金融与银行、计算机、会计、数学、物理学、语言学以及英语语法。

英汉双解版保留了英语原版的全部内容，并提供英语术语的汉语译名。书后附汉英术语对照表，旨在帮助读者集中学习专业词汇和从事汉英翻译。

上海外语教育出版社

2007年6月

Preface 序

The world of computing continues to expand and to cross new frontiers of public awareness. Jargon grows apace, and confusion abounds as the field moves from the domain of specialists into general knowledge. In preparing the Dictionary of Computing, we have recognized the need for clear explanations of the concepts that affect more and more aspects of life and the terminology that accompanies them. The dictionary is aimed mainly at students and teachers of computing but should also be of value to professional and amateur computer users.

The fourth edition of the dictionary contains nearly 6000 entries and a comprehensive cross-reference system. Almost 1700 new entries have been added and many of the existing entries have been extensively updated. This reflects recent advances in all aspects of computing, especially in personal computing, multimedia, and graphics, networking and the Internet, artificial intelligence, and computer security.

The principal areas of interest include:

- computer applications, for example in industry, the office, science, education, and the home;
- the means of achieving these applications in terms of hardware, software, computer organization, telecommunications, and user interaction;
- security, safety, and legal aspects of computing;
- the world of computing – the major computer manufacturers and organizations;
- underlying concepts and theories of computing and where appropriate of electronics, mathematics, and logic.

We would like to express our thanks and appreciation to all those involved in the preparation of the new edition. Over thirty-five practitioners in diverse branches of computing and associated fields produced the new and updated entries. The dictionary has been compiled and prepared for computer typesetting by Market House Books Ltd.

February 1996 *Valerie Illingworth*
Ian Pyle

Guide to the dictionary 使用说明

Alphabetical order in this dictionary ignores spaces, punctuation, and numbers in the entry titles. Greek letters in an entry title are spelt out. Entry titles that consist only of numbers appear at the beginning of the dictionary.

Synonyms and generally used abbreviations are given either in brackets immediately after the relevant entry title, or occasionally in the text of the entry with some additional information or qualification.

An asterisk (*) used before a word or group of words indicates to readers that they will find at the entry so marked further information relevant to the entry that is being read. The asterisk is not used before all the words in an article that are themselves entry titles as this would lead to an unhelpful proliferation of asterisks.

Some entries simply cross-refer the reader to other articles. These may be synonyms or abbreviations or terms more conveniently discussed under the article referred to. In the latter case, the relevant term will appear in the entry in italic type.

A distinction is made between an acronym and an abbreviation: an acronym can be pronounced while an abbreviation cannot. The entry for an acronym usually appears at the acronym itself whereas the entry for an abbreviation usually appears at the unabbreviated form, unless the abbreviation is in common use.

Some terms listed in the dictionary are used both as nouns and verbs. This is usually indicated in the text of an entry if both forms are in common use. In many cases a noun is also used in an adjectival form to qualify another noun. This occurs too often to be noted.

Typography and character set
版式、符号说明

The typefaces and characters used in the dictionary entries follow normal conventions for printing mathematical and technical texts (rather than the more rigorous styles used in some specialist computing texts).

The special characters shown in the table have been used to express specific logic, set theory, and mathematical opera-tions; for further information, see relevant entry. Letters of the Greek alphabet also occur in some entries.

operation, etc.	**symbol**
AND operation, conjunction	$\wedge$.
OR operation, disjunction	$\vee$ +
NOT operation, negation	′ $\neg$ ~
NAND operation	$\mid$ Δ
NOR operation	$\downarrow$ ∇
For set S and/or set T:	
x is a member of S	$x \in S$
x is not a member of S	$x \notin S$
S is a subset of T	$S \subseteq T$
S is a proper subset of T	$S \subset T$
complement of S	$S' \sim S \bar{S}$
union of S and T	$S \cup T$
intersection of S and T	$S \cap T$
Cartesian product of S and T	$S \times T$
relation	R
function of x	$f(x)$, etc.
function f from set X to set Y	$f: X \rightarrow Y$
inverse function	f^{-1}
inverse relation	R^{-1}

operation, etc.	**symbol**
sum, with limits	$\sum_{i=1}^{n}$
integral, with limits	$\int_{b}^{a} \mathrm{d}x$
elements of matrix A	a_{ij}
transpose of matrix A	A^{T}
inverse of matrix A	A^{-1}
equivalence	$\leftrightarrow \equiv$
biconditional	$\leftrightarrow \equiv$
conditional	$\rightarrow \Rightarrow$
general binary operation	[illegible]
universal quantifier	$\forall$
existential quantifier	$\exists$
greater than	$>$
greater than or equal to	$\geqslant$
less than	$<$
less than or equal to	$\leqslant$
approx. equal to	$\approx$
not equal to	$\neq$
infinity	∞

Greek alphabet

alpha	α, A	eta	η, H	nu	ν, N	tau	τ, T
beta	β, B	theta	θ, Θ	xi	ξ, Ξ	upsilon	υ, Υ
gamma	γ, Γ	iota	ι, I	omikron	o, O	phi	ϕ, Φ
delta	δ, Δ	kappa	κ, K	pi	π, Π	chi	χ, X
epsilon	ϵ, E	lambda	λ, Λ	rho	ρ, P	psi	ψ, Ψ
zeta	ζ, Z	mu	μ, M	sigma	σ, Σ	omega	ω, Ω

目　　录

386 *See* Intel.
486 *See* Intel.
80386, 80486 *See* Intel.
68000 *See* Motorola.

A

A* algorithm A* 算法 A member of the class of *best-first *heuristic search techniques that attempt to find a "best" path from a given start node to a designated goal node in a problem *graph. An *evaluation function is used to estimate the cost of the (unknown) distance from the current node being explored to the goal and this is then added to the (known) cost of the shortest path from the start node to the current node to give a figure of merit for the current node. At each iteration the node with the best cost figure is used to pursue the search. The operation of the algorithm displays a behavior that is a mixture of *depth-first and *breadth-first searching.

abduction 回溯法 An *inference process widely used in *artificial intelligence, particularly in *expert systems and *rule-based systems. In diagnosis, for example, there may be a rule like "if measles then red spots" so that, when the symptom red spots occurs, we may use the rule in reverse to conclude that measles is present. However, unlike *deduction, abduction is not logically sound because of inherent uncertainty that can lead to false conclusions - note that measles is not the only cause of red spots. Abduction is an example of a *plausible-reasoning technique.

abelian group (commutative group) 交换群,阿贝尔群 *See* group.

ABI 应用程序二进制接口 *Abbrev. for* application binary interface. Definition of the binary-level interface between application programs and the operating system, including the format of executable files. Compiled binary applications can be ported between systems with the same ABI.

ablative 烧蚀法 An optical recording technique in which the heat generated by the recording beam melts or vaporizes a small area of the recording medium, leaving the underlying layer (with a different reflectivity) exposed.

abnormal termination 异常结束 A termination to a *process brought about by the operating system when the process reaches a point from which it cannot continue, e.g. when the process attempts to obey an undefined instruction. In contrast, a process that reaches a successful conclusion terminates normally by issuing a suitable supervisor call to the operating system. It is common practice to inform the initiator of the process as to whether the termination was normal or abnormal.

abort (of a process) 异常中止 To undergo or cause *abnormal termination. Abortion may be a voluntary act by the process, which realizes that it cannot reach a successful conclusion, or may be brought about by the operating system, which intervenes because the process has failed to observe system constraints. Thus, computationally, the term has a rather similar meaning to its medical meaning of spontaneous or induced fetal death.

absolute address 绝对地址 A unique number that specifies a unique location within the *address space where an operand is to be found/deposited, or where an instruction is located. It generally specifies a memory location but in some cases specifies a machine register or an I/O device. In the case of a binary machine, it is an n-bit number specifying one of 2^n locations. The result of calculating an *effective address is usually an absolute address.

absolute code 绝对代码,目标程序 Program code in a form suitable for direct execution by the central processor, i.e. code containing no symbolic references. *See also* machine code.

absorption laws 吸收规则 The two self-dual laws

$$x \vee (x \wedge y) = x$$
$$x \wedge (x \vee y) = x$$

(*see* duality) that are satisfied by all elements x, y in a *Boolean algebra possessing the two operations $\vee$ and $\wedge$.

abstract computability theory 抽象可计算性理论 The theory of functions that can be computed by algorithms on any *algebra. Its aim is to explore the scope and limits of computation on any kind of data. It is a generalization to arbitrary many sorted algebras of the theory of the effectively calculable or recursive functions on the natural numbers.

Abstract computability theory starts with an analysis and classification of many models of computation and specification that apply to algebras. This reveals the essential features of methods, and results in a *generalized *Church-Turing thesis* (广义丘吉-图灵论题) that establishes which functions on an *abstract data type are programmable by a *deterministic programming language. Comparisons can be made between computations on different algebras, modeling data types and their implementations. The theory also provides a foundation for new theories of computation for special data types, such as algebras of real numbers, which can be used in applications.

The ***while** programming language is a simple example of a method for computing functions on any many-sorted algebra A (that possesses the Booleans). On the natural numbers it can compute all *partial recursive functions. Computation is based on the operations of the algebra – sequencing, branching, and iteration – and has available a limited means of searching A. However, a vital missing component is the capacity to compute with finite sequences of data from A. On the natural numbers finite sequences can be simulated using pairing functions, but it is not possible to simulate finite sequences on an algebra A. Finite sequences and operations for every data set in A are therefore added to A to make a new algebra $A^{\star}$. It turns out that **while** programs on $A^{\star}$ (i.e. **while** programs equipped with finite sequences) have all the essential properties of the computable functions on A. This class of functions is the subject of the generalized Church-Turing thesis.

Most of the main results in the theory of computability on the natural numbers can also be proved for abstract computability theory on any finite generated *minimal algebra.

abstract data type 抽象数据类型 A *data type that is defined solely in terms of the operations that apply to objects of the type without commitment as to how the value of such an object is to be represented (*see* data abstraction).

An abstract data type strictly is a triple (D, F, A) consisting of a set of domains D, a set of functions F each with range and domain in D, and a set of axioms A, which specify the properties of the functions in F. By distinguishing one of the domains d in D, a precise characterization is obtained of the *data structure that the abstract data type imposes on d.

For example, the natural numbers comprise an abstract data type, where the domain d is

$$\{0, 1, 2, \ldots\}$$

and there is an auxiliary domain

$$\{\text{TRUE}, \text{FALSE}\}$$

The functions or operations are ZERO, ISZERO, SUCC, and ADD and the axioms are:

$$\text{ISZERO}(0) = \text{TRUE}$$
$$\text{ISZERO}(\text{SUCC}(x)) = \text{FALSE}$$
$$\text{ADD}(0, y) = y$$
$$\text{ADD}(\text{SUCC}(x), y) =$$
$$\text{SUCC}(\text{ADD}(x, y))$$

These axioms specify precisely the laws that must hold for any implementation of the natural numbers. (Note that a practical implementation could not fulfill the axioms because of word length and overflow.) Such precise characterization is invaluable both to the user and the implementer. Sometimes the concept of function is extended to procedures with multiple results.

The Ada programmer can obtain many of the benefits of abstract data types by defining *packages.

abstract family of languages (AFL) 抽象语言系列 There are many useful types of *formal language, and classes often have similar properties. An AFL is a class of formal languages that is closed under all the following operations: *union, *concatenation, Kleene-plus (*see* Kleene star), *intersection with *regular set, Λ-free homomorphic image, and inverse homomorphic image (*see* homomorphism). An AFL is *full* if it is also closed under Kleene star and homomorphic image. The motivation for the concept of an AFL is to investigate properties of classes of languages that follow merely from the assumption of these *closure properties. Each member of the *Chomsky hierarchy is an AFL; all except for the class of context-free languages are full.

abstraction 抽象 The principle of ignoring those aspects of a subject that are not relevant to the current purpose in order to concentrate solely on those that are. The application of this principle is essential in the development and understanding of all forms of computer system. *See* data abstraction, procedural abstraction.

abstract machine 抽象机 A machine can be thought of as a collection of resources together with a definition of the ways in which these resources can interact. For a real machine these resources actually exist as tangible objects, each of the type expected; for example, addressable storage on a real machine will actually consist of the appropriate number of words of storage, together with suitable address decoders and access mechanisms. It is possible to define an abstract machine, by listing the resources it contains and the interactions between them, without building the machine. Such abstract machines are often of use in attempting to prove the properties of programs, since a suitably defined abstract machine may allow the suppression of unneeded detail. *See* virtual machine.

abstract reduction system 抽象精简系统 (**abstract rewrite** (or **replacement**) **system** 抽象重写系统) A general characterization of the process of deriving or transforming data by means of rules. It is an abstraction based primarily on examples of *term rewriting systems: it is simply a reflexive and transitive binary relation $\rightarrow_R$ on a nonempty set A. For a, $b \in A$, if $a \rightarrow_R b$ then a is said to *reduce* (还原) or *rewrite* (重写) to b.

Using this abstraction, it is easy to define a range of basic notions that play a role in computing with rules.
(1) An element $a \in A$ is a *normal form* (范式) for $\rightarrow_R$ if there does not exist b, different from a, such that $a \rightarrow_R b$.
(2) The reduction system $\rightarrow_R$ is *Church-Rosser* (丘吉-罗瑟) (or *confluent* (汇合的)) if for any $a \in A$ if there are b_1, $b_2 \in A$ such that $a \rightarrow_R b_1$ and $a \rightarrow_R b_2$ then there exists $c \in A$ such that $b_1 \rightarrow_R c$ and $b_2 \rightarrow_R c$.
(3) The reduction system $\rightarrow_R$ is *weakly terminating* (弱终结) (or *weakly normalizing* (弱正规化)) if for each $a \in A$ there is some normal form $b \in A$ so that $a \rightarrow_R b$.
(4) The reduction system $\rightarrow_R$ is *strongly terminating* (强终结) (or *strongly normalizing* (强正规化) or *Noetherian* (诺特的)) if there does not exist an infinite chain

$$a_0 \rightarrow_R a_1 \rightarrow_R \dots \rightarrow_R a_n \rightarrow_R \dots$$

of reductions in A wherein

$$a_i \neq a_{i+1} \text{ for } i = 0, 1, 2, \dots$$

(5) The reduction system $\rightarrow_R$ is *complete* (完整的) if it is Church-Rosser and strongly terminating.
(6) A reduction system is Church-Rosser and weakly terminating if, and only if, every element reduces to a *unique* (唯一) normal form. Let $\equiv_R$ denote the smallest equivalence

relation on A containing $\twoheadrightarrow_R$. If $\twoheadrightarrow_R$ is a Church-Rosser weakly terminating reduction system then .he set $NF(\twoheadrightarrow_R)$ of normal forms is a transversal for $\equiv_R$, i.e. a set that contains one and only one representative of each equivalence class.

abstract specification 摘要说明 A specification for software expressed in a (mathematically) *formal language such that the specification is completely independent of, and does not imply, any design and implementation method and languages. It does not normally express the constraints that the final software must satisfy. *See also* formal specification.

A-buffer A-缓冲器 A buffer used with a *Z-buffer to hold information concerning the visible transparent surfaces to be considered at each *pixel of an image. The A-buffer originated in an *anti-aliased *hidden-surface removal algorithm developed by Loren Carpenter around 1984. It resolves visibility among an arbitrary collection of opaque, transparent, and intersecting objects. The algorithm was developed for the REYES system at Lucasfilm Ltd. Road to Point Reyes was a famous image produced by the system.

acceleration time 加速时间 (**start time** 启动时间) The time taken for a device to reach its operating speed from a quiescent state.

accept (**recognize**) 接收 a formal language. *See* automaton, finite-state automaton.

acceptable use policy (**AUP**) 许可使用策略 The set of rules governing the use that can be made of a network. All network users are expected to conform to any existing legislation, and to any commercial conditions that form part of any contract for the use of commercial networks. In the case of academic or research networks there are also likely to be constraints on using the network to carry commercial traffic, and these will be embodied in the AUP.

acceptance testing 检测 *See* testing. *See also* review.

access 1. 读写(数据) The reading or writing of data, with the connotation that the content of the reading or writing is taken into account. The word is most commonly used in connection with filed information and is often qualified by an indication as to the types of access that are to be permitted. For example, read-only access means that the contents of the file may be read but not altered or erased.

2. 存取权 The right or opportunity to read or write data or programs. The UK *Computer Misuse Act 1990 states that "a person secures access to any program or data held in a computer if by causing a computer to perform any function he alters or erases the program or data, copies or moves it to any storage medium other than that in which it is held or to a different location in the storage medium in which it is held, uses it or has it output from the computer in which it is held (whether by having it displayed or in any other manner)".
3. 访问 To gain entry to data, a computer system, etc. In the US, to access strictly means to instruct, communicate with, store data in, retrieve data from, or otherwise obtain the ability to use the resources of a computer or any part thereof.

Access Access 数据库(微软公司生产的) *Trademark* A relational database management system for personal computers from Microsoft.

access control 访问控制 A *trusted process that limits access to the resources and objects of a computer system in accordance with a *security model. The process can be implemented by reference to a stored table that lists the *access rights of subjects to objects, e.g. users to records. Optionally the process may record in an *audit trail any illegal access attempts.

access method 存取法 Any algorithm used for the storage and retrieval of records from a *data file or *database. Access methods are of two kinds: those that determine the structural characteristics of the file on which it is used (its *file organization) and those that do not (as in secondary indexing (*see* indexed file) and *data chaining). In the first case essentially the same algorithm is used for the retrieval of a record as was used for its original physical placement, whereas in the second these algorithms are quite distinct. Hence in the first case the same term may be used interchangeably (and loosely) for both the access method and the file organization (*see* random access (def. 2), sequential access).

access path 存取路径 The name given to the set of names of devices, *directories, *subdirectories, and a specific *file, by means of which the file-management system is able to reach the specified file. Depending on the details of the file-management system actually in use, the access path may start with the name of a physical or logical device, which holds a number of directories that associate the identity of an object with its location on the device; these objects may in turn be further directories (usually then known as subdirectories) or they may be files containing end-user data. The complete set of intermediate objects, in the order in which they are used, is the access path.

access rights (access privileges) 存取权 A classification of the modes of access to an object granted to particular subjects, or groups of users. Thus, the owner of a file will typically have rights to read, write, or delete the file. Some or all these rights may also be granted to other users on the system. *See* access control.

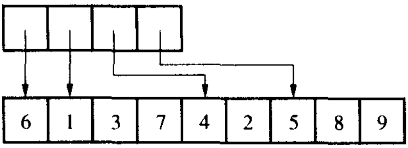
access time 存取时间, 访问时间 The time taken to retrieve an item of information from storage. The access time may be counted in nanoseconds for a semiconductor device, in milliseconds for a magnetic disk, or in minutes if the file containing the required data is on magnetic tape.

In the case of disk storage, the access time is the average time taken for a disk drive to provide the first byte of data, measured from the time the host issues a read command. To a good approximation, the average access time is the sum of the average *seek time, the command overhead, and the average *latency. *See also* memory hierarchy.

access vector 存取向量 A vector that is used in the representation of a *ragged array. For example, the elements of a row-ragged array, *A*, would be stored row by row in a vector *B*. The *i*th element of the access vector would then point to the position in *B* where the first element of the *i*th row of *A* was stored (*see* diagram). A column-ragged array

6		
1	3	7
4	2	
5	8	9

row-ragged array



representation using an access vector

Access vector