Linux/UNIX 高级编程

中科红旗软件技术有限公司　编著

清 华 大 学 出 版 社

（京）新登字 158 号

内 容 简 介

本书全面系统地介绍了 Linux/UNIX 高级编程的方法。其中第一章介绍了 Linux 的软件开发工具；第二章至第五章讲述了 Linux 的文件系统，I/O 系统调用、系统数据文件等；第六至第九章讲述了 Linux 进程模型等方面的内容；第十章讲述终端 I/O 编程；第十一章讲述网络套接字编程；第十二章讲述多线程、系统服务等方面的内容。第十三章讲述多线程编程等内容。

本书结合大量实例，概念清晰，特色鲜明，实用性强。

本书适合于 Linux/UNIX 高级编程人员。

书　　名：Linux/UNIX 高级编程
作　　者：中科红旗软件技术有限公司
出 版 者：清华大学出版社（北京清华大学学研大厦，邮编 100084）
http://www.tup.tsinghua.edu.cn
责任编辑：柴文强
印 刷 者：清华大学印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×960 1/16　**印张**: 31.5　**字数**: 703 千字
版　　次：2001 年 7 月第 1 版　2001 年 7 月第 1 次印刷
书　　号：ISBN 7- 302-04605-0/TP・2728
印　　数：0001~5000
定　　价：52.00 元

前　言

Linux 是运行在个人计算机和工作站上的 UNIX 操作系统，它继承了 UNIX 的全部优点，是真正的多用户、多任务、多平台的操作系统。在个人计算机和工作站上使用 Linux，能充分发挥硬件的功能，使个人计算机能够作为工作站和服务器使用，提高工作站效率；Linux 符合 POSIX 标准，能同时支持 i386、Alpha、SPARC 和 PowerPC 等处理器；对于在 UNIX 上运行的软件，不用改动或稍加改动就可以运行在 Linux 操作系统上。

同 UNIX 相比，Linux 要灵活得多。Linux 的用户界面非常友好，用户使用很方便。Linux 采用市集开发方式，受自由软件委员会指定的 GNU（General Public License）公用许可证保护，Linux 系统及其应用软件的源代码都是向用户公开的，允许用户使用 Linux 自带的 C/C++ 语言编译器修改并编译自己喜欢的程序，同时也可以在 GNU 的许可下使用这些源程序的部分或全部代码 这种利用 Linux 源程序的可能性大大激发了世界范围内热衷于计算机事业用户的兴趣和创造力，吸引了众多程序员为 Linux 工作；目前，所有 UNIX 的主要功能都已经有相应的 Linux 工具和程序，大量的自由软件被移植到 Linux 操作系统上，世界各地流行着越来越多的 Linux 发行版本，如 Red Hat Linux，Red Flag Linux 和 Turbo Linux 等；在这些发行版本中，不仅包含 Linux 操作系统的内核，还包含各种编程语言的编译程序、数据库管理系统、图形用户界面、通信联网工具以及大量的其他应用软件。可见，Linux 已经是一个相当优秀的应用软件开发平台。

同 Windows 相比，Linux 也一样具有优势。Linux 强大的网络功能使 Internet 以及许许多多局域网中的服务器都被 Linux 占据，这不仅是由于 Linux 的稳定性相当出色，也是由于 Linux 的系统效率极高；与同类型的 Windows NT 服务器相比，Linux 对硬件的要求可以降低一到两个档次，一台普通的 PC 机就能胜任复杂的网络服务工作；诸如基于文本的 BBS 这样的电子公告牌的讨论性网络服务是 Windows NT 无法实现的，因为迄今为止，只有基于 UNIX 系统的使用文本方式的 BBS 系统；另外，Linux 下的 TCP/IP 网络方面的应用软件更是应有尽有，能实现许多在普通系统中无法实现的网络功能。

不可否认，Linux 具有较 UNIX 和 Windows 更卓越的性能；但是，价格却低得多。因此，如果您想提高自己的计算机使用和编程水平，学习 Linux 确实是一条捷径。

为促进 Linux 在我国的推广应用，使您能更快地步入 Linux 的奇妙世界，少走一些弯路。针对目前市场上缺乏全面、翔实的 Linux / Unix 编程书籍，红旗 Linux 教育培训部拟组织经验丰富的 Linux 编程人员，参考国际经典 Linux、Unix 编程书籍，编写了这本《Linux / Unix 高级编程》。选择在自由软件日趋成熟之际发行，旨在充分发挥我国软件工作人员的积极性和创造力，振兴我国的软件事业。本书内容涵盖了 Linux / Unix 编程的主要方面，具有一定

的深度，具体内容如下：

- 第一章，为您详细介绍 Linux 的软件开发工具，包括编译、调试、维护工具以及广泛应用的集成开发环境。
- 第二、三、四、五章，讲述 Linux 的文件系统、I/O 系统调用、系统数据文件方面的内容。本部分不仅详细讲述了传统 Unix 的文件、I/O 系统调用，还专门介绍了 Linux 文件、I/O 系统的特点。
- 第六、七、八、九章，讲述 Linux 进程模型，包括进程关系、进程控制、进程间通信、信号处理等方面的内容。
- 第十章，讲述终端 I/O 编程，它的用途主要包括：终端、计算机之间的连接、调制解调器、打印机等方面，因此非常复杂。
- 第十一章，讲述网络套接字编程，socket 是最广泛使用和最成熟的网络编程接口规范，用来实现网络的互联。
- 第十二章，守护进程、系统服务等方面的内容，可以编写自己的守护进程来增加、增强系统服务。
- 第十三章，讲述多线程编程，在操作系统层面上讲述了线程的层次和模型，还包括线程同步、互斥等方面的内容。

本书结合大量实例，概念清晰，内容丰富翔实，特色鲜明，实用性强。本书不仅仅是一本手册，很多内容都在操作系统、核心中得到相互印证，真诚期待您的会心一笑。

本书由中科红旗软件技术有限公司编著，耿增强、宋立秋、张荣华、滕靖、朱昊、张焕强、时培植、黄小婉参与了写作工作，在此表示深深的感谢。由于本书涉及的内容丰富，加之篇幅、时间有限，书中难免有纰漏和不足，失误之处请务必谅解和指正。

中科红旗软件技术有限公司

2001 年 4 月

目　　录

第一章　Linux 软件开发工具

1.1　gcc 和 g++

gcc 和 g++ 分别是 gnu 的 C 及 C++ 编译器，是 Linux 系统中将 C 及 C++ 语言源文件生成可执行程序的工具。

一个可执行的 C 或 C++ 程序需要经过如下四步生成：

第一步：由预处理程序对 C 或 C++ 语言源文件（*.c 或 *.cpp、*.C、*.cxx 等）进行宏扩展和条件处理，并导入前导文件，生成以 .i 为后缀的文件；

第二步：由编译程序将预处理后的文件（*.i）中的 C 或 C++ 语言源代码转换成汇编语言代码，生成以 .s 为后缀的汇编文件；

第三步：由汇编程序将汇编文件（*.s）中的汇编语言代码转换成目标代码（机器代码，*.o），生成以 .o 为后缀的目标文件；

第四步：由连接程序将目标文件（*.o）同指定的库文件进行连接，生成可执行程序。

若非特别指定，gcc 或 g++ 将自动调用上述工具，生成可执行程序。

gcc 和 g++命令的调用格式为：

```
gcc [options] filenames
g++ [options] filenames
```

其中：options 为 gcc 和 g++ 命令的选项，可以为空；多个选项必须分开写，如“ -dr ”不同于“ -d -r ”。filenames 是 gcc 和 g++ 命令的输入文件列表，多个文件之间以空格分隔。

gcc 和 g++ 命令的选项很多，它们规定着 gcc 和 g++ 命令的行为；掌握 gcc 和 g++ 命令的关键就是正确使用它们的选项。

gcc 和 g++ 命令的选项大致可分为如下几类：

- 全局选项，影响编译的全过程（从预处理到连接）；
- 语言选项；
- 预处理程序选项；
- 汇编程序选项；
- 连接程序选项；
- 目录选项；
- 优化选项；
- 调试选项；

- 警告选项。

下面分别介绍各类的常用选项（除非特殊声明，选项将同时适用于 gcc 和 g++ 命令）。

1.1.1 全局选项

1．-x language filenames

此选项指明 filenames 文件的编程语言（C 语言、C++语言等）及 gcc / g++ 编译的起始阶段（预处理、编译、汇编、连接）。

language 的可选值为 C、C++、assembler-with-cpp 等；其中：C、C++ 指明输入文件为 C 语言、C++ 语言文件，编译过程从预处理阶段开始；assembler、assembler-with-cpp 指明编译过程从汇编阶段开始。

-x language 选项对其后的多个文件都有效，直至遇到下一个-x 为止。

在不使用此选项时，gcc/g++ 利用文件名后缀来确定语言及编译的起始阶段。如：输入文件名以 .c 或 .cpp 为后缀时，从预处理阶段开始编译；以 .i 为后缀时，从编译阶段开始编译；以 .s 为后缀时，从汇编阶段开始编译；以 .o 为后缀时，从连接阶段开始编译。使用 -x 选项后，文件名后缀将不再对编译的起始阶段起作用。

2．-x none filenames

此选项使 -x language 选项失去作用，其后文件以名字后缀决定语言及编译的起始阶段。

3．-c、-S、-E

这三个选项决定编译过程到哪一个阶段（预处理、编译、汇编、连接）结束，它们不能同时使用。

-c 选项激活预处理、编译及汇编程序，不执行连接程序；编译过程到汇编阶段结束；输出文件为目标文件（*.o），可用 -o 选项（见下文）另行指定输出文件名。

-S 选项激活预处理及编译程序，不执行汇编和连接程序；编译过程到编译阶段结束；输出文件为汇编文件（*.s），可用 -o 选项（见下文）另行指定输出文件名。

-E 选项激活预处理程序，不执行编译、汇编和连接程序；编译过程到预处理阶段结束；输出被送到标准输出，可用 -o 选项重新定向到文件输出。

4．-o filename

此选项以 filename 覆盖 gcc/g++ 默认的输出文件名；它可以同-c、-S、-E 中的任何一个选项一同使用，为各阶段的输出文件（可执行文件、目标文件、汇编文件、预处理后的 C/C++ 语言源文件）改名。

因为 -o 选项只能指定一个输出文件名，所以当有多个输出文件产生时，-o 选项不起作用。

如不使用此选项，默认情况下，可执行程序以 a.out 命名，目标文件以 *.o 命名，汇编文件以 *.s 命名，预处理后的 C/C++ 语言源文件定向到标准输出。

5．-pipe

此选项用管道代替临时文件来处理不同编译阶段（预处理、编译、汇编、连接）间的通信。这个选项在某些不支持管道操作的系统上不工作，但在 GNU 编译器上不会有问题。

1.1.2　语言选项

1．-ansi

此选项关闭 GNU C 中与 ANSI C 不兼容的特性，激活 ANSI C 的专有特性。当使用此选项时，预编译器定义了一个宏 _STRICT_ANSI_ ，您可在源程序中加入对此宏的判断来避免使用 ANSI 标准不兼容的特性。

比如：在 ANSI C 中，禁止使用 asm、inline 和 typeof 关键字及 UNIX、vax 等预定义宏，取而代之的是 _asm_ 、_inline_ 、_typeof_ 、_UNIX_ 和 _vax_ ；这些被释放出来的关键字和宏可由您赋予新的含义。

注意：-ansi 选项并不拒绝非 ansi 程序。

2．-fno-asm

此选项实现 -ansi 选项功能的一部分，它禁止将 asm、inline 和 typeof 用作关键字。使用此选项的目的是程序已经为 asm、inline 和 typeof 赋予新的含义，不希望编译器（gcc、g++）将它们解释为关键字。

3．-fno-strict-prototype

此选项只对 g++ 起作用，对 gcc 没有影响。

通常情况下，g++ 编译器将不带参数的函数声明理解为没有参数；使用此选项后，g++ 认为不带参数的函数声明是没有显式地对参数的个数及类型进行声明，而不是没有参数。

不论是否使用此选项，gcc 编译器总是将不带参数的函数声明理解成没有对参数的个数及类型进行声明。

4．-fthis-is-variable

此选项只对 g++ 有效。

默认情况下，C++ 语言中的类已经将其对象指定给 this 指针，因此不能再将 this 用作一般变量；为了同传统 C++ 语言兼容，此选项允许将 this 用作一般变量。

5．-fcond-mismatch

此选项允许条件表达式的第二和第三参数类型不匹配。这样的表达式的值为 void 类型。

6．-funsigned-char

```
-fno-signed-char
-fsigned-char
-fno-unsigned-char
```

这四个选项对 char 类型进行设置。char 类型可能为 signed-char，也可能为 unsigned char。每种机器对 char 类型都有默认设置，使用这四个选项可以重新指定 char 类型的设置。其中前两个选项将 char 类型指定为 unsigned char，后两个选项将 char 类型指定为 signed char。

1.1.3 预处理程序选项

此类选项用于设定 gcc/g++ 在预处理阶段的行为。前文曾经讲过，-E 选项使 gcc/g++ 只执行预处理程序；此类中的部分选项必须同-E 选项一起使用，因为它们使预处理程序的输出不适合继续编译。

1．-include file

```
-imacros file
-Dmacro
-Dmacro=defn
-Umacro
```

上述五个选项的作用相当于 C/C++ 语言源文件中的 #include、#define 和 #undef 语句，但是优先于 gcc/g++ 的输入文件被处理。当 gcc/g++ 的命令行中有上述选项出现时，gcc/g++ 不论 -Dmacro、-Umacro 选项的位置如何，总是最先处理它们；然后再按照命令行中的顺序处理 -include 和 -imacros 选项；只有在命令行中的预处理选项都被处理完之后，gcc/g++ 才去处理输入文件中的预处理语句。

在上述五个选项中：-include file 选项的作用相当于 C/C++ 语言源文件中的“ #incude ”语句，导出 file 文件内容到 gcc/g++ 的输入文件中；-imacros file 选项的作用是将 file 文件中的宏定义扩展到 gcc/g++ 的输入文件中，宏定义本身并不出现在输入文件中；-Dmacro 选项的作用相当于 C/C++ 语言源文件中的“ #define macro ”语句，定义 macro 宏为串“ 1 ”或“ TRUE ”；-Dmacro=defn 选项的作用相当于 C/C++ 语言源文件中的“ #define macro defn ”语句，定义 macro 宏为串 defn；-Umacro 选项的作用相当于 C/C++ 语言源文件中的“ #undef macro ”语句，取消对 macro 宏的预定义。

2．-undef

此选项取消对任何非标准宏的定义。

```
-Idir
-I-
-idirafter dir
-iprefix prefix
-iwithprefix dir
```

这些选项限定 gcc/g++ 搜索头文件（前导文件）的路径。

C/C++ 语言源文件中约定使用 #include <file.h> 语句来包含任何由 C 编译系统提供的标准前导文件，使用 #incldue "file.h" 语句来包含您自己目录中的前导文件。