21世纪计算机编程指南系列

北京希望电脑公司 总策划
唐靖飚 周良源等 编 著

UNIX平台下C语言高级编程指南

宇航出版社
北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

21世纪计算机编程指南系列

北京希望电脑公司 总策划
唐靖飚 周良源等 编 著

UNIX平台下C语言高级编程指南

学报出版社
北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

图书在版编目（CIP）数据

UNIX 平台下 C 语言高级编程指南 / 唐靖飚等编著.－
北京：宇航出版社，2000.5
（21 世纪计算机编程指南系列）
ISBN 7－80144－171－0

I. U… II. 唐… III. C 语言－程序设计－指南
IV. TP312－62

中国版本图书馆 CIP 数据核字（2000）第 06765 号

宇航出版社出版发行
北京市和平里滨河路 1 号（100013）
发行部地址：北京阜成路 8 号（100830）
零售书店（北京宇航文苑）地址：北京海淀大街 31 号（100080）
北京媛明印刷厂印刷
新华书店经销

2000 年 6 月第 1 版　　2000 年 6 月第 1 次印刷
开本：787×1092　1 / 16　　印张：28.875　字数：524 千字
印数：1-5000 册　　定价：45.00 元

前　言

UNIX 操作系统发展至今已经有 35 年了，随着越来越多的计算机厂家提供对 UNIX 系统的支持，目前得到了广泛的应用。UNIX 不仅以它的功能强、稳定性高和开放性深受广大程序员的喜爱，而且随着它的发展，越来越多的新技术已经融入其中，使得 UNIX 不仅功能越来越强，而且与用户的友好性也有了长足的进步。

以前，因为对系统速度的要求，使得只有在工作站上以及更高性能的计算机上才提供 UNIX 系统，使得 UNIX 系统总给人一种高不可攀的感觉。随着 PC 机性能的不断提高，现在已经有了多种 UNIX 操作系统运行其上。例如 SCO 的 SCO UNIX、SUN 的 Solaris x86 以及现在流行的免费的 Linux 和 FreeBSD。在我国 PC 机普及范围比较广的情况下，这使得许多程序员可以用廉价的微机使用 UNIX 操作系统。尤其这几年随着网络在我国普及，越来越多的地方需要提供稳定快速的网络服务，UNIX 以其优秀的稳定性、对网络良好的支持和众多的网络产品，成为许多 ISP 的首选操作系统。

UNIX 操作系统是 Ken Thompson 和 Dennis Ritchie 于 1974 在《ACM 通讯》中发表的一篇文章中首次提出的。他们不仅对 UNIX 做了一个实现，而且公开了操作系统的内核。从那时起，UNIX 系统得到迅速的传播并在工业中得到了广泛的应用。不少计算机厂家都推出了各自的 UNIX 系统，但由于没有标准的缘故，使得 UNIX 发展成为了一个比较混乱的家族，它们在某些方面类似，但又有不兼容的地方。到了 80 年代中期，主要的两个组织 UI 和 OSF 开始着手对 UNIX 进行标准化，但这两个组织各自为政，推出了 OSF/1 和 SVr4 两个标准和对应的系统。这期间 IEEE、ISO 提出 POSIX（Portable OS based on unIX）报告，目前有不少系统采用了 POSIX 的标准。现在 UNIX 家族中常见的系统有 SVr4、BSD、SCO UNIX、SunOS、solaris、IRIX、OSF/1、DG-UX、HP-UX、AIX、Ultrix、UnixWare、NetBSD、FreeBSD、Linux 等。它们大部分符合 SYSV、BSD 或 POSIX 标准，或者融合了 SYSV、BSD、POSIX 的一些部分。

本书主要分为四个部分介绍了 UNIX 系统下编写 C 语言应用程序的知识。因此希望读者对 C 语言有一定的了解。第一个部分主要是介绍基本的系统调用。了解它不仅是作为 UNIX 系统的概念基础，而且在 UNIX 编程中会经常用到其中一些系统调用，只有了解它们的运作，才有利于写出更好的程序。其中包括 UNIX 文件系统的介绍及其系统调用。终端操作一章介绍了 UNIX 下终端的概念，并介绍了对终端设置的系统调用。UNIX 是多任务的操作系统，多进程是它的特色之一，所以介绍了有关进程及其进程间通讯的系统调用。目前 UNIX 系统大部分对线程有所支持，因为不是十分规范，这里我们只介绍了 POSIX 线程，但由于目前用线程做并行计算的用户不是很多，所以将它放到了网络编程部分，并结合网络编程的知识，举了线程使用的例子。信号也是 UNIX 的特色之一，本书用了一章的篇幅介绍了有关使用信号编程的知识。

第二部分是对网络编程的介绍。随着 Internet 的应用越来越广泛，TCP/IP 也越来越被大家熟知，这里以 Berkeley 的 Socket 为主，介绍了基于 TCP/IP 的程序编写。还用了一章的篇幅介绍了 CGI 程序的编写并在其中简要介绍了 Perl 语言的知识。

第三部分是基于 Motif 的 X Window 应用程序的开发，现在对应用程序不仅需要它功能强大，还提出了易用性和用户友好性的要求。这一部分，主要是介绍如何使用 Motif 开发 X Window 下的应用程序。其中是以 Motif 的 widget 为基础，如果读者 Gadget 有兴趣，可以参考联机手册(man)，它们在使用上类似。

第四部分主要是一些常用的编程工具的介绍，这些工具包括 cc 编译器、调试器 dbx 和 gdb、make 和 makefile 的规则以及版本控制工具的介绍。其中版本控制一章由周良源编写。

本书由唐靖飚、周良源组织编写，参加本书编写、校对、录排等工作的人员有：唐靖飚、周良源、陈曙辉、孙志刚、刘亚杰、李志、孙大庆、赵军、龚波、张巧莉、田丽韫、陈大晖、邓波、邓涛、李林、李卓林、聂宛析、田敏、金玉露、王小光、龚露娜、马军、马丽、田军、田洗县、王小将、高翔、丁建华、崔羽、王大军、李节、蒋华、郭祥雷、孙庆、周国庆、安洁、张志超、杨垒、叶涛、董金云等。本书在写作的过程中得到了陈曙辉、孙志刚、刘亚杰等同志的热心帮助，特别是陈曙辉从开始到结尾为本书做了大量的工作，在此表示衷心的感谢。

十分希望这本书能对读者带来益处，由于作者水平有限，如有错误之处，敬请读者原谅。也希望将您的意见告诉我们。

1999 年 11 月 • 长沙

目　录

第一部分　基本的系统调用

第二部分　网络编程

第三部分　X Window 应用程序开发

第四部分 常用的编程工具

第一部分　基本的系统调用

第 1 章　文件子系统

UNIX 核心的两个主要组成部分是文件子系统与进程子系统。文件子系统控制用户文件数据的存取与检索。在 UNIX 中，设备也是作为一种特殊的文件处理的。关于终端的编程将在第 2 章中介绍。UNIX 的文件子系统管理文件空间的分配，管理文件系统的空闲空间。进程通过系统调用来对文件进行操作，也可以通过 C 语言的函数调用来操作文件，而 C 的函数库使用系统调用来实现这些函数调用。本章先阐述文件系统的基本概念，再分两个小节分别介绍基本的调用和复杂的调用。读者应先熟悉文件系统的基本概念，这对后续内容的理解很重要。

1.1　文件子系统的基本概念

本节主要介绍 UNIX 文件子系统构成的和文件子系统的基本概念。为了更好的理解这一操作系统的文件子系统，我们先要了解如何分配与回收文件空间、如何组织文件、如何管理空闲空间。尤其对 UNIX 系统来说，设备是以特殊文件的方式来管理的。这里我们要区别一下文件子系统和文件系统，文件子系统(file subsystem)是指从体系结构角度上划分的核心结构的一部分。而文件系统(file system)是指文件存在的物理空间，它管理用户对数据和设备的读写访问，并保证信息的安全性和私有性。

UNIX 文件子系统使用缓冲机制存取文件，缓冲机制用来调节核心和二级存储设备之间的数据流。核心使用缓冲与块设备驱动程序交换数据，这样可以使得块设备速度慢而核心快的矛盾得以缓和。

UNIX 把正规文件和目录文件保存在磁盘或磁带这样的块设备上。在无盘工作站上，文件被存放在一个远程系统上，并通过网络进行存取，本书不讨论无盘工作站的情况。一个系统一般可以有若干物理磁盘，每个磁盘可以包含一个或多个文件系统。把一个磁盘分成多个文件系统是为了管理员易于管理文件。核心在逻辑上只涉及文件系统，把每个文件系统都当作一个逻辑设备，并给每个逻辑设备一个逻辑设备号来标识。一个文件系统是由一系列块(block)构成，每个块的大小一般在生成这个文件系统时指定，或依赖于系统的实现。一个文件系统一旦生成，其中的块的大小是固定的。一个文件系统包含这样一些结构：引导块(boot block)、超级块(super block)、索引节点表(inode list)、数据块(data blocks)。引导块包含该文件系统的引导程序；超级块包含空闲索引节点表和空闲数据块表；索引节点表用来存储文件相关信息及存储位置；数据块是磁盘上存放数据的磁盘块。

1.1.1　索引节点

索引节点(index node)在 UNIX 书籍中出现十分频繁，常被简称为 inode。它记录一个文件的存储位置，并包含存取权限、文件所有者及存取时间等信息。索引节点存储在磁盘上，核心把 inode 读进内存索引节点来操作它和它所对应的文件。这里区分一下概念，为了便于理解，现在把存储于磁盘上的 inode 称作磁盘索引节点，而把它在内存中的映像称作内存索引节点。实际上内存索引节点的结构要比磁盘索引节点的结构多几项，后面会提到它们的结构。磁盘索引节点由如下字段构成。

- 文件所有者标识：这个标识指出该文件的所有者 id 和所属的组 id。
- 文件存取许可权：系统按三个类别对文件实施保护：文件的所有者(owner)、文件的所属组(group)、其他用户(other)。每类都有读、写、执行三种权限，可分别设置。目录文件的执行权限是指搜索该目录的权力。
- 文件类型：文件可以是正规文件、目录文件、字符设备文件和块设备文件及管道文件。
- 文件的存取时间：共有三个有关时间的标识，分别是文件最后一次修改的时间、最后一次被存取的时间、最后一次索引节点修改的时间。
- 文件链接数：记录了引用该文件的目录表项数。即表示有多少个文件名指向该文件。
- 文件数据的磁盘地址明细表：指出含有文件数据的磁盘地址位置。
- 文件长度。

这里要注意改变文件内容与改变文件索引节点之间的不同，写文件才改变一个文件的内容，而改变文件所有者、改变文件存取许可权等操作都会改变文件的索引节点内容。写文件必然改变文件的索引节点，而改变文件索引节点并不意味着改变文件的内容。

UNIX 为了使文件操作速度较快，在这里又使用了缓冲的概念。系统将磁盘上的索引节点读入内存，不仅加快文件操作的速度，而且便于实现文件操作的同步与互斥。内存索引节点不仅将磁盘索引节点的内容复制到内存，而且还增加了以下字段。

- 内存索引节点的状态：该状态标识指示。索引节点是否被上锁、是否有进程等待该索引节点变为开锁状态、内存索引节点的数据是否已被更改与磁盘索引节点的数据不同、索引节点所指向的文件数据在内存中的映像是否已更改、是否该文件为安装点。
- 该文件所在文件系统的逻辑设备号。
- 索引节点号：磁盘索引节点是按线性顺序存放在磁盘上的，索引节点号是用来指出内存索引节点对应磁盘索引节点在磁盘上的位置。
- 指向其他索引节点的指针：这个指针是为了将一系列索引节点构造成队列，以方便查找。一般分成两个队列，一个是使用中的 hash 队列，一个是空闲队列，这样使用缓冲的目的是为了可以减少读盘次数，提高内核效率。
- 引用计数：一个文件在内存中最多只有一个内存索引节点，但一个系统中可以有该

文件的多个活跃实例。该字段指示该文件的活跃实例数目。

UNIX 系统在打开一个文件时，先在内存索引节点的 hash 队列中查找，看索引节点是否已在队列中。若查不到这个索引节点，则从空闲队列中分配一个索引节点，并上锁。然后再将磁盘索引节点的内容读入到这个已上锁的内存索引节点。如果有其他进程需要使用这个内存索引节点，则置内存索引节点标志，指出有进程等待该索引节点，当这个索引节点被解锁时会唤醒等待该索引节点变为空闲的所有进程。当索引节点使用完毕，如果没有其他进程等待这个索引节点，核心要释放一个索引节点时，将它的引用计数减一。如果该计数值为 0，且它与磁盘索引节点不同时，核心要向磁盘写该索引节，并将索引节点放入空闲队列。如果又一次打开该文件则可以从空闲队列直接将索引节点放入 hash 队列，而不需要读盘，这样使得核心非常高效。如果空闲队列的索引节点已被分配给其他磁盘索引节点，就只能再分配另一索引节点，并重新读入索引节点值。如果索引节点中该文件的链接数为 0（即在磁盘上没有其他文件与该文件相连），则核心可以释放与该文件有关的所有磁盘数据块，并且释放该磁盘索引节点。

1.1.2　正规文件的结构

上节提到，索引节点包含文件数据的磁盘地址明细表，它指出文件数据在磁盘上的存储位置。系统在磁盘上给文件分配空间时，并不一定需要连续的空间来存放文件。核心有较大的灵活性，一次分配一块文件空间，并且允许文件数据分布在整个文件系统中，这样旧消除了造成碎片的风险，每个文件浪费的空间最多不到一个数据块。但是这样也导致了文件数据读取时，数据定位的复杂程度。另外，一个索引节点的大小是固定的，也就是说，索引节点包含文件数据的磁盘地址明细表的大小是固定的，当一个文件较大时，索引节点中的磁盘地址明细表就不够放。有几种解决办法，一种是加大 inode 的大小，但这样浪费磁盘空间；另一种是限制文件大小，但这种方法不可取。

第三种是 UNIX 的解决办法，在 inode 的磁盘地址明细表中前若干项是直接指向磁盘数据所在块的块号，在不同的文件系统中明细表中存放的直接块号的（磁盘块号）数目不一定相同，例如在 SYSV 的文件系统中是前 10 项，在 minix 文件系统中是前 7 项，在 second extended（ext2）文件系统中是前 12 项为直接块号。随后 3 项分别是一次间接、二次间接和三次间接。一次间接是指该项存放的磁盘块号所指向的磁盘数据块的内容是若干数据块的磁盘块号，即从该项得到数据块号，由此可得数据块的内容，而数据块的内容是一张磁盘块号表，每一项地址指向一个数据块。二次间接是指该项存放的磁盘块号所指向的数据块的内容是一张磁盘块号表，而由该磁盘块号表得到的数据块的内容仍然是一张磁盘块号表，再由该表得到的数据块才是文件的数据。三次间接以此类推。该结构可由图 1-1 说明（以 SYSV 为例）。虽然从逻辑上还可以做四次间接，但实际上不需要这样，现在的结构已经足够，它可以使文件的最大长度达到(10+1×256+1256×256+1×256×256×256) ×1kB=16843018kB（当一个块的大小为 1024Bytes、磁盘块号为 32bits 时），而在索引节点中文件长度是由一个 32bits 的

字段存放的，所以文件最大长度不能超过 4GB。

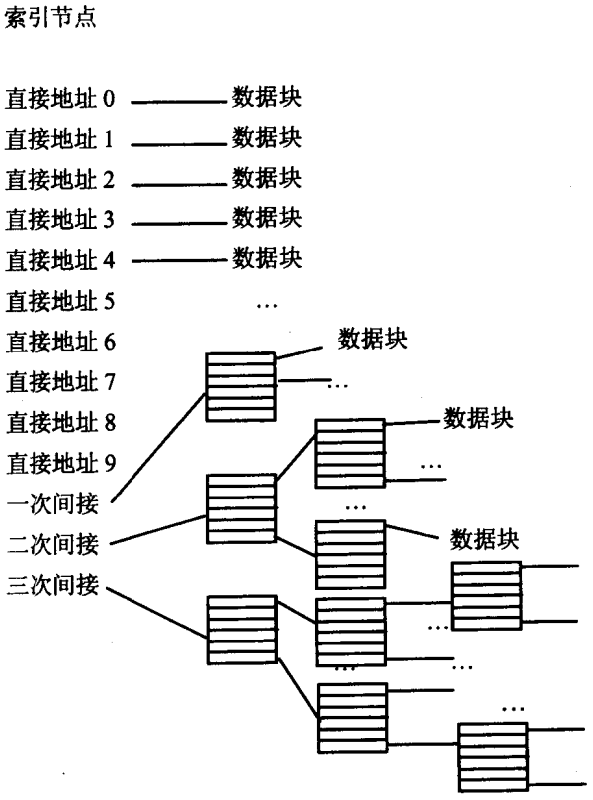

图 1-1　索引节点的数据块地址存放

UNIX 的这种分配方式虽然将数据定位的任务复杂化了，但有效地利用了磁盘空间，解决了磁盘碎片的问题。假设文件系统中一个块（block）的大小为 1024Bytes，那么存放一个大小不超过 10kB 的文件可以由直接块号表得到数据块的块号，数据定位齐不复杂。而对一个大小超过 10kB 的文件来说，前 10kB 可以从直接块号表得到磁盘块号，但后面的数据就要先读入一个一次间接块的内容，再去读数据，对磁盘操作稍有影响。如果文件长度大于 65802kB，则读取偏移量大于 65802kB 的数据就要多读入三个磁盘块，影响了效率。那么核心的效率到底如何，这取决于使用 UNIX 的用户群是经常存取大文件，还是更经常地存取小文件。经过 Mallender 和 Tannenbaum 对 19978 个文件取样并进行分析，发现其中 85%的文件都小于 8kB，48%的文件小于 1kB，这说明大部分文件可以通过一次磁盘存取而得到，进一步说明核心使用这种存放文件的方式是可行、有较的。

1.1.3　目录

目录是一种文件，起到从文件名到索引节点号之间进行转换的桥梁作用，也是使得文件系统成为树状结构的关键。目录文件的内容是由一系列的目录登记项构成，每项由该目录

包含的一个文件名及该文件的索引节点号两部分组成。目录登记项一般含有以下几个字段。

- 文件名：在 SYSV 的一些系统中，文件名的长度为 14 个字符，并固定为 14 个字符。在 second extended(ext2)文件系统中，文件名的最大长度为 255 个字符。现在很多系统都支持长名文件，文件名长度一般都可以达到 255 个字符。
- 索引节点号：该字段指出本目录中该文件名所对应的索引节点号。
- 文件名长度：在 SYSV 的文件系统中，没有该字段。这个字段是 ext2 等文件系统中用来支持长文件名，指出文件名的长度（<256），这样可以节约目录文件的存储空间。
- 该目录登记项长度：这个字段也只在 ext2 文件系统中出现，指出该目录登记项的长度。

每个目录文件的前两项是两个特殊文件“.”和“..”，其中“.”表示当前目录自身，对应“.”文件的索引节点号就是该目录文件的索引节点号。文件“..”表示上级目录，即对应“..”的索引节点号是当前目录的父目录的索引节点号。“/”是一个比较特殊的目录，称为根目录。该目录是在 mkfs 程序生成文件系统时产生的目录，该目录文件的“.”和“..”对应的索引节点号是“/”目录自身的索引节点号。

在一个文件操作中我们常用路径名来指出文件所在的位置，路径名第一个字符为“/”时，我们称这个路径是绝对路径，即该路径从根目录开始指出文件的位置。路径名以“/”为分隔符，区分各目录层次。最后一个分量可以是目录，也可以是文件名。当路径名不是以“/”为第一个字符时，我们称这个路径是相对路径，即该路径从当前目录指出文件所在的位置。有关当前目录、绝对路径、相对路径概念的详细介绍请参考有关 UNIX 系统管理的书籍。

核心对目录文件数据存储的操作与普通文件一样，也使用 inode 来保存目录文件的信息及存储的磁盘块号。目录文件的读权限表示进程可以读取这个目录，写权限表示进程可以创建或撤销目录登记项（通过 creat、mknod 等），执行权限表示进程可以搜索这个目录。核心在目录树中找一个文件的索引节点的过程，是一个转换过程。核心内部对文件都是用索引节点来操作的，这就是说必须有一个路径名到索引节点的转换过程。对于一个绝对路径来说，就是从根目录起，在根目录文件中寻找路径名的下一个分量所对应的索引节点，再从索引节点找到磁盘数据块，如此循环直到搜索完毕为止。对于一个相对路径，就是在当前目录文件中寻找下一个分量所对应的索引节点。对于路径名中跨越安装点的问题将在 1.3.11 节中讨论。

举例来说，要打开/etc/passwd 文件，核心先读取根目录文件在该文件中找到目录登记项中文件名是 etc 的项，并由此得到 etc 的索引节点号，由索引节点找到 etc 目录文件。在读入 etc 目录文件后，查找文件名为 passwd 的目录登记项，并得到索引节点，这时可以打开该文件了。对于一个相对路径来说，要打开../../etc/passwd 文件，首先根据当前工作目录，得到该目录文件，并可以查找其父目录的索引节点号，重复上述查找过程，即可得到该文件的索

引节点。最后要注意的是，一个目录文件的目录登记项中，如果文件名存在，而索引节点号为 0，表示该文件在这个目录存在过，现在不存在了（已经被删除了）。

1.1.4 超级块

超级块(Super Block)是用来描述一个文件系统的状态，如：文件系统的大小、空闲空间位置等信息。超级块由如下字段构成：

- 文件系统的规模，如 inode 数目、数据块数目、保留块数目、块的大小等
- 文件系统中的空闲块的数目。
- 文件系统上部分可用的空闲块表。
- 空闲块表中下一个空闲块号。
- 索引节点表的大小。
- 文件系统中空闲索引节点数目。
- 文件系统中的部分空闲索引节点表。
- 空闲索引节点表中下一个空闲索引节点号。
- 超级块的锁字段。
- 空闲块表的锁字段和空闲索引节点的锁字段。
- 超级块是否被修改的标志。
- 其他字段。各种系统为了维护本身文件系统的可靠，采用了不同的实现。如 ext2 增加了安装次数、最大安装次数、安装时间、写时间、最后一次检查（check）的时间、inode 结构的大小、文件状态（是否只读、上次关机是否拆卸）等。

由以上字段可以看出，超级块对于空闲inode和空闲块的管理，便于文件系统存取和管理文件。超级块中的锁字段是为了保证互斥操作，这将在下一小节中详细介绍。在SYSV中和ext2中都有一字段标出该文件系统是否完整。例如，UNIX要求关机时要先将缓冲数据写回文件系统，并拆卸(umount)该文件系统，如果没有拆卸文件系统就关机，很可能导致数据丢失，所以在安装一个文件系统时核心会检查超级块中字段，如果上次没有做拆卸就要对文件系统进行检查(fsck)。ext2文件系统还对安装次数有规定，一旦安装次数超过超级块中最大安装次数，就要做文件系统检查fsck，并重新记录安装次数。

1.1.5 索引节点与磁盘块的分配与释放

文件系统包含一个索引节点表，如果在分配索引节点时让核心去搜索整个索引节点表，效率太低了。这时，为了判断每个索引节点是否空闲都要读一次，而且很可能是从磁盘读入，所以核心采用另一种方法。将部分空闲的 inode 构成部分空闲 inode 表，放入超级块中。每次分配 inode 时，如果超级块中的空闲索引节点表不空，就从该空闲索引节点表中取一个分配，读出该磁盘索引节点，并给该索引节点上锁。如果超级块中的空闲索引节点表为空，则从铭记节点开始向后（从低序号向高序号）查找空闲索引节点，并把它们登记到超级块的空闲索引节点表中，直到超级块的空闲节点表满为止，并设置铭记节点为搜索到的最后一个空

闲节点号。铭记节点是用来记录下一次搜索空闲索引节点的起始位置，铭记节点前的索引节点都是已经分配了的，所以每次搜索只要从铭记节点开始即可。在释放索引节点时，要看该索引节点号是否大于铭记节点号，大于则直接释放该索引节点，小于则要在释放后将铭记节点号改成刚释放的索引节点号。

上述情况是一种简化的分配方案，还没有考虑到竞争的情况。下面考虑一下互斥的简单情况。在该算法入口处，首先要看超级块是否上锁，如果上锁则等待超级块变为空闲；如果没有上锁或超级块已经变为空闲，则看空闲索引节点表是否为空。如果为空，则要将超级块上锁，取铭记节点，从铭记节点开始搜索磁盘，并将空闲索引节点登记到超级块的索引节点表中，直到超级块满或再也找不到空闲索引节点为止，最后为超级块解锁（这里核心将唤醒等待超级块变为空闲的那些进程），并记录下铭记节点。如果空闲索引节点表不为空，则看空闲索引节点表是否上锁，若上锁则等待解锁；否则给空闲索引节点表上锁，分配索引节点，再将空闲空闲节点表解锁（关于算法，可以以 linux 核心源代码中/linux/fs/ext2/ialloc.c 作为学习的参考）。

在不同的文件系统实现中有不同的安排索引节点表的方法。例如，在 ext2 中文件系统划分成了若干组(group)，每组都含有一个索引节点表的位图块（bitmap block）和一个数据块表的位图块以及若干数据块。这里举 ext2 的例子是为了说明，系统的基本原理是相同的，但实现的方法有所不同。下面我们再来看看数据块的分配。

当进程写文件时，核心要从文件系统分配磁盘块，用来作为文件的数据块。现在的磁盘容量都非常大，如果直接在磁盘中寻找空闲的数据块是费时低效的。如何组织磁盘块，就成了各文件系统的不同实现方法。其基本原理是，在生成文件系统时(mkfs)，将数据块组织成链表。超级块中空闲块表记录的是整个文件系统空闲块的一部分，其中最后一个块号所指向的磁盘块的内容又是一个空闲块表，以此而组织成了空闲磁盘块号链表。当核心要分配一个磁盘块时，首先看超级块中的空闲磁盘块表，如果不是表中最后一块，则分配该块，空闲块总数减 1，标记超级块被修改过；如果是表中最后一块，则将超级块上锁，读入该磁盘块，该块的内容是一个空闲块表，将该表拷贝到超级块的空闲块表，并将超级块解锁，最后将该磁盘块分配。这里要注意空闲磁盘块链表是在生成文件系统时，由 mkfs 程序构造的。

释放磁盘块的过程正好相反。首先，如果超级块中的空闲磁盘块表不满，则将该磁盘块地址放入超级块的表中；如果超级块的空闲磁盘块表满了，则将超级块的空闲磁盘块表拷贝到该磁盘块（要释放的磁盘块）中，并把该块写回磁盘，将该块号写入超级块的空闲磁盘块表，成为表中唯一的成员。

至此，我们介绍了 UNIX 文件系统的基本概念和组织方法，希望能对读者从总体上了解 UNIX 的文件系统结构和阅读后续章节有所帮助。如果读者对核心实现有兴趣的话，可以看看 Linux 的核心源代码，因为这个源代码是有 GPL 的，到处都可以得到，并且它符合 POSIX 标准，比 minix 功能强大。

1.2 基本的文件输入和输出

本节将详细讲述普通文件的基本输入和输出操作，包括建立文件（create）、打开文件（open）、文件的写操作（write）、文件的读操作（read）、关闭一个打开的文件（close）、设置文件读/写指针位置（lseek）等。介绍它们的目的，是为了让读者了解一下核心的一些基本的系统调用。UNIX 上的 C 语言库函数中的对文件操作的函数如 fopen、fscanf、fprintf、fclose 等，都是以系统调用为基础实现的。系统调用本身看起来就象 C 语言中普通的函数调用，而库把这些函数调用映射成进入操作系统的原语。系统调用是进程与核心的接口。在第 3 章中我们将介绍进程的状态，这里为了讲述一下系统调用和函数调用的区别，我们简单提一下进程的状态。我们知道，进程一般是在用户态下执行的，一直到进程需要核心为其服务，这时通过一个系统调用，就转入核心态运行，直到服务完毕，才从核心态退回到用户态。而函数调用无法改变进程的运行状态。C 语言函数库中有许多函数都使用系统调用为其服务。关于系统调用实现的原理，将在第 3 章进程的基本概念中再讲述。

本节讲述的基本文件操作是为了以后了解各种系统调用而做的准备工作，了解这些系统调用为以后做自己的库函数也奠定了一定的基础。下面在讲述基本文件操作的系统调用之前，我们先来看看文件描述符。

1.2.1 文件描述符

UNIX 中每个进程可以拥有若干文件描述符（file descriptor），数量多少依赖系统的实现(Linux 中一个进程可以有 256 个文件描述符，在内核源代码中是由 NR_OPEN 定义的)，编号从 0 到 NR_OPEN，也就是表项的索引值。每个进程有自己的用户文件描述符表（我们常简称它为文件描述符表或 fd 表）。文件描述符表的前三项，对于一般进程来说是一些特殊的文件。文件描述符 0 是标准输入文件，对于一般进程来说是键盘；文件描述符 1 是标准输出文件，一般是输出到显示器；文件描述符 2 是标准错误输出文件，一般也是输出到屏幕。这三个标准文件描述符都可以打开给一个文件，这对于实现一些复杂程序很重要，如做一个 shell 等。如果遵守这三个描述符的习惯，会使程序更容易通过管道通信。

我们在用 open 等系统调用时，该调用将返回一个 0～255 之间的整数作为文件描述符，该数字是这个文件在该进程的 fd 表中表项的索引值。用户文件描述符表中表项的内容是一个指针，它指向全局文件表（又称作核心文件表）的一个表项。该文件表的表项内容有：该表项的引用数、一个指向该文件在索引节点表中的索引节点号、一个记录读/写指针位置的域。注意文件表是一个全局表，有关 fd 表、文件表、索引节点表之间的关系，下面介绍 open 时，将详细讲述。