万水计算机实用编程技术系列

UNIX网络实用编程技术

陈远森 邓可 杜威 等编著

中国水利水电出版社
www.waterpub.com.cn

万水计算机实用编程技术系列

UNIX 网络实用编程技术

陈远森　邓可　杜威　等编著

中国水利水电出版社

内 容 提 要

随着计算机网络的迅猛发展，网络应用的日益增多对软件开发人员提出了新的要求，要求他们熟悉并掌握网络编程技术。为顺应技术发展趋势，本书详细地介绍了 UNIX 系统环境下的网络编程技术。全书由浅入深、全面介绍了如何使用 Socket（套接字）来编写网络应用程序，系统地介绍了 TLI、Streams（流）编程和 RPC（远程过程调用）编程技术，同时结合了大量实例进行说明。

本书可以作为希望进入网络编程世界的人们从入门到精通的台阶，也可供从事网络编程的人员参考，同样适合大专院校各年级学生配合操作系统和网络原理的学习和使用。

图书在版编目（CIP）数据

UNIX 网络实用编程技术/陈远森　等编著. —北京：中国水利水电出版社，2000. 5

（万水计算机实用编程技术系列）

ISBN 7-5084-0355-X

I. U…　II. 陈…　III. UNIX 操作系统-程序设计　IV. TP316. 81

中国版本图书馆 CIP 数据核字（2000）第 08885 号

书　　名	UNIX 网络实用编程技术
作　　者	陈远森　邓呼　杜威　等
出版、发行	中国水利水电出版社（北京市三里河路 6 号　100044） 网址：www.waterpub.com.cn E-mail：sale@waterpub.com.cn 电话：（010）63202266（总机）、68331835（发行部）
经　　售	全国各地新华书店
印　　刷	北京市天竺颖华印刷厂
规　　格	787×1092 毫米　16 开本　21.5 印张　488 千字
版　　次	2000 年 5 月第一版　2000 年 5 月北京第一次印刷
印　　数	0001-5000 册
定　　价	35.00 元

凡购买我社图书，如有缺页、倒页、脱页的，本社发行部负责调换

前　言

随着网络应用发展的突飞猛进，越来越多的应用转移到网络平台上来，如丰富多彩的 Web 主页、信息繁多的 BBS、方便快捷的电子邮件、不受地域限制的全球聊天室等，网络已经日益成为人们生活不可缺少的一部分。越来越多的应用软件采用客户/服务器模式和浏览器/服务器模式，采用这两种模式的软件都离不开网络编程，而这就要求开发人员迅速熟悉并掌握网络编程技术。

在众多适合网络编程的操作系统里，UNIX 系统因其具有简洁、高效、使用方便和可移植性好等优点而得到广泛的应用，它的另一个显著特点是其强大的网络功能，不但支持多种网络协议，而且还提供丰富的网络编程接口。在这些编程接口里，最常用的是 BSD 的 Socket（套接字）、AT&T 的传输层接口（TLI）和远程过程调用接口（RPC）。

随着大量的自由操作系统如 Linux、FreeBSD 等的出现，UNIX 不再为工作站和专用机所独有，在 PC 上建立 UNIX 操作系统已成为可能，这就大大扩展了 UNIX 的应用范围。同时从操作系统的性能和安全性来看，UNIX 系统经历了数十年的发展，已经成为一个成熟健壮的操作系统，相比之下，Windows NT 发展时间还不长，虽然采用了很多新技术，但相对 UNIX 来说还远谈不上成熟。正因为如此，现在因特网上的大型网站大都构架于各种 UNIX 系统上。

由于以上优点，UNIX 系统理所当然地成为网络服务的中坚，掌握 UNIX 下的网络编程成为一个优秀的网络程序员的必然要求。另外，从网络的发展史上看，UNIX 系统也是孕育现在的因特网的母体，网络运作的原理和过程在 UNIX 中清晰地表现了出来。掌握了 UNIX 下的网络编程，也就掌握了网络编程的精髓，再学习 Windows 等系统下的网络编程就非常轻松了。

本书是针对有一定 C 语言和网络知识基础的读者而编写的，全书分为三部分，共 12 章。前两章为准备知识，介绍 UNIX 操作系统及其中的 C 语言编程、调试、管理工具，熟悉这些内容的读者可以跳过这两章。从第三章到第七章为第一部分，由浅入深地介绍 Socket 编程，是全书的重点：第三章介绍了网络的体系结构及各种协议，第四章介绍面向传输层的 Socket 编程，第五章介绍面向网际层和数据链路层的 Socket 编程，第六章介绍 Socket 编程的一些高级专题，第七章介绍 Socket 实用编程；第二部分包括第八章、第九章和第十章，分别介绍 TLI 编程和流编程；第十一章和第十二章组成第三部分，这部分介绍 RPC 编程。最后，本书的附录部分还给出一些有关常见的编程问题的解答，并列举了一些 Internet 上关于 UNIX 网络编程的一些资源，读者可通过 Internet 访问这些资源，获得更多的资料和信息。

本书遵循循序渐进、由浅入深的原则来编写，结合程序介绍原理，力求清晰易懂。在编写过程中，使用了 SCO UNIX 5.0.4、Solaris 7 和 Linux 2.2.6 三个操作系统来调试程序。

虽然每一个程序都在机器上调试运行过，但错误和缺漏在所难免，敬请读者批评指正。

本书由陈远森、邓可和杜威等人编著。另外参加编写的还有：胡玉中、朱仲、石文、伊任、赵河冰、程江华、蔡未华、上官杰云、朱开元、宋荣、郭旷、赵子治、钱晓红、刘亦建、周星云等同志。

本书的出版得到了中国水利水电出版社北京万水电子信息有限公司多位编辑等人的悉心指导和大力支持，他们为本书的出版付出了辛勤的劳动。在此表示由衷的感谢。

由于时间仓促、作者水平有限，本书错漏之处在所难免，欢迎广大读者批评指正。

编者

2000 年 2 月

目　录

前言
第一章　绪论 1
1.1 UNIX 操作系统 1
1.2 计算机网络 2
1.3 网络编程和套接字 3
1.4 本书的组织 3
第二章　UNIX 系统下常用的 C 语言开发工具 4
2.1 版本管理工具——sccs 4
2.2 源程序静态检查工具——lint 9
2.3 编译器——cc 11
2.4 程序组维护工具——make 15
2.5 调试工具——dbx 19
2.6 库管理工具——ar 24
2.7 小结 25
第一部分　Socket 程序设计
第三章　网络体系结构概述 26
3.1 分层结构与协议 26
3.2 开放系统互联参考模型 OSI/ISO 27
3.3 TCP/IP 参考模型 28
3.4 UNIX 网络编程概述 30
3.5 小结 31
第四章　面向传输层的 Socket 编程 32
4.1 TCP 协议的实现机制 32
4.2 面向连接的 Socket 编程 34
4.2.1 Echo 客户端例程 34
4.2.2 Echo 服务端例程 39
4.2.3 进程阻塞 42
4.2.4 并发服务端程序 43
4.3 TCP 状态 48
4.4 Socket 选项 51
4.4.1 应用层选项 51
4.4.2 传输层选项 54
4.4.3 网际层选项 56
4.4.4 其他选项 57

4.5 无连接的 Socket 编程 57
4.5.1 UDP 编程概述 58
4.5.2 UDP 编程的第一步：Talk（谈话）程序 59
4.5.3 改进 Talk 程序：过滤数据报 62
4.5.4 进一步改进 Talk 程序：超时机制 63
4.5.5 使 UDP 更加可靠 67
4.6 小结 68
第五章 面向网际层和数据链路层的 Socket 编程 70
5.1 网际层综述 70
5.2 面向 IP 层的 Socket 编程 72
5.2.1 IP 数据报报头格式 73
5.2.2 ICMP 数据报格式 80
5.3 数据链路层简述 88
5.4 面向数据链路层的 Socket 编程 89
5.5 小结 95
第六章 Socket 高级编程专题 96
6.1 多路复用 96
6.1.1 多路复用的基本原理 96
6.1.2 使用多路复用的 UDP Talk 程序 98
6.1.3 多路复用应用举例：Chat 100
6.1.4 多路复用的另一种方式：poll() 107
6.2 非阻塞 Socket 108
6.2.1 非阻塞 Socket 的基本原理 108
6.2.2 非阻塞 UDP Socket 例程——Talk 109
6.2.3 非阻塞 TCP Socket 例程——echo 客户端程序 112
6.3 信号驱动输入输出 116
6.3.1 信号驱动 I/O 的基本原理 116
6.3.2 信号驱动 I/O 的 UDP Talk 117
6.3.3 信号驱动 I/O 在 TCP 中的应用 122
6.4 名字服务 127
6.4.1 名字服务的原理 127
6.4.2 名字服务例程 131
6.5 广播与群播 133
6.5.1 广播的基本原理 133
6.5.2 广播例程 135
6.5.3 群播的基本原理 138
6.5.4 群播例程 139

6.6 OOB 数据 141
6.6.1 OOB 数据的基本原理 142
6.6.2 OOB 数据应用例程 146
6.7 小结 155
第七章 Socket 实用编程 157
7.1 进程间通讯 157
7.1.1 用 Internet 协议族的 Socket 实现 IPC 158
7.1.2 用 UNIX 协议族的 Socket 实现 IPC 162
7.2 客户/服务器程序设计 171
7.2.1 用多进程机制实现服务程序 171
7.2.2 用多线程机制实现服务程序 177
7.3 守护进程 181
7.3.1 守护进程的设计原理 182
7.3.2 inetd 守护进程简介 184
7.4 平凡文件传输协议 187
7.4.1 TFTP 数据报格式 188
7.4.2 “发送”和“接收”请求的处理 190
7.4.3 服务方和客户端的通讯 190
7.4.4 文件传输格式 191
7.4.5 TFTP 协议的实现 191
7.5 小结 213
第二部分 传输界面编程
第八章 传输界面编程综述 214
8.1 传输界面 214
8.2 函数 216
8.2.1 初始化及绑定传输端点 216
8.2.2 建立连接 218
8.2.3 传输数据 221
8.2.4 释放连接 226
8.2.5 关闭传输端点 228
8.2.6 其他函数 229
8.3 状态转换 234
8.4 小结 238
第九章 传输界面编程 240
9.1 连接方式 240
9.2 非连接方式 254
9.3 传输界面编程与 Socket 编程对比 258

9.4 小结 260
第十章 流编程 261
10.1 流编程原理 261
10.2 流编程例程 263
10.3 小结 270

第三部分 RPC 程序设计

第十一章 远程过程调用 271
11.1 RPC 概述 271
11.2 RPC 的原理和实现机制 273
11.2.1 RPC 的实现机制 273
11.2.2 RPC 的有关问题 274
11.3 XDR 数据表示 275
11.3.1 XDR 的工作原理 276
11.3.2 XDR 流 277
11.3.3 XDR 过滤器 279
11.4 RPC 协议 283
11.4.1 RPC 信息 283
11.4.2 鉴别协议 284
11.4.3 端口映射器程序协议 284
11.5 小结 285
第十二章 RPC 程序设计 286
12.1 RPC 的远程过程定义 286
12.2 高层 RPC 程序设计 287
12.3 低层 RPC 程序设计 300
12.3.1 传送句柄和客户句柄 306
12.3.2 server 端库函数 307
12.3.3 client 端库函数 308
12.3.4 server 端的分派函数 310
12.4 鉴别机制的编程 312
12.5 RPC 的其他特性 314
12.5.1 无阻塞 RPC 315
12.5.2 回叫 RPC 315
12.5.3 广播 RPC 316
12.6 RPC 语言编译器 317
12.6.1 低层 RPC 程序的自动生成 317
12.6.2 RPC 语言 319
12.7 小结 320

附录 A socket 编程常见问题解答 321
附录 B Internet 上有关 UNIX 网络编程的资源 332
附录 C 参考文献 334

第一章　绪论

现在的操作系统都具备网络功能，而且网络部分是操作系统的一个重要组成部分。学习网络编程将不可避免地涉及到操作系统的一些知识，要深入地学习网络编程，网络知识也是必不可少的。在这一章里，将对 UNIX 操作系统、计算机网络以及网络编程作一简单的阐述。

1.1　UNIX 操作系统

UNIX 是一个通用的交互式多用户分时操作系统。它首先由 AT&T 公司 Bell 实验室的 Ken Thompson 于 1969 年在 PDP-7 小型计算机上开发出来，然后在 1971 年被移植到 PDP-11 机器上。当 Dennis Ritchie 为 UNIX 开发出 C 语言编译器后，他和 Ken Thompson 就一起用 C 语言重新设计了 UNIX 系统。由于 C 语言具有良好的可移植性，所以 UNIX 系统很容易被移植到其他机器上去，从而极大地促进了 UNIX 系统的推广。

UNIX 系统一开始并不作为商业软件出售，它除了在 Bell 实验室内部使用外，还被免费赠送到各大学供教学和科研之用。由于 UNIX 具有简洁、高效、使用方便和可移植性好等优点，深受用户欢迎，获得巨大的成功，并因此而流传开来。很多学生在现有的 UNIX 系统上做进一步的开发，给它增加了许多新的功能，其中的一些功能后来被添加到正式发行的版本中去。由于很多学生在校时已习惯于使用 UNIX 系统，他们毕业后也希望能继续使用该系统。在这种情况，AT&T 于 1981 年发布了商用版本 UNIX System III，并且从 1984 年起开始对 UNIX 进行商业化支持。

随着计算机的迅速普及，在各大学对 UNIX 系统进行使用和开发的同时，很多计算机公司也相继把 UNIX 系统移植到自己开发的计算机上去。于是在短短的时间内，就开发出了许多 UNIX 的“变种”，出现了“百家争鸣”的局面。在这众多的 UNIX 版本里，影响最大的是 AT&T 公司和加州大学伯克利（Berkeley）分校的版本。众多的厂家和用户竞相开发，使得 UNIX 系统得到进一步的发展和完善，但随之而来的问题就是各版本之间的兼容性问题，用户迫切需要一个标准化的版本。

在这种情况下，AT&T 公司融合了 BSD 版（伯克利分校版本的简称）的功能，于 1987 年发布 UNIX 4.3BSD 并取得了巨大的成功。这个版本的成功，使得 UNIX 系统在全美国各大学里迅速得到推广。1989 年，UNIX 国际组织（UNIX International）博采众家之长，主要是融合了 Microsoft 的 Xenix、SUN OS、UNIX System V 和 UNIX 4.3BSD 的功能，推出 UNIX System V Release 4.0（简称 SVR 4.0），随后又在 1992 年发布了 SVR 4.2 版。这两个版本不但在功能上较之以前的版本有所增强，而且都符合 POSIX 和 X/OPEN 的国际标准，对 UNIX 系统和软件业的发展有着深远的影响。

在 UNIX 系统不断向前发展的同时，UNIX 系统的标准化工作也在进行着。POSIX 标

准和 X/OPEN 标准即是两个 UNIX 系统的标准。POSIX（Portable Operating Systems Interface）是由 IEEE 制定出来的标准，而 X/OPEN 则是国际上一些著名的计算机厂商所采用的 UNIX 系统标准。由于这些计算机厂商的影响，它们所采纳的标准和 IEEE 制定出来的标准也一样具有较强的权威性，所以一般计算机厂商推出自己的 UNIX 产品时，总是声称自己的产品符合 POSIX 标准和 X/OPEN 标准。

UNIX 系统由于其简洁的设计、强大的功能、良好的可移植性和开放性，在众多的操作系统中脱颖而出，成为一个公认的优秀的操作系统。在开放系统已成为计算机技术和产品主流的今天，UNIX 的技术和产品亦将成为主流产品，具有很强的发展潜力。

1.2 计算机网络

在计算机发展的早期，由于计算机价格昂贵、数量较少，那时候计算机最常见的使用方式是一台主机带若干个终端。在这种使用方式下，每台主机是一台孤立的计算机，不和其他主机互连。随着计算机数量的逐渐增多，使用的人数也随之增多，用户之间就有了相互交流和交换信息的需求，这种需求就是计算机网络出现的源动力。

在网络发展的初期，计算机网络主要是用来连接各主机（一般是中大型机）。主机之间通过电话线连接，一台机器上的用户可通过电话线接收来自其他主机上用户发来的电子邮件，也可以从其他机器上拷贝文件，UNIX 系统上的 mail 和 uucp 命令就是用来完成这两个功能的。到后来，个人计算机以其低廉的价格使得计算机的数量急剧增长，同时也由于网络技术的成熟，计算机网络得到了飞速的发展，计算机的应用出现了一次革命。到现在，计算机网络无所不在，从银行的专用系统到日常生活中经常接触到的火车、汽车联网售票系统，直至一个小游戏厅里的几台计算机，无不可以找到计算机网络的影子。当今，计算机技术，尤其是网络技术，正在逐渐渗入人们的生活，成为日常生活中不可缺少的一部分。

人们组建计算机网络的主要目的是进行通讯，而两台计算机间要能够进行正常的通讯就必须遵守一定的“规则”，这些“规则”就是网络协议。为了简化网络协议的设计，通常把协议分成若干层（可参看第三章的 OSI/ISO 模型），分别规定了每层要实现的功能。由于同一功能可用不同的方法实现，所以每层可使用不同的协议。如对应于 OSI/ISO 模型的网络层和传输层，就可使用 IPX/SPX 或者是 TCP/IP 协议，前者是 NOVELL 网的协议，而后者是现在国际互联网（Internet）所使用的协议。值得一提的是 TCP/IP 协议，该协议最早被应用于 Internet 的前身 ARPA 网，该网由美国国防部（DOD）组建，所以常将 ARPA 网称为 DARPA 网，而将这种网的体系结构称为 DOD 结构。TCP/IP 协议经过几十年的运行，被证明是可靠的，所以现在除了在广域网内广泛使用 TCP/IP 协议外，在局域网范围内应用也很广泛。考虑到实用性，本书将主要介绍 DOD 网络体系结构的网络协议，在这过程中也涉及到 OSI/ISO 的网络参考模型。

计算机网络较之单机运行方式，有很多优点，这些优点可通过下边一些典型的网络应用看出：

- 信息发布——使用 WWW 方式在网上发布信息。
- 电子邮件——与其他机器上的用户交换电子邮件。
- 共享外设——通过网络使用其他机器的外部设备。
- 交换文件——通过 ftp、rcp（remote copy）等命令在不同的机器间拷贝文件。
- 在别的机器上运行程序——利用远程过程调用在其他机器上执行命令。

以上这些应用都是提供给用户直接使用的，它们的实现都基于一定的网络协议（如 TCP 或 UDP 协议）。在编写这些程序时都调用了网络协议提供的编程接口（库函数），如何使用这些编程接口就是本书要叙述的内容。

1.3 网络编程和套接字

Client/Server 模式是当今被广泛采用的一种计算模式。在这种模式下，客户程序（Client）向服务程序（Server）发请求，服务程序接收来自客户程序的请求并进行处理，处理完后把结果回送给客户程序。在这种模式下，涉及到两个程序（确切说应该是两个进程）间的通讯，所以如果采用 Client/Server 模式进行编程，那就必须解决好进程间的通讯问题。如果服务进程和客户进程都在同一台机器上，可有多种方式实现进程间的通讯；但如果这两个进程分别驻留在通过网络连接的不同的机器上，那么在编写服务程序和客户程序时将不可避免地要涉及到网络编程。由于 Client/Server 模式很常见，所以大量的程序编写涉及到网络编程。

网络编程并不是要实现各协议层的功能，而是如何利用各协议层所提供的功能实现自己的应用，不过要更好地进行网络编程，必须具备一定的计算机网络知识。操作系统一般都把各协议层的功能作为系统内核的一部分来实现，应用程序只需调用系统提供的编程接口即可。各操作系统厂家在实现这些功能时实现机制可能不完全一样，接口的使用也不尽相同。在 1.1 节介绍 UNIX 操作系统时，提到了 UNIX 的 BSD 版本和 SVR 4，BSD 版提供的一套网络编程接口称为“套接字”（socket），而 SVR 4 的网络编程接口称为“传输界面”（TLI：Transfer Layer Interface）。在这两者中，socket 由于 BSD 版本的影响而应用得更广泛些。本书将就 socket 和 TLI 编程分别进行介绍。

1.4 本书的组织

正如这本书的书名一样，本书将主要介绍如何在 UNIX 系统下进行网络程序设计。本章后面的第二章是编程的准备知识，介绍 UNIX 系统下的 C 语言的开发工具，读者具备这些知识后，即可着手进行网络编程。网络编程共分三部分：第一部分是套接字编程，介绍如何在传输层、网际层和数据链路层使用套接字进行编程，这也是本书的重点；第二部分介绍 SVR 4 的传输界面编程（TLI）；第三部分介绍 RPC 编程。

第二章　UNIX 系统下常用的 C 语言开发工具

在 UNIX 系统下编程，除了可使用像 shell、awk 和 perl 等开发工具外，还可以使用诸如 Fortran、C 等高级程序设计语言。在众多的高级程序设计语言里，C 语言由于其功能强大、高效、可移植性好等特点而倍受青睐，本书将介绍如何以 C 语言为开发工具来实现 UNIX 系统下的网络编程。“工欲善其事，必先利其器”，在具体介绍网络编程之前，有必要先介绍一下 UNIX 系统下常用的 C 语言开发工具。

和其他 C 语言的开发平台一样，UNIX 系统也提供了丰富的开发工具，如编译器、调试器和版本管理工具等。但需要注意的是：虽然 C 语言是平台无关的，而且大部分系统的编译器都支持 ANSI C 的标准，但开发工具的具体用法却与操作系统有很大的关系，不同厂家的 C 语言开发工具提供的功能不尽相同，使用起来也会略有差异。本章将介绍 UNIX 系统下经常用到的几个 C 语言开发工具，读者具备这些知识后可在后边的学习中加以运用并逐步掌握。

编写一个程序一般要经过如下几个阶段：编辑、编译和调试。本章将循着这条主线来介绍常用的开发工具。在编辑阶段引入版本管理，介绍 sccs 的用法；在编译阶段介绍编译器 cc、与 cc 紧密相关的程序组维护工具 make 及其描述文件的编写；在调试阶段将介绍调试器 dbx 的用法，最后还将就库文件管理工具 ar 作一简单介绍。所有这些工具的用法均结合简单的例子来说明。

2.1　版本管理工具——sccs

在开发软件过程中，一个产品可能要经过若干次修改，从而形成多个版本。每一版本都跟前一版本有所不同，或是修正了前一版本的某些错误，或是增加了某些新的功能。在一个产品的开发过程中经常会出现类似于这样的问题：在版本 3.0 以前一直正常的某个功能在版本 3.0 中却无法使用，这时就得回溯到 3.0 以前的各个版本的程序中去，看在哪一个版本开始增加这个功能，在哪一版本中开始引入了错误，而这就要求能取得以前的各个版本。若采用拷贝文件的方式来保留不同的版本，不但费时、费力，还浪费磁盘空间，而版本管理工具 sccs（Source Code Control System）就有效地解决了这个问题。用这个工具来管理一个由群体开发的软件，或是为多种操作平台开发的软件，效果尤为显著。

版本管理工具 sccs 相当于一个数据库管理系统，它把源文件及其变动记录到一个数据库文件（称为 sccs 文件）里去，并提供了一个命令集来管理该数据库文件。用编辑器生成一个源程序后，用 admin 命令将源程序文件入库，创建 sccs 文件（前缀为 s.），并给出一个标志号 SID 作为该版本的版本号。以后每次修改源程序之前，均需用 get 命令从 sccs 文件中取出某一个版本的源程序，然后才能进行修改，修改完毕要用 delta 命令将变动记

录到 sccs 文件中去，存为一个新的版本，该版本对应于一个新的标志号 SID。标志号 SID 由发行版本号、层号、分支号和序号组成（一般只使用前两项），如标志号 1.2.3.4 的具体解释为：

1	2	3	4
发行版本号	层号	分支号	序号

除了 SID 外，sccs 还有一个重要的概念是 ID 关键字，它具有“%C%”的形式，其中“C”为某一大写字母。可以把 ID 关键字嵌在源程序的注释中，当把源文件记录到 sccs 文件中时，sccs 将用相应的值取代这些关键字。如果源程序中没有关键字可被替换记录，在把源程序文件记录到 sccs 文件中时，系统将发出警告。常用的关键字如下：

关键字	值
%B%	分支号
%E%	最近修改日期（YY/MM/DD）
%I%	SID 标志号
%L%	层号
%M%	sccs 文件的名字（不包括前缀“s.”）
%R%	发布版本号
%S%	序列号
%T%	最近修改时间（HH：MM：SS）

sccs 命令的用法如下：

sccs 命令 [选项] [文件...]

在上述 sccs 命令行中，“命令”不可默认，经常使用的命令有如下一些：

- admin 创建新的 sccs 文件或修改已存在的 sccs 文件。
- cdc 改动某个版本的注释。
- delta 把修改过的源程序文件记录到 sccs 文件中去。
- get 从 sccs 文件中取出某一个版本的源文件，当加上-e 选项时，取出的源文件可被修改，否则为只读文件。
- prs 显示 sccs 文件中各个版本的信息。
- rmdel 从 sccs 文件中删除一个版本。
- sccsdiff 显示 sccs 文件中两个版本间的异同。
- unget 在把源文件的改动记录到 sccs 文件之前，取消这次改动。

下面通过一个简单的例子来说明以上这些命令的用法，假设有一个 C 源程序文件 test.c，为方便说明在每行前加上行号，其内容如下：

```
1  /* %B%  %E%    %I%    %L%    %M%    %R%    %S%    %T% */
2  #include <stdio.h>
3
4  int main(void)
```

```
 5  {
 6      int num1,num2;
 7
 8      printf("Please input an interger: ");
 9      scanf("%d",&num1);
10      printf("Please input another interger: ");
11       scanf("%d",&num2);
12
13       printf("The sum of %d and %d is %d\n",num1,num2,num1 + num2 );
14       return 0;
15  }
```

编辑完源文件 test.c 后，在命令行执行以下命令（以字符$开始的是命令行，其他为系统显示的信息）：

```
$mkdir sccs
$sccs admin -itest.c test.c
$ls -l sccs
total 1
-r--r--r--    1 cys        group          461 Apr   5 10:54 s.test.c
$rm test.c
```

首先在当前目录下建立一个名为 sccs 的子目录，sccs 文件将默认地放在该目录下，并具有“s.”前缀；然后用 admin 命令（注意“-i”选项和 test.c 中间没有空格）把源程序 test.c 记录到 sccs 文件中去，ls 命令显示确实在 sccs 子目录下生成了 sccs 文件 s.test.c；最后，当确保 sccs 文件已生成后，可把源文件 test.c 删去。

```
$sccs get test.c
1.1
15 lines
$ls -l test.c
-r--r--r--    1 cys        group          320 Apr   5 10:55 test.c
$cat test.c
/* 0     00/04/05        1.1      1       test.c  1       0       11:18:04 */
#include <stdio.h>

int main(void){
    int num1,num2;

    printf("Please input an interger: ");
    scanf("%d",&num1);
```

```
    printf("Please input another interger: ");
    scanf("%d",&num2);

    printf("The sum of %d and %d is %d\n",num1,num2,num1 + num2 );
    return 0;
}
```

get 命令根据 sccs 文件 s.test.c 在当前目录下生成文件 test.c，系统显示当前 test.c 的版本号为 1.1，共有 15 行。命令 ls 的结果表明源文件 test.c 已生成（注意前边已把源程序文件删除），且为只读文件。cat 命令显示生成的源文件的第一行中的关键字已被替换。

```
$sccs get -e test.c
1.1
new delta 1.2
15 lines
$ls -l test.c
-rw-r--r--    1 cys        group          315 Apr   5 11:11 test.c
```

在 get 命令加上选项-e 再执行一遍，和上面一样，系统仍然显示当前 test.c 的版本号为 1.1，共有 15 行，唯一不同的是多了一行“new delta 1.2”，该行指出：新生成的 test.c 的版本号为 1.2，即当用 delta 命令再次把 test.c 记录到 sccs 文件中去时，它的版本号为 1.2；命令 ls 显示生成的 test.c 文件可修改。

在刚生成的源程序 test.c 的 13 行之后加上一行语句：

```
printf("The difference between   %d and %d is %d\n",num1,num2,num1 - num2 );
```

然后用 delta 命令将修改记录到 sccs 文件中去，如下所示：

```
$sccs delta test.c
comments Add the capability to subtract
1.2
1 inserted
0 deleted
15 unchanged
```

delta 命令要求输入新版本的注释，在输入注释“Add the capability to subtract”后指出该 test.c 的版本号为 1.2。和原来的 1.1 版相比，该版本插入了一个新行，所以 1.2 版的 test.c 应该有 16 行。

除了可以使用系统给定的版本号外，还可以在 get 命令里加上选项-r 来指定新的版本号。下面命令指定 test.c 的发行版本号从 2 开始：

```
$sccs get -e -r2 test.c
1.2
new delta 2.1
16 lines
```