



红旗 Linux 系统 开发技术

北京中科红旗软件技术有限公司 编著

石油工业出版社

红旗 Linux 系统开发技术

北京中科红旗软件技术有限公司 编著

石油工业出版社

内 容 提 要

本书全面介绍了 Linux 系统的开发方法, 主要内容包括 Linux 的软件开发环境, 包括编译、调试、维护工具以及广泛应用的集成开发环境; Linux 的文件系统、I/O 系统调用、系统数据文件; Linux 进程模型, 包括进程关系、进程控制、进程间通信、信号处理; 终端 I/O 编程; 网络套接字编程; 守护进程、系统服务; 多线程编程等。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。

本书可作为从事 Linux 操作系统开发人员的参考用书。

图书在版编目 (CIP) 数据

红旗 Linux 系统开发技术/北京中科红旗软件技术有限公司编著.
北京: 石油工业出版社, 2013. 1
ISBN 978 - 7 - 5021 - 9388 - 1

I. 红…

II. 北…

III. Linux 操作系统

IV. TP316. 89

中国版本图书馆 CIP 数据核字 (2012) 第 293260 号

出版发行: 石油工业出版社

(北京安定门外安华里 2 区 1 号 100011)

网 址: <http://pip.cnpc.com.cn>

编辑部: (010) 64255590 发行部: (010) 64523620

经 销: 全国新华书店

印 刷: 北京中石油彩色印刷有限责任公司

2013 年 1 月第 1 版 2013 年 1 月第 1 次印刷

787 × 1092 毫米 开本: 1/16 印张: 29.5

字数: 747 千字

定价: 75.00 元

(如出现印装质量问题, 我社发行部负责调换)

版权所有, 翻印必究

前 言

Linux 操作系统是可运行在多种硬件平台的个人计算机和工作站上的类 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 操作系统，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、MIPS、SPARC 和 PowerPC 等处理器；对于 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。组成 Linux 操作系统的 Linux 内核以及各类开源组建受自由软件委员会指定的 GPL (General Public License) 公用许可协议保护，Linux 系统内核及其大多应用软件的源代码都是向用户公开的。各地区系统发布商，根据各自地域文化和技术理念，在遵循 GPL 等开源软件协议的 Linux 内核与组件的基础上，加以工程化打造而成为可产品化的 Linux 操作系统，服务于社会应用需求。用户可使用 Linux 操作系统中自带的 C/C++ 等编程语言编译器修改并编译自己喜欢的程序，同时也可以 GNU 的许可下使用这些源程序的部分或全部。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的人们的兴趣和创造力，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了相应的 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 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 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；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。在超级计算机应用领域中，以 500 强网站公布的数据可以看出，近 92% 的超级计算机系统是基于 Linux 构建的，这其中也得益于开源领域的集群技术的优势，为当今云计算技术架构的发展提供了坚实的技术基础。

不可否认，Linux 具有较 Unix 和 Windows 更卓越的性能，但价格却低得多。因此，如果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。

在国家“核心电子器件、高端通用芯片及基础软件产品”重大专项进程中，作为基础软件主要承载平台的 Linux 操作系统得到极大重视，为此，需要更多的开发人员熟悉和了解基于 Linux 系统技术体系的开发技术。为促进 Linux 在我国的推广应用，让开发人员更快地步入 Linux 的奇妙世界，少走一些弯路，北京中科红旗软件技术有限公司红旗教育学院组织经验丰富的 Linux 编程人员，参考国际经典 Linux、Unix 编程书籍，编写本书，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：

(1) 第 1 章详细介绍 Linux 的软件开发环境，包括编译、调试、维护工具以及广泛应用的集成开发环境。

(2) 第 2、3、4、5 章介绍 Linux 的文件系统、I/O 系统调用、系统数据文件方面的内容，本部分不仅详细介绍了传统 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。

(3) 第 6、7、8、9 章介绍 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。

(4) 第 10 章介绍终端 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。

(5) 第 11 章介绍网络套接字编程，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。

(6) 第 12 章介绍守护进程、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。

(7) 第 13 章介绍多线程编程，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。本书由北京中科红旗软件技术有限公司红旗教育学院组织编写，对参与本书编写的人员，在此表示感谢。由于本书涉及的内容丰富，加之篇幅、时间有限，书中难免存在纰漏和不足，欢迎读者批评指正。

北京中科红旗软件技术有限公司

2012 年 10 月

目 录

1 Linux 软件开发工具	1
1.1 gcc 和 g++	1
1.1.1 全局选项	2
1.1.2 语言选项	3
1.1.3 预处理程序选项	4
1.1.4 汇编程序选项: -Wa, option	6
1.1.5 连接程序选项	6
1.1.6 目录选项	8
1.1.7 优化选项	8
1.1.8 调试选项	8
1.1.9 警告选项	9
1.2 gnu make	9
1.2.1 make 命令的输入文件	10
1.2.2 gnu make 的使用	16
1.3 autoconf	17
1.3.1 configure.in 文件的获得	18
1.3.2 aclocal.m4 文件和 acsite.m4 文件的获得	23
1.3.3 configure 脚本的生成	24
1.3.4 configure 脚本的执行	24
1.3.5 软件包的发行和使用	26
1.3.6 autoconf 的流程框图	26
1.4 automake	27
1.4.1 编写 Makefile.am 文件	27
1.4.2 修改 configure.in 文件	32
1.4.3 执行 automake	32
1.4.4 automake 应用示例	33
1.5 gdb	33
1.5.1 gdb 的命令行选项及参数	34
1.5.2 gdb 的内部命令	36
1.5.3 gdb 应用示例	43

2 文件系统	47
2.1 文件系统简介	47
2.1.1 概述	47
2.1.2 文件模式	48
2.1.3 进程的 umask	51
2.2 Linux 文件系统简介	52
2.2.1 EXT2 文件系统的结构	53
2.2.2 超级块	53
2.2.3 组描述符	54
2.2.4 inode	54
2.2.5 目录结构	55
2.2.6 EXT2 磁盘空间的分配策略	56
2.2.7 EXT3 文件系统的改进	56
2.2.8 EXT4 文件系统	58
2.3 基本文件操作	60
2.3.1 文件描述符和流	60
2.3.2 打开关闭文件	61
2.3.3 文件共享	63
2.3.4 顺序文件读写	65
2.3.5 随机文件读写	68
2.3.6 其他文件操作	70
2.4 inode 的操作	75
2.4.1 查询 inode 信息	75
2.4.2 存取权限	78
2.4.3 文件所有权	79
2.4.4 时间属性	80
2.4.5 EXT2 扩展属性	80
2.5 特殊文件	83
2.5.1 硬连接和符号连接	83
2.5.2 创建设备文件和管道	84
2.5.3 /dev/fd	85
3 目录操作	86
3.1 当前目录	86
3.1.1 获得当前目录	86
3.1.2 设置当前目录	87
3.1.3 改变根目录	87

3.2 创建删除目录.....	88
3.3 浏览目录.....	88
3.4 名字匹配.....	92
3.4.1 使用子进程.....	92
3.4.2 内部匹配.....	93
4 高级 I/O 操作	96
4.1 同时进行多个 I/O 操作 (I/O 复用)	96
4.1.1 轮询方式.....	96
4.1.2 使用非阻塞 I/O	97
4.1.3 效率较高的 I/O 复用.....	99
4.1.4 系统调用 poll	102
4.2 内存映像	103
4.2.1 分配页面	104
4.2.2 建立内存映像	104
4.2.3 内存与磁盘的同步	107
4.3 给内存区加锁	108
4.4 文件加锁	109
4.4.1 文件锁	109
4.4.2 记录锁	110
4.4.3 死锁	114
4.4.4 锁的继承与释放	114
4.4.5 建议锁与强制锁	115
4.5 非连续区域读写	116
5 系统数据文件和系统信息	117
5.1 简介	117
5.2 密码文件	117
5.2.1 /etc/passwd 文件和 passwd 结构.....	117
5.2.2 有关的函数接口	119
5.2.3 密码的 shadow 机制	122
5.3 组 (groups)	123
5.3.1 /etc/group 文件和 group 结构	123
5.3.2 有关的函数接口	124
5.3.3 附加组	126
5.4 其他数据文件	129
5.5 有关用户登录的系统文件	129
5.6 系统标识	131

5.7 系统时钟	132
6 进程模型与进程关系	136
6.1 进程	136
6.2 线程	136
6.3 启动例程	136
6.4 终止进程	137
6.5 atexit 函数	138
6.6 命令行参数	140
6.7 环境变量列表	140
6.8 C 程序在内存中的分布	141
6.9 共享库	142
6.10 内存分配机制	143
6.11 环境变量的访问与修改	144
6.12 setjmp 和 longjmp 函数	145
6.13 使用局部变量的问题	147
6.14 getrlimit 和 setrlimit 函数	148
6.15 getrusage 函数	151
6.16 终端登录	152
6.16.1 4.3 + BSD 终端登录	152
6.16.2 SVR4 终端登录	153
6.16.3 4.3 + BSD 网络登录	154
6.17 进程组	154
6.18 会话	155
6.19 控制终端	156
6.20 tcgetpgrp 和 tcsetpgrp 函数	157
6.21 作业控制	157
6.22 程序在 shell 下的运行	159
6.23 孤儿进程组	160
6.24 4.3 + BSD 对进程关系实现	162
7 进程控制	164
7.1 进程标识	164
7.2 fork 函数	164
7.3 exit 函数	168
7.4 wait 和 waitpid 函数	170
7.5 wait3 和 wait4 函数	173
7.6 竞争条件	174

7.7	exec 函数	175
7.8	setuid 和 setgid 函数	179
7.9	setreuid 和 setregid 函数	180
7.10	seteuid 和 setegid 函数	180
7.11	system 函数	181
7.12	getlogin 函数	183
7.13	times 函数	183
7.14	守护进程	185
7.14.1	守护进程的特点	185
7.14.2	守护进程的例子	186
7.14.3	syslog 函数	187
8	进程间通信	190
8.1	管道和命名管道	190
8.1.1	管道	190
8.1.2	流管道	204
8.1.3	FIFO	205
8.2	System V IPC	210
8.2.1	System V IPC 访问方式	210
8.2.2	消息队列	213
8.2.3	信号量	219
8.2.4	共享内存	226
8.2.5	System V IPC 使用总结	237
9	信号处理	239
9.1	概述	239
9.2	Linux 系统中的信号	240
9.3	对信号的处理	242
9.3.1	设置信号处理函数	242
9.3.2	系统对信号的处理	244
9.3.3	不可靠的信号	245
9.3.4	信号的阻塞	246
9.3.5	向进程发送信号	246
9.3.6	用定时器使进程睡眠	247
9.3.7	信号与系统调用	250
9.3.8	信号集	251
9.3.9	使用信号集屏蔽信号	251
9.3.10	设置信号的处理函数	252

9.3.11	非局部跳转	254
9.3.12	屏蔽信号并使进程等待	255
9.3.13	使进程退出	256
9.3.14	等待一个进程结束	257
9.3.15	实现函数 system 的一种方法	258
9.3.16	实现函数 sleep 的一种方法	260
9.3.17	作业控制信号	261
10	终端及伪终端编程	262
10.1	引言	262
10.1.1	终端	262
10.1.2	终端驱动程序	262
10.1.3	系统与终端之间的关系	262
10.1.4	版本	263
10.2	Unix/Linux 中的终端	263
10.2.1	概述	263
10.2.2	控制终端	264
10.2.3	数据传输	265
10.2.4	正则模式和非正则模式	265
10.2.5	正则模式下的编辑键	266
10.3	终端的应用程序设计	268
10.3.1	终端的打开与读写	268
10.3.2	库函数 ttyname 和 isatty	273
10.3.3	termios 结构	275
10.3.4	利用 ioctl 系统调用对终端进行控制	281
10.3.5	通过 13 个 termios 系统调用对终端进行控制	286
10.3.6	非正则模式	298
10.3.7	终端与 SIGHUP 信号	304
10.3.8	终端窗口大小	304
10.3.9	ctermid	306
10.3.10	termcap、terminfo 和 curses	307
10.3.11	stty 命令	308
10.4	程序 tty_transfer 的设计	309
10.4.1	总体描述	309
10.4.2	头文件、常量定义和 main 函数	310
10.4.3	serial_conn 函数	312
10.4.4	文件传输函数	316

10.4.5	tty_ transfer 的使用	320
10.5	终端管理的发展	321
10.5.1	数据结构的变化	321
10.5.2	流的概念的提出	321
10.6	基于 STREAMS 的终端子系统	322
10.6.1	流的概念	322
10.6.2	基于 STREAMS 终端的优点	323
10.6.3	线路规程模块	324
10.6.4	硬件仿真模块	329
10.7	伪终端程序设计	330
10.7.1	伪终端简介	330
10.7.2	SVR4 中的 ptym_ open 和 ptys_ open	333
10.7.3	BSD 中的 ptym_ open 和 ptys_ open	336
10.7.4	pty_ fork	339
10.7.5	pty 编程举例	343
10.7.6	Linux 下的伪终端例程	348
10.7.7	远程方式和分组方式	349
10.7.8	基于 STREAMS 的伪终端子系统	349
11	socket 编程	355
11.1	协议支持	355
11.1.1	网络基础知识	355
11.1.2	Linux 系统网络模块的结构	356
11.1.3	关于网络地址	357
11.2	几个工具函数	357
11.3	socket 编程的基本流程和要用到的函数	358
11.3.1	服务器端程序的基本操作	359
11.3.2	客户端程序的基本操作	361
11.4	Unix 域 socket	361
11.4.1	Unix 域地址	362
11.4.2	Unix 域 socket 服务程序	362
11.4.3	客户程序	364
11.4.4	运行 Unix 域 socket 示例程序	365
11.4.5	用 socketpair 函数建立未命名 Unix 域 socket	365
11.4.6	用 Unix 域 socket 在进程间传递文件描述符	366
11.5	TCP/IP 网络编程	370
11.5.1	关于字节序	371

11.5.2	IPv4 地址	371
11.5.3	Socket 编程中的 IP 地址结构	372
11.5.4	十进制点式 IP 地址与二进制 IP 地址间的转换	372
11.5.5	使用域名	373
11.5.6	域名解析示例	374
11.5.7	查询服务程序的端口号	376
11.5.8	TCP 服务程序示例	378
11.5.9	TCP client application	380
11.6	socket 出错常量	381
12	守护进程	383
12.1	守护进程简介	383
12.2	syslogd 守护进程	383
12.3	syslog 函数	384
12.4	daemon_ init 函数	386
12.4.1	fork	387
12.4.2	setsid	387
12.4.3	忽略 SIGHUP 并再次调用 fork	387
12.4.4	改变工作目录并清除文件创建掩码	387
12.4.5	关闭所有打开的描述符	388
12.4.6	打开 syslog	388
12.5	Inetd 守护进程	388
12.6	如何编制一个由 inetd 启动的服务器程序	390
12.6.1	程序	390
12.6.2	配置/etc/services 文件	391
13	多线程编程	393
13.1	概念	393
13.1.1	线程	393
13.1.2	内核线程与用户层线程	394
13.1.3	进程、LWP 和线程	395
13.1.4	线程的优势	403
13.2	多线程编程基础	404
13.2.1	概念	404
13.2.2	线程的创建和终止	405
13.2.3	线程的合并和分离	408
13.2.4	线程属性	410
13.2.5	其他	414

13.3 线程之间的互斥与同步	416
13.3.1 线程互斥与同步的概念	416
13.3.2 互斥锁	416
13.3.3 利用条件变量实现同步	421
13.3.4 利用互斥锁和条件变量实现同步	426
13.4 线程编程中的其他问题	429
13.4.1 线程的私有变量	429
13.4.2 初始化函数	436
13.4.3 撤销其他线程的执行	437
13.4.4 清场函数	440
13.4.5 线程的信号操作	445
13.4.6 Semaphore. c	446

1 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；多个选项必需分开写，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：

- (1) 全局选项：影响编译的全过程（从预处理到连接）；
- (2) 语言选项；
- (3) 预处理程序选项；
- (4) 汇编程序选项；
- (5) 连接程序选项；
- (6) 目录选项；
- (7) 优化选项；
- (8) 调试选项；
- (9) 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

1.1.1.1 -x language filenames

-x language filenames 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。

1.1.1.2 -x none filenames

-x none filenames 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。

1.1.1.3 -c、-S、-E 选项

-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。

1.1.1.4 -o filename

-o filename 选项以 filename 覆盖 gcc/g++ 缺省的输出文件名；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，缺省情况下，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。

1.1.1.5 -pipe

-pipe 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）