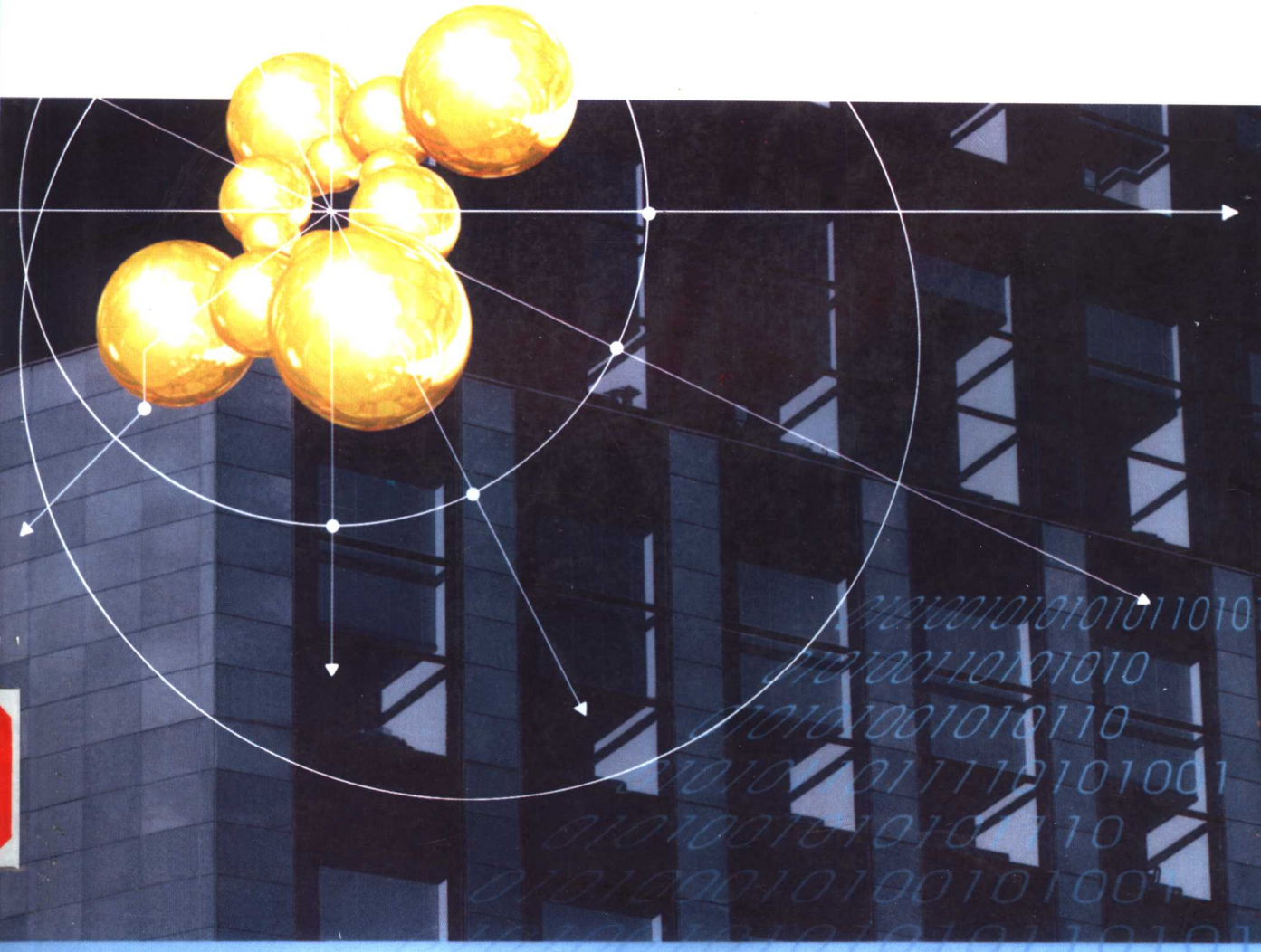
Linux C/C++

入门与进阶

位元文化　编著



科学出版社
www.sciencep.com

Linux C/C++入门与进阶

位元文化　编著

科学出版社

北京

图字：01-2004-3698

内 容 简 介

Linux 是现今流行的操作系统，它具有强大的功能和高度的稳定性；而 C/C++语言是目前应用最广泛，且功能最强大的程序开发语言之一，本书将详尽地介绍在 Linux 环境下运用 C/C++开发程序。

全书共分为 29 章和一个附录，主要介绍了变量、数据类型、运算符与表达式、流程控制、数组、指针、函数、对象导向、类别与对象、类别的继承、信息、模板、例外处理、标准模板链接库、模块化开发、自定义标头文件、条件式编译、批次编译、版本管理、窗口的事件处理、对话框与控件等方面的内容。

本书内容丰富，讲解详尽，可作为计算机及相关专业的研究生、本科生、大专生的参考书，同时也可作为相关专业从业人员的实用参考书。

图书在版编目（CIP）数据

Linux C C++入门与进阶/位元文化编著 一北京：科学出版社，2004
ISBN 7-03-013768-X

Ⅰ.L… Ⅱ.位… Ⅲ.C 语言-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2004）第 062865 号

责任编辑：吕建忠 丁 波/责任校对：都 岚
责任印制：吕春珉/封面设计：北新华文

科学出版社 出版
北京东黄城根北街16号
邮政编码:100717
http://www.sciencep.com

新蕾印刷厂 印刷

科学出版社发行 各地新华书店经销
*
2004 年 10 月第 一 版 开本：787×1092 1/16
2004 年 10 月第一次印刷 印张：43
印数：1—4 000 字数：1000 000

定价：66.00 元

（如有印装质量问题，我社负责调换〈路通〉）

前　言

免费的 Linux 平台是现在流行的操作系统，它的高度稳定性与强大服务器功能，都让人无法忽视它的存在。而 C/C++语言是目前应用最广泛，且功能最强大的程序开发语言之一。在 Linux 环境中运用 C/C++开发程序，可同时兼有免费且稳定的系统与功能强大的语言辅助的功能。

本书将以最详细的说明，让读者熟悉 Linux 下的程序设计环境，首先介绍编写程序的文字编辑程序（vi/vim、Kate、Gedit、Kwrite）与编译程序的 g++编译器，再介绍 C/C++的基础观念与语法，之后慢慢地深入对象导向、模板、STL、模块这些进阶的主题，并且介绍 Linux 环境下最常用的批次编译（make 指令与 Makefile 文件）与版本管理（CVS），最后介绍 Linux 环境下的窗口程序开发（使用 wxWindows 工具集），让你完整体验 Linux 环境的 C/C++程序开发过程。

书中程序的源代码放在 www.abook.cn 上，请感兴趣的读者下载阅读。

本书的完成要感谢许多人。在此特别感谢位元文化在写作上所提供的指导，以及排版校对的技术协助。

对于每一本书，虽然我们都尽了最大的努力，但是个人的学识、能力毕竟有限，如果你发现书中有不妥之处，欢迎指正。当然，除了指正错误外，如果你对于书中的内容有所建议，同样也欢迎与我们讨论。

杨青长　郭尚君
chingzz@mail.apol.com.tw
2004 年

目　录

第1章

简介 C/C++程序语言

本章重点

- 了解程序语言。
- 简介 C/C++。
- Linux 下的程序开发环境。

本章导读

在本章中，我们将简单介绍程序语言的基本概念、C/C++的发展过程，以及在 Linux 操作系统下，开发程序的环境与基本操作。

1.1 了解程序语言

下面来了解一下程序语言。

1. 程序语言的简单定义

什么叫程序？程序的作用是什么呢？为什么要学习程序？要怎么样才能学好程序呢？我想初次接触到程序的读者都难免有这些疑问。自从计算机发明后，人类的生活因为计算机而产生了重大的变化。但是计算机只是一台机器，人们必须通过指令（Instruction）来指使计算机执行我们想要它做的动作。而依照顺序执行的一组指令，便被称为程序语言（Program Language）。

为了要使计算机能够看得懂，所以，编写程序时，必须遵守一定的格式，并使用特定的语法。同样地，计算机在读取这些程序时，也将遵循一定的规则执行，例如，人与人之间沟通时所用的语言，不管是中文还是英文，每一种语言都有它一定的语法，这就像程序必须遵守特定的格式一样，下面总结一下程序所具有的特征。

- 程序是与计算机沟通的语言。
- 程序是由特定语法与关键词构成的一行一行的语句。
- 程序是一行一行地执行的。
- 程序的执行，是从进入点开始，原则上是由上而下、从左至右来执行。

2. 程序语言的演进

若依照语言的特性与发展的时间顺序，程序的发展过程可分为以下 5 个阶段。

- 机器语言。
- 汇编语言。
- 高级语言。
- 第四代语言。
- 自然语言。

分别叙述如下。

（1）机器语言

最早发展出的语言为机器语言（Machine Language），是计算机硬件唯一能看得懂的语言，是运用二进制的 0 与 1 来组合出指令的，故又称为机器码。所有程序语言最后都必须转化为机器语言，才能在计算机上执行。以一连串 0 与 1 组合而成的机器语言，一般人很难记忆其所代表的意义，因为 0 与 1 的组合可能代表数据，也可能代表指令，非常难以解读。但由于机器语言可被计算机硬件直接解读，所以执行速度最快。

由于不同类型的计算机可接受的指令并不相同，所以，运用机器语言开发程序时，仅能针对该类型计算机设计。完成设计的程序，无法在其他类型的计算机上执行，所以不具有可移植性（Portability）。

（2）汇编语言

汇编语言（Assembly Language）相当接近于机器语言，它是以人类比较容易记忆的辅助记忆码（Mnemonics）代替机器语言由 0 与 1 组合成的指令，故又称为低级语言（Low-level Language）。用汇编语言编写出的程序，必须经过汇编程序（Assembler），才能转换为机器语言，并在计算机上执行。与机器语言相同，不同类型的计算机所接受的汇编语言亦不相同，所以，开发出的程序也不具有可移植性。

（3）高级语言

因为汇编语言太接近机器语言，并不适合一般用户使用，且受限于计算机硬件，无法很容易地移植到不同类型的计算机中。而汇编语言的每个指令所能执行的动作相当有限，并不适合于开发稍具规模的程序。为了克服上述问题，发展出了高级语言（High-level Language）。

高级语言的语法较为接近人类的自然语言（Natural Language），且执行动作的单位并不是指令，而是语句，一句语句将对应数个指令。此外，高级语言对于数据处理与算法应用，亦提供较佳的支持，适合于开发大型程序。

运用高级语言开发出的程序，必须运用翻译程序，将程序翻译为机器码，才能够在计算机上执行。依据翻译方式的不同，可分为编译式语言与直译式语言。编译式语言使用的翻译程序称为编译器（Compiler），直译式语言使用的翻译程序则称为直译器（Interpreter）。

以高级语言开发的程序较不受限于计算机硬件，只要稍微修改重新编译后便可在不同类型的计算机硬件上执行，具备较佳的可移植性。目前常见的高级语言有 C、C++、Visual Basic、Java、Pascal、Python 等。

（4）第四代语言（超高级语言）

第四代语言简称为 4GL（Fourth Generation Language），又有人称为超高级语言（Very High-level Language）或非程序性语言（Nonprocedural Language），为问题导向的程序

语言（Problem Oriented Language）。不同于上述 3 种类型的程序语言，第四代语言的运用仅需告诉计算机要“做什么”，而不是要教计算机“如何做”，常见的有 SQL（Structural Query Language，结构化查询语言）、QBE（Query By Example，实例查询）。

超高级语言通常由数个整合性系统开发软件组成，并提供许多模块，程序设计师只要在系统开发软件内运用选取方式即可快速开发程序，而不需要花太多心思编写程序，这可大大提升程序开发的效率。

（5）自然语言（第五代程序语言）

自然语言（Nature Language）是运用人工智能，提供人类以接近口语的指令操作计算机，又称为知识库语言（Knowledge Based Language）。自然语言大约从 1988 开始发展至今，技术尚未完全成熟，仍无法处理较复杂的逻辑问题。

3. 编译语言与直译语言

高级语言以执行方式区分时，对于不能直接执行，需要先经过编译与连接程序产生可执行文件的语言，称为编译语言，如 C、Cobol、Fortran、Pascal、C++与 Java 等，都是编译语言。

我们将那些不需要经过编译与连接的过程，直接在特定的程序或环境下即可执行的语言称为直译语言，例如，网页所使用的 JavaScript、早期的 Basic（Basic 后来也有编译式的版本）以及具有对象导向能力的 Python 语言。以下将详细介绍编译语言与直译语言。

（1）编译语言

程序的原始代码文件需要经过编译与连接的过程，才能产生可在计算机系统中执行的可执行文件。可执行文件的内容为经过编译与连接后，可被计算机接受的机器指令码，机器码并不是人所能看得懂的，机器码亦无法还原成程序的原始码。关于 C/C++的编译过程，请参考 1.2 节的介绍。

（2）直译语言

直译语言并不需要经过编译与连接的程序，而是直接以特定程序（也就是直译器）一行一行地读取程序，并翻译成为机器执行指令直接执行。若程序内有错误则必须要执行到那一行时，直译器才会告诉你有错误，直译器执行程序的过程如图 1.1 所示。

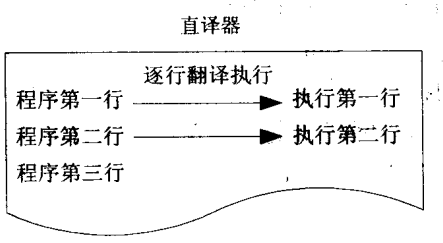


图 1.1　直译器执行程序的过程

直译语言采用逐行翻译的方式执行，并不需要写好一个完整的程序才可以执行，可以先执行已经写好的部分程序，看看结果是不是我们所要的，然后再接下去写未完成的部分。这点在编写程序时相当方便，但是也有缺点，那就是程序在执行前才逐行翻译，

将增加执行的时间，导致程序执行效能较差，因此，直译语言较适合用于发展小型或效率不为优先考虑的程序。

编译语言因为已先经过编译与连接，将程序码转换为机器码的可执行文件。因此在执行时不需要和直译语言一样再经过翻译步骤，所以程序的执行时间较短，效率也较高。所以，现在大部分的程序语言都采取编译式的做法，以求得较好的执行效率。

1.2 简介 C/C++

下面来了解一下 C/C++。

1. C/C++语言与 g++编译器的发展过程

（1）C 语言

C 语言是由 BCPL 与 B 这两种语言延续发展而来的。原来所有的程序都是以低级语言编写的，在 1967 年，Martin Richards 为了编写操作系统与编译器，开发出 BCPL 语言；1970 年，贝尔实验室的 Ken Thompson 为了同样的目的，以 BCPL 为基础发展出 B 语言；1972 年，同为贝尔实验室的 Dennis Ritchie 再以 B 语言为基础开发出 C 语言，用于编写 DEC PDP-11 计算机的系统程序，此可谓以 C 语言发展 UNIX 操作系统的开始，之后，UNIX 操作系统就大部分是以 C 语言发展了，直到后期才出现以 C++语言来发展。

（2）C++语言

20 世纪 80 年代初，贝尔实验室的 Bjarne Stroustrup 博士将对象导向的观念加入 C 发展出 C++语言（1985 年），其中还加入了配合对象观念的特性，例如，类别与类别成员函数的定义、对象声明、运算符重载等。使程序设计走向对象化，变得更容易修改与重复使用。此时也有许多有名的 C++版本，例如，Turbo C++、Microsoft C++、Visual C++、Borland C++与 Borland C++ Builder、g++等。

（3）Java 语言

到了 20 世纪 90 年代，Sun 公司为了让相同的程序可以在不同控制芯片的电子消费性电子产品上执行，便以 C++为蓝本去除容易导致系统产生错误的机制，例如，指针、运算符重载、资源参考等，避免因为程序错误导致产品损毁，从而开发出 Oak 语言（Java 的前身），但采用的公司并不如预想的踊跃。

1993 年网络风潮席卷全球，Sun 公司为因应此风潮，将 Oak 更名为 Java，并于 1995 年正式发表。Java 语言的特点为简单、安全、具有完全的可移植性、支持对象导向技术以及可靠性高。Java 乘着这股网络风潮顺势发展，在目前与可预见的未来，都将是被广泛应用，并占有重要地位的程序语言。

（4）g++编译器

Linux 环境中最常用来编译 C++程序的编译器为 g++（GNU C++），它是由自由软件基金会（Free Software Foundation，FSF）所推动的 GNU 计划所开发的。此计划中几乎提供了常见各种语言的编译器，而其中的 GCC（GNU Compiler Collection）套件中，则包含了表 1.1 所列的多种支持 ANSI C 标准的语言。

其实，GCC 编译器可用来编译下列所有语言的程序，但习惯上，我们仍会以各语

言所对应的编译器来编译程序，以免除复杂程序编译时所需指定的详细连接设定。目前 GCC 的版本为 3，对于版本与更新的数据请查询 GCC 的首页 http://gcc.gnu.org。

表 1.1　支持 ANSI C 标准的语言

	语言种类	编译器名称
GCC 套件	C 语言	GCC
	C++语言	g++
	Fortran 语言	G77
	Java	GCJ
	Ada	GNAT（GNU Ada 95 Compiler）
	Objective C	GCC

GNU 计划与自由软件基金会是由 Richard M. Stallman 所推动。GNU 计划开始于 1984 年，旨在发展一个“类-UNIX”且为“自由软件”的完整操作系统——GNU 系统。自由软件基金会成立于 1985 年，其目的在于促进自由软件的使用、研究、复制、修改和重新散布，也是推动 GNU 计划的主要单位。

GNU 系统后来与核心为 Linux 的程序相结合，成为一个完整的操作系统 GNU/Linux，即现在通称的 Linux 操作系统。通常现在所使用的 Linux 操作系统是不同厂商所整合的发行版（Distribution），其内容、标志、安装方式皆有所不同，表 1.2 列出了一些常见版本与其网址。

表 1.2　常见的 Linux 版本名称及其网址

发行的 Linux 版本名称	网　址
Red Hat Linux	http://www.redhat.com
Mandrake Linux	http://www.mandrakelinux.com
SuSE Linux	http://www.suse.com
Debian GNU/Linux	http://www.debian.org
Turbo Linux	http://www.turbolinux.com
Lycoris Linux	http://www.lycoris.com
Lindows Linux	http://www.lindows.com
红旗 Linux	http://www.redflag-linux.com

2. C/C++语言的特性

C/C++语言具备以下 10 个特点。

- 具有精简的架构，但不影响它发展复杂程序的能力。
- 自由的语言格式。
- 高效率的编译式语言。
- 功能强大，同时具有低级语言与高级语言的能力的中等语言。
- 具有跨平台与操作系统的可移植性。
- 灵活的流程控制与结构化的程序。
- 可发展高度模块化的程序。
- C++支持对象导向观念。

- 可作为学习 Java 程序的基础。
- 为业界广泛使用。

3. C/C++程序的编译过程

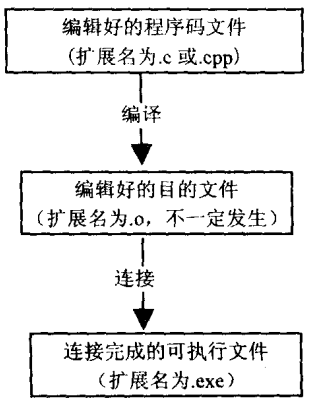


图 1.2 C/C++程序的编译与连接过

C/C++程序的原始代码文件（.c/.cpp），需先经过编译器的编译与连接并产生可执行文件方可执行，编译与连接的过程如图 1.2 所示。

以下将详细说明编译与连接的过程。

（1）编译

写好的程序码（.c/cpp）在编译的过程前，会先进行前置处理，处理以“#”开头的指令。完成前置处理后，将进行编译的操作，把文字的语法指令转换成为机器码。在进行编译的动作时，编译器会检查程序中的语法是否正确（前置处理一遍，再编译一遍）。若程序中有不正确的语法，编译器会告诉用户那里的程序写错了，以及相关的错误信息，有关错误的处理，请参考第 2 章。

若程序都没有语法上的错误时，编译器便会将程序原始代码文件（扩展名为.c 或.cpp）编译后，进入下一个连接步骤。在 g++中只有使用“-c”参数，通知编译器只进行编译不连接，才会产生出目的文件来。

Linux 环境下扩展名所代表的文件种类如表 1.3 所示，其中，C++的程序码文件除了以.cpp 来指定外，尚有许多不同扩展名的命名方式。

（2）连接

经过编译后的目的文件还必须经过连接器（Linker）的连接步骤。在连接过程中，除了原来程序的目的文件外，还将自动连接程序中所使用的，其他的目的文件或程序库的机器码再产生可执行文件，这一个文件才是可以在计算机上执行的程序。

表 1.3 扩展名所代表的文件种类

文件种类	扩展名
C++程序代码文件	.c
	.cpp
	.cp
	.cxx
	.c++
	.cc
	.C
标头文件	.h
可执行文件	以文件属性“执行”来指定
目的文件	.o

在 Linux 环境下，文件可否执行是以文件属性的方式来指定的，这与在窗口操作系统中，以扩展名.exe 来判别是不同的。用 g++编译程序时，若未以参数“-o”指定输出的可执行文件名称，则预设为 a.out。

以 g++编译程序时，若不特别指定参数，则编译与连接的过程将一次进行，不产生中间的目的文件，直接建立出可执行文件来。

1.3　Linux 下的程序开发环境

本节将介绍 Linux 环境下，开发 C++程序的环境，将说明下列主题。

- 首先以 Red Hat Linux 9 与 Mandrake Linux 9.1 发行版为例，说明 GCC 套件的安装。
- 介绍用以编辑程序代码文件的文字编辑程序。
- 说明 Linux 环境下的查询指令。

除了以 GCC 套件来编译程序之外，Linux 中也有一些具有整合开发环境（Integrated Developing Environment，IDE）的编译程序可使用，这类编译器整合了编辑程序、编译器与连接器，在使用上更为方便，但不如 GCC 已经都附在各个发行版中那么普遍与通用，表 1.4 中列出几种常见的编译程序。

表 1.4　几种常见的编译程序

名　称	说　明
RHIDE	为一个类似 Turbo C++的编译程序
xwpe-alpha	有文字与图形模式两种版本，必须先安装文字版本后再安装图形版本。原始版本为 xwpe，但原作者已停止更新与参与，现在以计划的类型继续更新
Kdevelop	Linux 环境中功能完整的 IDE，大部分的套件中都包含，可编写多种语言程序，且支持 Qt 窗口程序的写作，版本 3 为目前最新版
Kylix	由 Borland 公司所发展，可在 Linux 环境下开发 C/C++程序，为一个功能完整的 IDE，亦可开发窗口程序，有免费版与付费版两种，版本 3 为目前最新版

2. 文字编辑程序

以 g++编译程序前，需要先用文字编辑程序来编写与编辑程序代码文件。以下将介绍可在 Linux 中进行文字编辑的程序，其中 vi/vim 是最常见的文字式编辑程序，而图形窗口化程序 Kate 则同时具有编辑与命令窗口，可分别进行编辑与指令执行，用来编写程序最方便。下面将介绍的编辑程序，除了 Gedit 之外，其他的在编写程序码时，关键词皆会自动以不同的颜色区别显示。

（1）vi/vim

vi 与 vim 为 Linux 上的命令式文字编辑器，vim 为 vi 的加强版，现不管是执行 vi 或 vim 都将为 vim 版本。在终端机的命令行中执行 vim 程序的程序如下所示。

```
$ vim
```

执行 vim 指令时，可直接在其后指定要打开或新建的文件名称，如下的指令将以 vi 编辑 my.cpp 文件。

```
$ vim my.cpp
```

vim 有 3 种操作模式，其关系与切换按键图标如图 1.3 所示，vi 执行时，预设为指令模式。

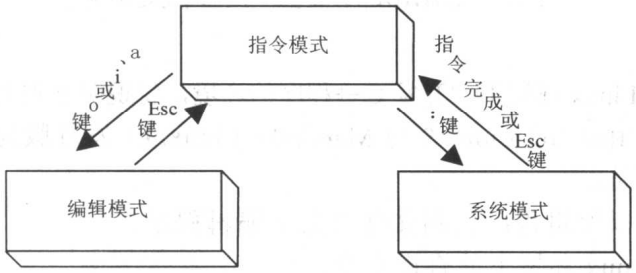


图 1.3　3 种操作模式的关系与按键图标

以下将一一说明这 3 种模式的作用，以及可使用的功能与按键。

- 指令模式。指令模式为打开文件后的预设模式，在此模式中，可随意移动光标的位置，并可针对光标所在的位置，进行复制、删除文字以及复原的操作，但无法直接编辑文字的内容。利用表 1.5 所列的按键可移动光标的位置，其中“^”表示同时按下 Ctrl 键，例如，“^b”为同时按下 Ctrl 与 b 键。

表 1.5　按键及其说明

按　键	说　明	按　键	说　明
h 或←	光标向左移动一个位置	^b 或 Page Up	屏幕向上卷动一个页面
l 或→	光标向右移动一个位置	^f 或 Page Down	屏幕向下卷动一个页面
j 或↑	光标向上移动一个位置	^e	屏幕向上卷动一行
k 或↓	光标向下移动一个位置	^y	屏幕向下卷动一行
^g 或^G	显示光标位置、文件总行数、文件位置与名称等信息	数字 g	移动光标到数字指定行的开头位置，例如，3g，将移动光标到第 3 行开头位置
n	跳动光标到下一个搜索标记的位置	N	跳动光标到上一个搜索标记的位置

表 1.6 为复制功能的按键与说明。

表 1.6　复制功能的按键与说明

按　键	说　明
yy	将光标所在列复制到缓冲区之中
Yw	将光标所在的字，复制到缓冲区
数字 yw 或 y 数字 w	由光标所在位置计算，将指定个数的文字复制到缓冲区，如 3yw 或 y3w 将复制 3 个字