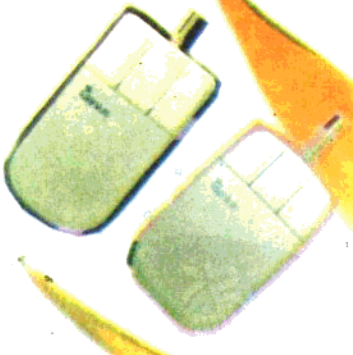
UNIX 系统源代码控制系统下的源文件管理

孙德和 编
王 岩 审校



北京希望电脑公司

UNIX 系统源代码控制系统下的源文件管理

孙德和　　编
王　岩　　审　校

北京希望电脑公司

前 言

源文件管理系统是 UNIX 系统中的一个重要环节，尽管目前市场上 UNIX 的书籍大量涌现，但有关源文件管理方面的问题仍没有一个详尽的论著．不少工作人员因缺乏这方面的知识而陷于被动．本书是管理一个 UNIX 代码源树和自动处理版本控制任务的工具．不论是软件管理员、系统构造管理员、还是系统管理员，使用该实用指南将有助于解决许多源文件组织问题．该书在 UNIX 环境中开发构造、维护和使用源文件库，并且包括了 SCCS 的指令．本书要求读者有基本的 UNIX 外壳编程和 C 语言编程知识．通过学习本书，你可以：

- 用SCCS为一个产品管理源文件，以便跟踪整个应用的版本而不只是单个源文件．
- 实现 SUID 界面程序以及外壳原本的源代码．
- 用过程建立一个源库，把所有应用放到一单个库中．
- 为整个产品建立一个由SCCS文件组成的源文件，并且将产品集中到一个源库中．

本书使用下述约定来演示 UNIX 命令．

- 在举例语句中，必须输入的字母用黑体表示．
- 用花括号"{ }"括起来的字是命令行参数，将要由适当的正文取代．
- 方括号"[ ]"指示括起的正文是可选择的．多个选择项之间用竖杠"|"分开．
- 省略号"(...)"指示前面的项可以重复．
- 括在尖括号"< >"中的正文指示其内容由执行的命令确定的变量．
- 所有的例子都使用 Bourne 外壳．外壳命令行的开始由 $ 指示．
- C 程序都已经在系统Ⅴ／386 版本 3 中编译并测试过了．

编者

目 录

第一章　为什么要进行源文件管理

1.1　源文件的定义

对于任何工作，要保持其清晰度，就必须对其有一定程度的通盘了解．例如在计算机这个领域，旧的术语含义不断更新，即使下最共用的定义也是十分困难的．因此，为确保清晰，本书中使用了下述基本术语的定义，根据需要，还将定义与源文件管理的特定范围有关的附加术语．

在本书中与我们有关的文件称源文件（source file），它是一个公用术语，用来描述动态应用文本文件．要定义动态应用文本文件，必须先定义文本文件（text file）．包含原始的字母数字式字符，包括标点符号，也可以包含用来控制打印的特殊字符的文件叫文本文件．这些文件通常使用文本编辑器或字处理器来建立和修改．

应用文本文件（application text file）是用户为某项应用而建立和修改的文本文件，同如／etc／passwd 的特殊系统文件正好相反．当应用文本文件不再修改时，就成为静态文件（static file）．这些文件可能包括函件、书本、杂志文章等．在处理信件的过程中，可以进行校订修改，但一旦信件完成并发送出去，文本文件就不能再修改了．从这个意义上讲，这个文件就变成静态应用文本文件（static application text file）．

一些应用文本文件连续经过几个周期修改后，存在几种不同的版本，这种类型的文件我们叫做动态应用文本文件（dynamic application text fSile）．因这些文件通常是程序的来源，可通称它们为源文件，但源文件不仅仅限于程序，也可以包括文档文件．

因一个源文件可能有多个版本，有必要定义一些术语来描述不同版本之间的关系．原始的源文件叫做父源文件（parent source file），修改后的源文件叫子源文件（child source file）．正常情况下一个父源文件仅有一个子源文件．如果一个父源文件有多个子源文件，那么额外的子源文件就称为分支（branch）．因此，一个源文件修改后，它就会有许多后代，这些后代称为子孙（progency）．

一个源文件很少孤立存在，通常几个源文件通过各种工具进行组合而形成产品（product），有时也称为系统、软件包或应用．例如，一个产品可以是会计帐目统计软件包，也可以是参考手册．正象源文件能够形成产品一样，一项产品也有每个父亲通常只有一个孩子那样的血统关系，然而正象源文件那样它也可能有许多分支．

1.2　促使源文件动态的原因

为了说明问题，我们考虑一个软件系统以及相关的文件．这个软件包叫做最终编辑器（Ultimate Editor），根据设计说明书，这个编辑器优于所有其它编辑器．

经过多年的工作之后，这个最终编辑器完成并向全世界发放．所有工作完成后，我们说工作完毕（done)；所有工作完成，是这样的吗？通过它能得到所有的文件吗？随着时间的推移，又提出了附加的要求，计算机环境改变后，少数未查出的错误又暴露出来．现在又有必要开始研制另外一个版本，除了上一个版本的源代码和文件作为起点外，又一个周期开始了．

这种自身不断重复地修改过程贯穿于软件产品的一生．事实上，这个产品永远不会完工，永远没有最终结束，除非它停止使用．正是这个连续不断的变化过程使那些源文件构成了动态应用文本文件．

1.3 管理修改的必要性

自从有计算机以来，软件管理者就不得不处理软件连续修改的问题，这个问题我们称为软件信息量（software entropy)，正象众所周知的那样，"极度混乱"．这个信息量等于源文件修改的次数乘以分支数．从下面的两个例子中可以看出，在没有源文件修改管理的时候，软件信息增长得有多么快．

在A公司，顾客是其各部门（例如财会，市场，制造)．在A公司里，源文件通常有一系列的世系关系（第3版本基于第2版本，而第2版本又基于第1版本)．在这种环境中的软件信息量呈现出后代减少的形式，如第3版本的某个特别源文件是基于第1版本而不是基于第2版本．在这种情况下，当几个组共享部分公用源文件时就可能出现错误．A组通过修改第1版的源文件而建立了第2版，而B组在不知道A组已做了任何修改之后，又修改了第1版本而生成了另一个新版本．这种情况下，哪一个版本是下一代的父亲呢？(在第十章，将阐述怎样避免这种错误)．

B公司是一家服务社或软件室．对于这种类型的公司，由于其特殊的应用或不同顾客的硬件需要，其源文件可能有很多的分支．当每个分支在其不同开发组的范围内时，软件信息量将增加得很快．最终的结果就是每个分支开发自己的生命周期．对于公用源文件一旦修改，信息量与分支数将成倍增长．通常，并不是所有的分支都修改，或者对某一特定分支来说这种修改与另外的修改相矛盾．这种情形即使在有源文件管理系统的公司中也可能出现，因为正向我们将看到的那样，源文件的管理比拥有一个可以信赖的源文件库更复杂．

每次有源文件损失时，控制它就要花费金钱．源文件的维护和增强已是费钱的事，但糟糕的源文件管理并不一定导致更多的花销．

下面讨论源文件控制系统的需求．学习了这些需求之后，在那些没有软件控制的地方，就可以想象得出所要承受的开销．

1.4 源文件控制系统的需求

正如上面提到的那样，源文件修改的管理，不论是源程序还是其它文档文件，恰好就

是不同版本的档案管理．如果说保存一个历史拷贝仅仅是其结果的话，源文件控制系统就将有两个需求：保留新版本的能力和恢复先前版本的能力．但是，正如下面的需求列表中所显示的那样，源文件管理是多用途的．

1.4.1 恢复任何先前版本的能力

对下一代的源文件来说，最重要的源文件版本是其父版本．为理解这种需求，可以回顾一下动态文本文件与静态文本文件的不同．

在开发阶段（例如，书信的不同草稿或程序的不同草稿），一个文本文件可能有许多草稿．一旦开发阶段结束，就得到了文本文件．如果文件本文件是一个静态文件，开发便告结束．所得到的源文件版本就成为下一代版本的基础．不能把源文件的档案拷贝与从定期文件备份中得到的文件拷贝相混淆．建立备份是用来防止由于硬件损坏或操作错误所造成的文件丢失．建立档案意在保存每一代源文件的拷贝．

1.4.2 重建先前版本的能力

从先前的版本中重建一个软件产品就是为那个版本恢复源文件．也就是说，作为最终产品，不论它是一个可执行文件还是一张打印好的页，都必须与由原始源文件生成的产品相一致．为了满足这种需求，需要建立一个环境，这个环境包括编译器、预处理器或者文本格式化工具．换句话说，用来建立这一软件产品特别版本的工具及建立的过程与用来建立产品的源文件同样重要．这就意味着软件产品准确的建立过程和顺序，包括各种工具的选择，都必须做记录（存档）．最后，通过改变产品的使用工具或建立过程而不必修改构成产品的源文件，就可得到不同版本的软件产品．

1.4.3 控制后代的能力

让我们从最终编辑器的第 3 版本开始．如果没有另外的计划，版本 3 必须是版本 2 的直接后代，而不是其它版本（如第 1 版本）的后代．这是一个后代控制或者后代源文件控制的简单说明．当源文件管理系统首先需要从任何归档的版本中恢复文件的时候，它并不需要任何有关继承的知识．然而，如果没有每个版本的继承知识，我们就不能控制源文件所发生的任何修改．

控制后代的另外一个方面，就是具有在后代版本生成过程中锁住任何版本的能力．没有这种能力，就不可能从没有生存能力的源文件中生成后代版本．当为常规版本建立分支的时候，这种锁住能力的使用特别重要．

1.4.4 控制存取能力

不论是预防未经许可的修改，还是预防偷窃，限制源文件的存取都是很重要的．这是一个十分棘手的领域并将产生很大的刺激，应该控制住存取的次数．为了这个目的，存取

度并不重要，重要的是我们必须对此进行处理，并提供适当有效的安全等级的建立工具。

1.4.5 编辑检查控制

除了建立新分支外，每个版本都应只有一个后代。防止一个父代偶然生成多个后代也是很必要的，同时知道当前哪个父代在生成新后代也是重要的。这点小小的信息能节省很多到处查找谁做了源文件修改检查的工作。

1.4.6 归档源文件的方便存取

提供源文件方便存取手段的明显理由，就是当开发组等待从归档文件中恢复源文件时要花费很多时间。如果要得到最新的正式版本需花费很长时间的话，使用碰巧可用的任何版本开始工作将是十分有吸引力的。一个开发者使用窃得的版本，将会使他或他的管理者相信这个版本与正式版本是一致的。这种策略并不总是得益的，更多情况是不得益，这是因为你要花很多时间来检查排除生来就有的错误。修改源文件所花费的时间与检查排出在正式版本中早已排除的错误所花费的时间直接成正比。提供方便的源文件存取手段可使由这种错误所造成的损失减至最小。

1.4.7 最小联机存贮需求

这种需求以计算机系统配置提供联机存取源文件库为前提。当计算机大量存贮越来越便宜时，磁盘空间仍然有限。这个问题就变成了怎样才能取得最少的存取并保持最少量的联机存贮需求，其答案就是要在最少的存取时间和磁盘空间开销中找到平衡点。

1.4.8 实现和支持的最小开销

建立源文件控制系统的主要原因就是要避免时间、金钱和精力的损失。正因为这样，源文件控制系统就是为免遭这类损失而购买的安全保险设施。象保险设施一样，不管源文件控制系统是手工系统还是高度自动化，都要花费金钱、时间和努力来建立和维护，但这些花费远比没有控制所需要的花费要少。我们的目标就是要提供一个最低花费的合适保护等级。有时，也包括在期望的目标和要达到目标的花费之间做一笔交易。例如，从根本没有安全的全面存取控制到每次源文件的存取的完整记录，需要多少源文件的安全性。当提供不同等级的文件安全的花费确定下来后，没有足够文件安全性的花费却很难固定。因此，确定一个适宜的安全等级是一件困难的决定。作为一种保险，问题是需要一个多大的保险设施。

1.4.9 用户认可

最终，任何源文件控制系统只有用户接受以后才能工作，满足这个需求不是一件简单

的事情，因为它包括系统的设计、用户的观念（偏爱最少控制的人）和管理的态度．当我们回顾传统的手工控制文件的进展告一段落之后，本书的其余部分我们将转向 SCCS 和怎样建立一个用户友好的源文件控制系统．

1.5 源文件管理的过程

过去的几年里，提出了很多不同的软件开发模型，在每一个模型中，有一些人负责管理源文件．正因为存在很多的管理方法，源文件管理员（source administrator）这一术语将被用来描述那些拥有源文件管理责任的人或组．尽管总是提到源文件管理的需要，但却很少提到源文件管理的实际过程．正向我们看到的那样，与要做什么相比，谁去做是第二位的．为了理解这个过程，先介绍一下传统的手工系统．然后，我们会看到 UNIX 工具是怎样使这一手工系统自动化的．

1.5.1 传统的手工系统

象任何系统一样，手工源文件控制系统依赖于用户，并遵循一组规则．下面的五条规则说明了一个典型的手工系统．

> **规则 1**：所有与产品有关的源文件，不包括多余的文件，一定要交给源文件管理员．

源文件管理员怎样知道她或他收到了正确的文件呢？用来建立产品的源文件都提供了吗？所有的工作文件和另外的无用文件都移走了吗？不幸的是，试图确定哪些文件是真实的哪些是虚假的要浪费很多时间．唯一可靠的方法就是比较给定文件的列表与审定好的文件列表．

> **规则 2**：把从提供的源文件建立产品的完整的指令集提供给源文件管理者．

任何人当他试图重新建立产品的时候，都知道精确而详细的指令是必要的．不仅这些指令指定了生成产品的过程，同时也必须指定建立产品时用到的所有工具和其版本号．

> **规则 3**：软件开发组建立的产品必须交给源文件管理者．

规则 1，2 和 3 构成了这样一个公式：源文件+构造产品的指令＝软件产品．如果源文件是正确的，构造产品的指令是正确的，而且提供的产品也是正确的，那么，源文件管理员就能够建立一个与提供给他（她）的产品一样的产品．如果两个产品相匹配，那么我

们就知道源文件是正确的，用来建立产品的指令也是正确的。但是，我们无法知道虚假的文件是否与源文件一起都交给源文件管理者，除非那些文件名字与那个产品版本的源文件名字核查证实后相矛盾。

> **规则 4**：只有那些有适当权限的人才能请求源文件的拷贝或发送新版本的源文件。

通常情况下，这指的是开发组的领导者。当然，如果得到适当的管理批准也可以例外。

> **规则 5**：开发组的领导者负责确保几个人不能同时修改一个源文件。

这个规则很少针对小的开发组。但是，当多个开发组使用同一组源文件时，两个人在同一个源文件工作的可能性便戏剧性地增加了。例如，为多个机器开发了一个 C 编译器，而每台机器都有自己的开发组。在这种情况下，一个人要想对源文件做修改，就必须首先向组长说明。

有了以上 5 条规则，我们就可最多地模拟处理源文件管理。这个手工系统是怎样处理象 1.4 节中所说明的那样来处理源文件控制系统的需求呢？

1. 源文件既可以维护在单个等级中也可维护在产品等级中。当一个产品只有一个顾客时，单个级的源文件管理是最常用的方法，但是不能控制整个产品。事实上，重复建立整个产品的过程可能是很困难的（不同的模块可能是用同一工具的不同版本建立的）。当单个源文件的历史包含于产品的历史中时，整个产品的源文件通常就是源文件管理的目标而不只是单个源文件的目标。但是，如果不对单个源文件进行控制，就可能导致正确的版本会被不正确的版本所替代。因此，源文件必须在产品级和单个级中控制。
2. 假设所用工具的所有版本都保留并且详细的构造指令确实存在，就可能在任何时候重建软件产品。当手工过程足够能建立产品时，那是很浪费时间并易发生错误。尽管讨论使用 UNIX 工具来建立产品不是本书的目的，但 makefile 文件（make 命令的规则文件）和其他用于控制产品生成过程的文本文件都是版本控制的主要因素。
3. 用手工源文件控制系统控制后代是十分困难的。即使源文件管理者对发送给他的文件保有最好的记录，也不能保证第 3 版本产品将是以第 2 版本为基础的，唯一的保证就是源文件的最新版本是用来建立当前的产品的。当建立分支的时候，真正的问题便开始了。正常的趋势就是，随着分支数的增加，每个分支都独立存在。如果所有分支都真正独立，也就没有问题。但是与所有分支有关的修改发生后该怎样呢？因为没有一个共同的基础，所有分支都必须做相同的修改。由于缺乏对后代的控制，导致软件信息混乱，并因而提高了管理维护费用。

4. 任何系统的安全极限都与其最弱的环节相关联，一个源文件管理者所能做的是控制谁取出了源文件（可读）和谁送进了修改后的文件（可写）。一旦超出他（她）的工作范围，管理者就不在控制源文件会怎么样。从这一点上说，存取控制掌握在接受者手中。潜在的接受者数目越大，就意味着存取控制问题越大。处理安全问题的正常办法就是从源文件管理者手中控制节省存取，其结果就是从前面的接受者得到了非正规的版本。因此，在源文件控制系统这一级中的存取控制是必不可少的，但这也仅仅是解决源文件安全问题的答案中的一部分。
5. 同时，从表面上看，让组长负责管理谁修改了哪些文件听起来是可行的，但这确实有些不足。首先，这意味着组长不得不有一套系统来记录谁在做什么。即使把这份工作委托给组里的另外的人，也会由于人为的错误而导致失败。其次，当几个组需要同一个源文件时，修改协调将变成一场可怕的恶梦。
6. 也有几个例外，当进行源文件控制时，开发者的倾向总是采取阻力最小的路线。当存取正规版本很困难时，最容易得到的源文件版本就会变成用来修改，特别是被认为与正规版本相匹配的正规版本。要记住，源文件的控制对管理项目的人来说比修改者关系更重大。
7. 是否要进行联机存贮决定于源文件管理者怎样得到那些源文件。如果它们存贮在磁带上，软盘里或其它媒介上，那么磁盘空间的保护将不成问题。花费的代价是想使用某一特别版本的人要求允许得到文件所需要的时间。如果每个版本的源文件都存放在硬盘上，那么源文件的有效性在磁盘空间保护的花销上将得到改进。
8. 因为手工系统是劳动密集型的，控制的花销直接与控制程度相关联。从某一点上讲，源文件的控制代价将超过发现任何潜在损失的代价。我们使用发现代价（perceived cost）这一术语的原因是当发生事故的时候，发现的代价最大。随着时间的推移，由事故引起的感情上的能量消散了，于是事故造成的可见损失就相应减少了。为使源文件控制系统生存下去，我们只得把它作为开发周期的主要部分并且值得花费。
9. 正如上面所说的那样，取得用户的认可才是源文件控制系统成功的重要关键。测量认可程度的一种方法就是看使人们认可文件控制过程所花的时间。存取控制过程就在后面或附近吗？用的是形成产品的源文件的正确版本或是最容易得到的版本吗？能否用给出的指令形成产品？用户怎样观察源文件控制系统？它是用户工作的障碍还是使他们的工作更容易？更多的情况是，手工系统被认为是一个障碍。

1.5.2 源文件管理自动化

软件开发者总是最不情愿用软件工具解决他们的管理问题。这并不是非那样做不可。本书的其余部分将讨论源文件管理问题的一些方法。为完成这项工作，让我们好好看一下SCCS是怎样工作的，一些附属的工作将SCCS从单个模块的控制移到系统的管理，然后我们讨论怎样完成源文件控制系统。作为这个讨论的一部分，我们还将再回到谁完成什么工作的问题上。

第二章　SCCS 简介

2.1　什么是 SCCS

SCCS（源代码控制系统）这一名词是专门用来控制和说明源文件修改的一组 UNIX 程序．SCCS 的研制直到 1972 年才开始．原始的版本是在 IBM 370 计算机上用 SNOBOL 语言编写，用 SPITBOL 编译器编译的．1973 年，在 PDP 11／45 计算机上研制出了 UNIX 的第一版本．直到 UNIX 系统Ⅴ问世之前）SCCS 一直是程序员工作台的一部分．SCCS 现在被作为 UNIX 系统Ⅴ的一部分．基于 UNIX 第 7 版本中的 SCCS，延着它自己的道路开发了 Berkeley 版本（BSD SCCS）．其结果是在概念上这个版本与系统Ⅴ SCCS 一样，但命令和语法上有些变化．

SCCS 是由一组程序构成的系统，这些程序包括用来生成和管理 SCCS 文件，从 SCCS 文件中取回源文件和把源文件存入 SCCS 文件的基本实用程序．此外，SCCS 还提供了另外的工具程序用来帮助管理 SCCS 文件．表 2.1 给出了与 UNIX 系统Ⅴ一样的完整 SCCS 工具清单．第四章到第九章提供了这些工具的使用信息．

当历史和名词趋于强调源代码文件时，SCCS 就已经用于很多种源文件了，包括文献．可能更有吸引力的名词曾是版本控制系统（Version Control System）．这个名字的意思以及 SCCS 所做的工作就是文本文件的版本管理．

表 2.1　SCCS 命令

工具名称	说明
admin	生成和管理 SCCS 文件
cdc	改变注释并装入 SCCS 文件中
Comp	将几个 deltas 组合成一个 delta
delta	用源文件更新 SCCS 文件
get	从 SCCS 文件中取出源文件
help	UNIX 帮助系统的一部分，提供有关 SCCS 命令使用的帮助
prs	打印 SCCS 文件控制记录和信息
rmdel	从 SCCS 文件中移走一个 delta
sact	打印当前 SCCS 文件编辑活动
sccsdiff	比较和打印 SCCS 文件中两个 delta 之间的差别
unget	通过编辑选项取消先前的 get
val	通过一组指定变量使 SCCS 文件生效
vc	提供文本中关键词替代的版本控制系统
what	查找源文件或其产品中的 SCCS ID 信息

2.2 SCCS 的哪个版本

本书中提供的信息基于 UNIX 系统Ⅴ，版本 3 的 SCCS．根据 SCCS／PWB 用户手册，对于那些使用 UNIX 第 7 版并安装了 PWB 的读者，本书中给出的大多数信息都是可用的．对于那些使用 UNIX 系统Ⅲ的读者，使用的命令与系统Ⅴ中的 SCCS 完全一样，仅有一点主要差别就是 UNIX 系统Ⅲ中没有 vc 命令．为集中精力于系统Ⅴ SCCS 上，系统Ⅴ与 BSD SCCS 之间的差别这里不讲．

从标准方面讲，本书亦遵循 X／OPEN 移植指南，但 X／OPEN 文件不包括 cdc，comp，sccsdiff 和 vc 命令．

2.3 选择 SCCS

版本控制系统已经开发出一段时间了，而在 UNIX 世界中的选择却是有限的．美国普度大学的 Walter F．Tichy 开发的 RCS（校订控制系统）是一个很有意义的选择．到写这本书的时候，RCS 已得到荣誉上的奖励了．其它的版本控制系统虽然已经为 UNIX 开发出来，但都没有象 SCCS 和 RCS 那样流行．

尽管 SCCS 和 RCS 存在明显的差别，但本书中所确定的四个源文件管理原理都适用于两者，因此我们可以集中于源文件管理原理．SCCS 是本书其余部分中使用的唯一源文件控制系统．

2.4 命令约定

在快速流览 SCCS 以前，我们先对 SCCS 的命令约定作个短暂的回顾．在讨论每个 SCCS 命令的时候，对于那些例外的约定将给出注释．一个 UNIX 命令的基本格式是：

$ 命令[-X {可选变量}]{哑元}

这里的-X 指一个可选项，有时也叫做一个标志（flag）．可选变量和哑元构成了修改 UNIX 命令动作的两个手段．大多数的 UNIX 命令在没有指定选项和变量情况下都可执行．因此，知道每个命令的隐含运行是很重要的．

可选项总是具有这样的格式：减号（-）后面跟一个叫关键字母（keyletter）的字母或数字．有时可选项也需要在关键字母后面跟一变量．关键字母说明可选项的本质；如 adimin 命令的-n 选项说明要生成一个新的 SCCS 文件．如果可选项需要一给定的变量，这个变量必须立即跟在关键字母后并且不能包含任何空格或制表符．如果变量包含空格或制表符，这个变量就必须用双引号" "括上，例如：