UNIX

高级编程技术

UNIX

周祖伦　尤晋元
王　舟　王保定　朱建忠　编著

上海科学技术文献出版社

UNIX高级编程技术

周祖伦　尤晋元
王　舟　王保定　朱建忠　编　著

上海科学技术文献出版社

(沪)新登字 301 号

UNIX 高级编程技术

周祖伦 尤晋元
王 舟 王保定 朱建忠 编著

上海科学技术文献出版社出版发行
(上海市武康路 2 号 邮政编码 200031)
全国新华书店 经销 上海海峰印刷厂印刷

开本 787×1092 1/16 印张 22.75 字数 568,000
1994 年 4 月第 1 版 1994 年 4 月第 1 次印刷
印数:1—2,500
ISBN 7-5439-0308-3/T・294
定价:23.60 元
《科技新书目》303-274

前　言

UNIX 系统问世以来已经度过了二十几个春秋。在此期间，它由一个规模小、功能比较单一、应用范围比较狭窄的操作系统发展成为规模较大、功能丰富、配置在从巨到微的各种计算机机型上、应用范围非常广泛的计算机软件平台。UNIX 系统在计算机科学工程界获得了最高的学术上的荣誉；它在计算机产业界推动了开放式系统的迅速发展，并成为这种系统的基本组成部分。它所获得的巨大成就和在计算机界引发的突飞猛进的变化令世人瞩目。可以预料，在国际计算机界的共同努力下，UNIX 将会顺应时代潮流更加迅速地发展，得到更广泛的应用，从而成为能够进入 21 世纪的少数几个软件系统平台中的一个。

为促进 UNIX 系统在我国的进一步推广，提高广大 UNIX 程序员的程序设计水平，我们将多年积累的 UNIX 程序设计经验进行总结归纳，并以 UNIX 诞生地 AT&T 最近推出和得到较广泛使用的 UNIX SVR 3.2 和 SVR 4.0 版为描述背景，编写了本书。

本书重点讨论 UNIX 系统的程序设计接口——系统调用的应用技术。系统调用是用户与内核的唯一接口，所有 UNIX 命令程序，例如 sdb、vi、ps 和 who 等都建立在此基础之上。在 UNIX 平台上进行软件开发工作的有关人员应当了解并掌握系统调用的使用方法及有关技术。

本书的主要特点是非常注意理论与实践相结合，面向应用、面向实践。本书提供了近 90 个完整的应用实例或程序段，这些应用实例程序可直接在机器上使用。每个应用实例程序都有较强的针对性，不是提醒读者在使用中应注意的问题，就是说明在程序设计中使用 UNIX 系统调用的方法与技巧。

全书共分十章和三个附录，内容如下：

第一章是系统概况，介绍了 UNIX 的发展、结构、功能和其中常用的概念、术语等。

第二章是文件基本操作，详细讨论了文件操作中最基本的系统调用的使用。

第三章是文件高级操作，详细讨论了较深层次的与文件操作有关的系统调用的使用。

第四章是设备文件操作，详细讨论了设备文件操作的特殊性和用于字符设备控制的系统调用的使用。

第五章是进程控制，详细讨论了用于进程控制系统调用的使用。

第六章是进程基本通信，详细讨论了信号、跟踪和管道通讯等方面系统调用的使用。

第七章是进程高级通信，详细讨论了消息通信、共享主存段和信号量等方面系统调用的使用。

第八章是 STREAMS 机制，详细介绍和讨论了流机制及其相关系统调用的使用。

第九章详细讨论了系统管理方面系统调用的使用。

第十章讨论了两个综合实例的设计。

附录 A：SVR4.0 主要系统调用列表。

附录 B：SVR4.0 ioctl 控制命令。

附录 C：UNIX System V 系统调用错误号。

参加本书编写工作的主要有：周祖伦、王保定、朱建忠(江南计算技术研究所)，尤晋元、王舟(上海交通大学计算机系)。朱建涛也撰写了部分章节的初稿。

在本书的编写过程中，得到了周德魁、王正德等领导的大力支持和热情鼓励，得到了周金炎、王海生的技术指导与帮助。此外，尉红梅、林珊珊、朱涛、朱爱红、张惠明、王侃等同志为本书的早日问世付出了大量的心血，在此，作者对他(她)们给予我们的支持和帮助表示最诚挚的感谢。

由于作者水平有限，加之时间较为匆忙，书中一定还有错误和不妥之处，恳切希望读者予以批评、指正。

作　者

1993 年 2 月

目　录

第一章 引 论

本章是全书的引子，阅读后读者可对 UNIX 系统及其发展概况有大致的了解。如读者已对该系统有一定的了解，则可越过本章，直接学习后续章节。

本章分为三节：第一节说明 UNIX 系统的发展概况，以及它的结构与特点；第二节介绍该系统中一些常用的概念和术语，例如：文件、文件系统、进程、进程通信等；第三节简要描述本书要讨论的内容——系统调用，介绍系统调用的概念、意义以及一般使用方法。

第一节 UNIX 系统概论

UNIX 系统自问世以来，以其简洁性和不断地自我完善而迅速风靡计算机界。当前无论是巨型机、大型机、中型机、工作站还是微型机，以 UNIX 作为其主操作系统配置的已经越来越多。世界上几乎所有重要的计算机厂商，都加入了 UNIX 系统的研究、开发、生产和销售的行列。作为系统软件的 UNIX 获得了如此巨大的成功并非偶然，这与其产生的时代背景及它自身具有的特点是分不开的。

1.1.1 UNIX 系统的发展概况

自 70 年代初由 AT&T 下属贝尔实验室的 K. Thompson 和 D. M. Ritchie 设计成功 UNIX 以来，UNIX 已走过了一段不平凡路程，图 1-1 是 UNIX 及类 UNIX 各主要版本的发展概况。

除了图 1-1 中所表示的各种 UNIX 主要版本外，在 80 年代 UNIX 的大发展时期还出现过大量其它 UNIX 版本，它们的功能虽然大体相同，但又存在这样或那样的差别。这不但影响了应用软件在这些 UNIX 版本之间的移植性，同时也影响了 UNIX 系统的进一步发展，违背了 UNIX 的最初设计原则——可移植性。随着 UNIX 系统在全球装机数量的迅猛增加及美国政府的大力支持，计算机界的有识之士及众多计算机系统厂商意识到 90 年代 UNIX 必将成为世界范围内的软件平台，并在一定程度上主宰操作系统市场，因此工业界和广大用户对统一标准的 UNIX 的呼声变得越发强烈。鉴于这种情况，在 1984 年下半年，若干著名的计算机厂商组织成立了 X/OPEN，它从事确定公共应用环境方面的标准化工作；IEEE 则于 1985 年成立了 P1003 委员会，开始制定基于 UNIX 的可移植的操作系统标准，并将其命名为 POSIX 。POSIX 包括了一组标准，其中一部分已经制定完成，并被国际标准化组织(ISO)采纳，成为正式的国际标准，大部分正在制定中，可望在近几年内完成。为了赢得广大用户的支持，迅速取得世界范围内更多的 UNIX 市场，UNIX 的始祖 AT&T 公司在 1987 年与 SUN Microsystem 计算机公司携手进行 UNIX 的开发，计划研制出符合标准要求的 UNIX 系统，以解决 UNIX 的不兼容问题。该计划引起了很多计算机公司的不安，它们

担心这将产生不公平的竞争，因此在 1988 年 5 月成立了 OSF（开放软件基金会），IBM、DEC、HP、HITACHI 和 Bull 等大公司参加了该组织，并联合开发一个开放式的软件环境 OSF/1，其中含有符合标准的操作系统部分。为了分庭抗礼，AT&T 等公司于 1988 年 11 月组织成立了 UI（UNIX 国际），AT&T、UNISYS、SUN、NCR、NEC 和 FUJITSU 等公司参加了 UI。

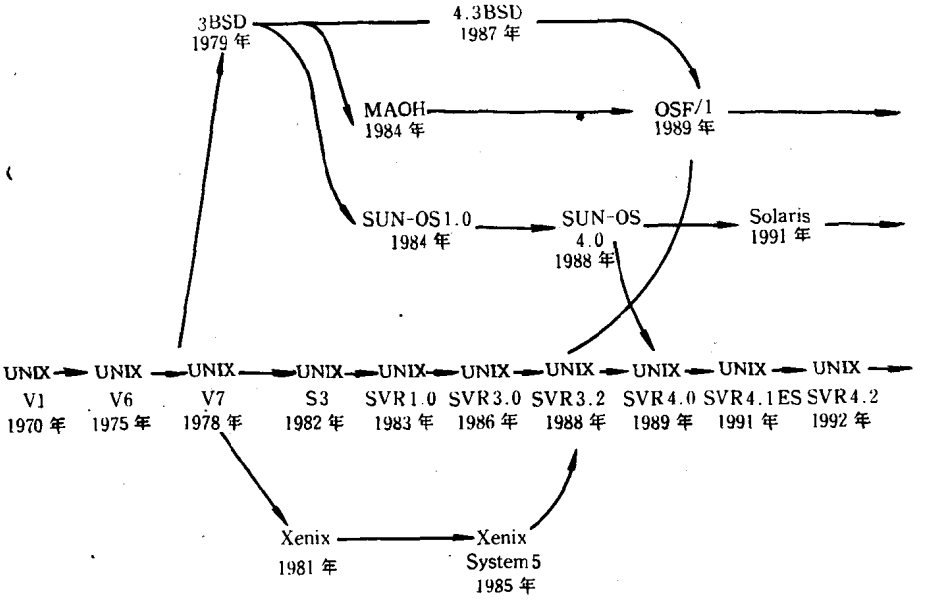

图 1-1　UNIX 系统发展图

虽然，UI 和 OSF 之间的竞争与合作可能产生两种 UNIX 系统，但是它们都在引进现有成功经验和技术的基础上，开发出符合标准规范的 UNIX 产品，从而促进 UNIX 系统的进一步发展和完善。可以预言：统一的 UNIX 系统必将是计算机工业发展的一个潮流及世界范围内的一种主要软件平台。

1.1.2 UNIX 系统的体系结构

UNIX 系统可分为核心层和用户层，图 1-2 是 UNIX 系统简要的结构示意图，图 1-3 和图 1-4 分别是核心层和用户层的简要示意图。

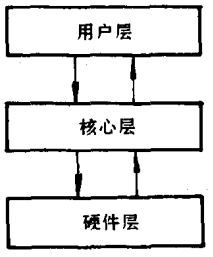

图 1-2　UNIX 系统结构示意图

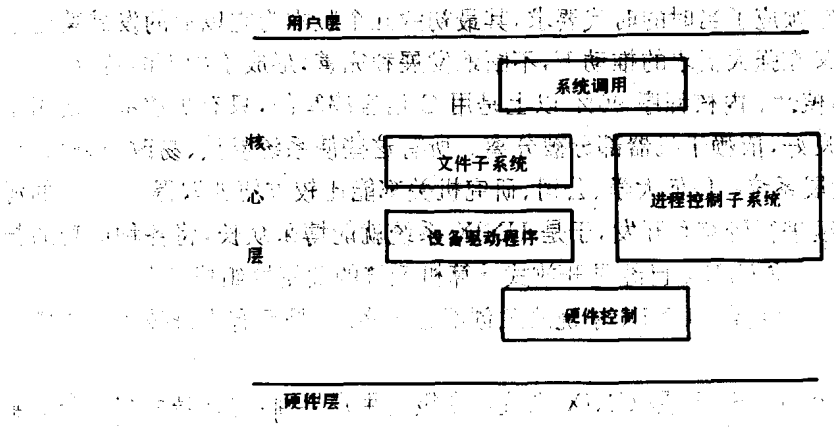

图 1-3 UNIX 核心层示意图

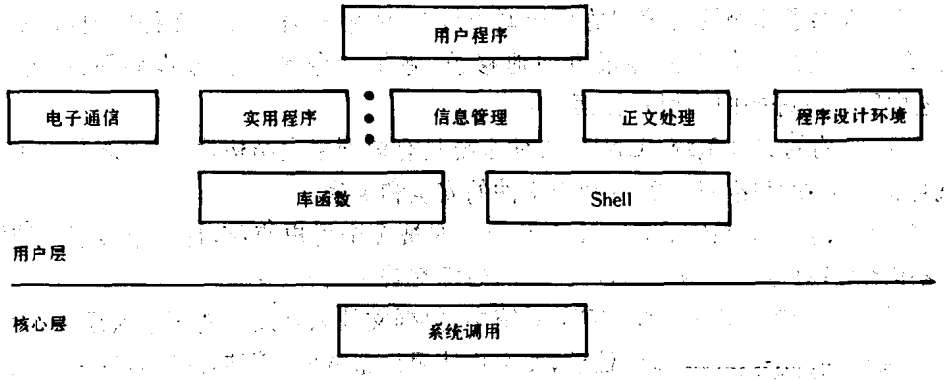

图 1-4 UNIX 用户层示意图

从图中可以看出，系统调用是核心与用户程序的界面，用户程序通过系统调用请求系统服务；shell 是用户与系统在命令级的界面，用户的所有命令都是通过 shell 提交给系统执行的。

1.1.3 UNIX 系统的特点

UNIX 系统在不到 20 年的时间里，取得如此惊人的成就并不是偶然的，有其内外的因素。外部环境为 UNIX 的产生，发展创造了良好条件，可以说 UNIX 是生逢其时，这主要表现在下列几方面：

(1) 操作系统的设计和实现技术已经日趋成熟，积累了很多可被选用的技术、可以借鉴的经验和教训。

(2) 计算机产业蓬勃发展，新的计算机机型不断出现，中、小规模的计算机厂商不断产生。这就要求有一个易于获得、易于移植、功能丰富、以交互作用方式为主又具有良好发展潜力和前景的操作系统，以便迅速地配置到这些新的机型上，从而增加其开发速度，减少其开发费用。

(3) 计算机的应用日益深入到各行各业，广大计算机软件开发者和最终用户要求在各

种计算机上只运行少数几种性能良好的操作系统，以减少他们在应用软件方面的开发、移植和维护费用，产生较高经济效益，减少他们为熟悉一个新的应用环境所需的时间。

UNIX 出现时，它顺应了当时的时代要求，其最初的几个版本为它以后的发展奠定了良好的基础。此后，它又在强大需求的推动下，不断地发展和完善，形成了如下的特点：

(1) 良好的可移植性。内核程序 90% 以上是用 C 语言编写的，只有少数部分使用了汇编语言；其模块结构良好，依赖于机器部分被分离。所有这些使系统易读、易改，移植方便。

(2) 开放式的发展系统。任何大学、公司、研究机关都能比较方便地取得 UNIX 的源代码，并在其基础上继续进行研究和开发，于是 UNIX 系统就能博采众长，将各种成功的技术溶入到其新版本中。目前 UNIX 已经是开放式计算机系统的关键性组成部分。

(3) 成熟经验的完美结合。UNIX 系统的独创部分不多，它是已有先进技术和成熟经验的完美结合。

(4) 良好的用户界面。Shell 是 UNIX 系统命令级的用户界面，它既是用户命令的解释器，又是一种程序设计语言。作为语言，它可将各种命令有效地组合在一起构成新的命令，使命令的二次开发变得容易。系统调用是核外程序要求系统提供服务，进入核心的唯一手段，UNIX 向用户提供了高级语言编程的系统调用接口，使用户程序设计变得方便，也使得不同 UNIX系统上开发的软件产品能很方便地移植。除了命令界面和编程界面外，近几年来 UNIX 还配置了良好的图形用户界面。

(5) 层次结构的文件系统。UNIX 文件系统具有树型层次结构；并可动态装卸子文件系统，这既可扩充文件存储空间，又有利于文件的安全和保密。

(6) 多用户、多任务系统。UNIX 系统可同时被多个用户使用，是一个共享系统资源的多用户操作系统；对每一个用户而言，又可同时提交若干任务。

(7) 文件、设备统一处理。在 UNIX 系统中，数据文件只是简单的字节流序列，无任何的结构形式；且外部设备也作为文件处理，使用户使用设备方便、简洁，同时也简化了系统程序设计。

(8) 丰富的核外实用程序。UNIX 系统提供了大量的工具性软件，这不仅满足了大量用户的需求，也促进了 UNIX 系统的普及与推广。

第二节　UNIX 系统基本概念

在详细讨论各类系统调用使用之前，本节简要介绍了 UNIX 系统的一些基本概念与专用术语。由于本书并非 UNIX 系统工作原理的入门书籍，因而对这些概念与术语没有作更详尽的描述，需了解这方面知识的读者可参阅有关书籍。

1.2.1 文件和文件系统

文件系统是操作系统的一部分，负责存取和管理文件信息。它为用户提供了简单、一致的存取和管理信息的方法。

UNIX 系统中，文件系统具有很大的灵活性，它由基本文件系统和若干子文件系统组成，用户可以根据需要拆卸与安装子文件系统；UNIX 系统中各个文件通过目录组织成树型结构的文件系统，存放在存储介质(通常为磁盘)上。文件系统中的数据文件只是简单的字符

流而无任何内部结构，这使得用户可以根据需要任意组织其文件格式。从系统的 SVR 3.2 版开始，UNIX 提供了对多类型文件系统的支持，即允许多种不同类型的文件系统存在于同一系统中，例如 XENIX 文件系统、s5 文件系统等，通常使用的文件系统是 s5 文件系统，本书讨论的文件操作都基于 s5 文件系统类型。

文件由一系列的逻辑块组成，每个逻辑块可能含有 512、1024 或 2048 个字节，具体大小由系统实现时决定。不同文件系统的逻辑块大小可以不同，但同一个文件系统中的逻辑块大小是相同的。s5 文件系统中逻辑块大小通常为 1024 字节，有时也选为 2048 字节，这样的文件系统称之为 1K 文件系统和 2K 文件系统。使用逻辑块较大时，因为每一次磁盘操作可以传输更多的数据，操作所花费的时间较少，所以可提高磁盘和内存间数据的传输率。当然，逻辑块太大，存储的有效容量也将下降。

1. 文件系统

(1) s5 文件系统的内部结构

s5 文件系统的内部结构可用图 1-5 表示。

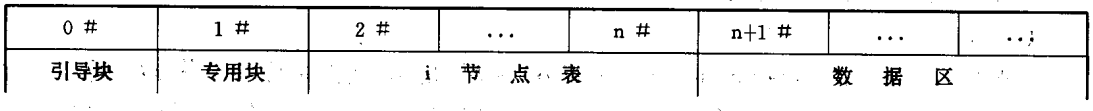

图 1-5 s5 文件系统结构

* 0 号盘块用于系统引导，称为引导块，存放系统的引导程序。虽然引导系统只需一个引导块，但 s5 中的每个文件系统都有自己的引导块(可能为空)。在 SVR4.0 中，引导系统的是 bfs 文件系统，它没有引导块，只有专用块、i 节点表和数据区。

* 1 号盘块称作专用块(superblock)，其中的内容按数据结构 filsys 进行存放和解释，它具体说明该文件系统的资源使用情况，filsys 结构内容如下：

```
struct filsys {
    ushort   s_isize;          /* 该文件系统中 i 节点表的块数 */
    daddr_t  s_fsize;          /* 整个文件系统的块数 */
    short    s_nfree;          /* 空闲磁盘块栈指针 */
    daddr_t  s_free[NICFREE];  /* 空闲磁盘块栈 */
    short    s_ninode;         /* 空闲 i 节点栈指针 */
    ushort   s_inode[NICINOD]; /* 空闲 i 节点栈 */
    char     s_flock;          /* 空闲块管理上锁标记 */
    char     s_ilock;          /* 空闲 i 节点管理上锁标记 */
    char     s_fmod;           /* 专用块修改标记 */
    char     s_ronly;          /* 只读安装标记 */
    time_t   s_time;           /* 上次专用块修改时间 */
    short    s_dinfo[4];       /* 设备信息 */
    daddr_t  s_tfree;          /* 文件系统中所有空闲块总数 */
```

```
    daddr_t   s_tinode;          /* 文件系统中所有空闲 i 节点总数 */
    long      s_fill[12];        /* 调整专用块大小的信息 */
                                 /* 对 80286,为 s_fill[14] */
    long      s_state;           /* 文件系统状态 */
    long      s_magic;           /* 新文件系统幻数 */
    long      s_type;            /* 新文件系统类型 */
}
```

其中的 s_isize 指明该文件系统中分配给存放 i 节点表的磁盘块数。s_fsize 指整个文件系统的总块数。s_nfree、s_free[NICFREE] 与 s_tfree 三项用于管理空闲磁盘块。s_ninode、s_inode[NICINOD] 与 s_tinode 用于管理空闲 i 节点。

* i 节点表存放在专用块以后。构造文件系统时,管理员可以指定 i 节点表的大小,核心通过 i 节点表的索引来引用 i 节点,下一部分将详细介绍 i 节点的结构。

* i 节点表之后是数据区,主要用于存放文件数据。

(2) 文件 i 节点

i 节点 (inode) 是 index node(索引节点)的缩写,是 s5 文件系统中最基本的概念。一个文件的控制信息通常由 i 节点给出,每个 i 节点对应着一个文件。i 节点中包含了文件数据在磁盘上存放的位置信息,还包含了如文件属主、访问权限、修改时间以及文件长度等信息。i 节点存放在盘上文件系统第二个逻辑块开始的连续区域内。以下列出了磁盘 i 节点 dinode 的结构,活动 i 节点 inode 的结构的说明参见系统的头文件 /usr/include/sys/inode.h。

```
struct dinode {
    o_mode_t        di_mode;     /* 文件类型和模式 */
    o_nlink_t       di_nlink;    /* 文件的链接计数 */
    o_uid_t         di_uid;      /* 文件主的用户标识数 */
    o_gid_t         di_gid;      /* 组用户的组标识数 */
    off_t           di_size;     /* 文件大小 */
    char            di_addr[39]; /* 数据块的磁盘地址 */
    unsigned char   di_gen;      /* 文件生成数 */
    time_t          di_atime;    /* 最后访问时间 */
    time_t          di_mtime;    /* 最后修改时间 */
    time_t          di_ctime;    /* 文件产生时间 */
}
```

其中的 di_mode、di_nlink、di_uid 及 di_gid 均为两字节长。di_addr 存放数据块地址,每个地址用三个字节表示,最后一字节保留为空。每个 i 节点占 64 字节的空间。

为了加快对文件的存取速度,提高系统效率和减轻 I/O 通道的负担,早期的 UNIX 系统在内存中开辟了一个活动 i 节点表,用以存放已打开文件的 i 节点,一般活动 i 节点表的大小为 100。活动 i 节点的内容,主要是复制盘上 i 节点得到的,同时有一些新的信息,例如

和其它活动 i 节点的勾连指针、i 节点是否已上锁等状态信息、包含该文件的文件系统的逻辑设备号、盘上相应的 i 节点号以及访问计数等等。在 SVR 4.0 中，核心中的活动 i 节点表由不依赖于文件系统的虚拟节点 vnode 结构表替代，它的大部分信息同样从盘上 i 节点中复制而来。有关 vnode 结构的具体内容可参见 SVR 4.0 版中系统的头文件 /usr/include/sys/vnode.h。

(3) 文件描述符

有关文件的各种操作，总离不开文件描述符，它与文件类系统调用的联系最密切。在 SVR 4.0 版以前的 UNIX 系统中，文件描述符是一个从 0 到 NOFILE-1 之间的整数，SVR 3.2 以前版 NOFILE 为 20，SVR 3.2 版时已改为 60。从中可看到在早期版本中，一个进程可打开的文件数是受限制的。在 SVR 4.0 中每个进程打开的文件数不受限制，可根据需要动态申请。

在 SVR3.2 中，进程的 user 结构中有一个域 u_ofile[NOFILE]，通常称用户打开文件表。它是属于各个进程私有的局部数据结构。每个用户进程打开的文件都在 u_ofile 中占用一项，该项的数组下标就是其文件描述符，该项的内容则指向系统打开文件表的某一项。系统打开文件表是一个全局的核心结构，可供多个进程共享，其中包括了文件访问计数、活动 i 节点(SVR 4.0 中为活动 v 节点) 指针以及读/写指针等信息。

通常，用户打开文件表中的前三项是在进程创建时就被分配的。它们分别代表标准输入(0#)、标准输出(1#) 以及标准错误信息输出(2#)。这三个文件描述符在 read、write 系统调用中可以直接使用。UNIX 系统中，五个系统调用 open、creat、fcntl、pipe 和 dup 为进程申请文件描述符，close 系统调用为系统回收文件描述符。

2. 文件及其分类

UNIX 文件按其使用形式分为五类：普通文件、目录文件、设备文件、管道文件和符号连接文件(SVR 4.0 版新增加)。

(1) 普通文件

普通文件仅含字节数据，被组织成字节流形式，存放在盘上。对文件可进行读、写操作，操作的位置由文件指针指明，文件指针不受文件大小的限制，可越过文件体，因此使用时需特别注意。

(2) 目录文件

目录文件的文件数据是有一定格式的，它由一些目录项组成。每个目录项和该目录下的一个文件相对应。一般情况下，每个目录文件中都包含“.”和“..”两个目录项，分别代表当前目录和父目录，它们通常在目录创建时生成。一个不包含“.”或“..”目录项的目录是不完整的，若无“..”目录项，用户就不能方便地访问当前目录的父目录。目录项由一个 i 节点号和文件名组成。i 节点号占两个字节，文件名一般占 14 个字节，因此 UNIX 中的文件名一般不得多于 14 个字符。

(3) 特别文件

特别文件也称设备文件，如磁盘、磁带、终端、打印机等都是设备文件。它有两种类型：块设备和字符设备文件。

字符设备文件是以字符为单位进行处理的；而块设备文件有如下特点：含有固定大小的

块；核心提供缓冲池来减少块设备访问频率，提高 I/O 传输率。在有大量数据的传送时，为了追求效率，一般使用字符设备文件，而不使用块设备文件，因为驱动程序允许在主存和设备之间建立直接存储存取，以 DMA 方式进行大量的数据传送。这种方式的传送越过了核心的缓冲池，使数据的传输量不再受核心缓冲池的限制，这是系统为什么将磁盘介质既作为块特别文件，又作为字符特别文件的原因。

(4) 管道文件

管道文件(也叫管道)是一个先进先出文件，主要用于进程间的数据传送。在 UNIX 系统里，管道文件有两种形式：命名 (FIFO) 和非命名 (pipe) 管道。命名管道有相应的文件名，非命名管道没有文件名。

管道文件的存贮方式与普通文件基本一样，但其最大文件长度比普通文件小得多。当管道建立后，从管道中读取数据的顺序和写入数据的顺序是一致的，系统不允许改变这种顺序。

(5) 符号连结文件

连结文件是 SVR 4.0 新增加的一种文件类型。它只与文件或目录建立一种符号上的连结关系，并不增加文件 i 节点的连结计数。用户可通过 "ls -l /unix" 命令列表根目录下的 unix 文件，可以看到它是一个符号连结文件。

```
                文件的连结计数
                      ↓
lrwxrwxrwx 1 root   root      11 May 16 16:55 /unix -> /stand/unix
↑
连结文件标志
```

在 SRV 3.2 之前的版本中，就存在文件的连结，在 SVR 4.0 版中将文件的这种连结称为硬连接，这种连结增加文件 i 节点的连结计数，它并非符号连结文件，也不是一种新的文件类型。一个文件可以有多个硬连结(link)，即可有多个文件名，用这些文件名中的任何一个访问文件时，都能找到同一个 i 节点，即找到同一个文件。从目录下移去一文件时，只有当该文件对应的 i 节点的连结数为 1 时，才将该文件的 i 节点和占用的数据块释放，否则只将该 i 节点的连结数减 1。对目录文件只提供符号连结，没有提供硬连结。

3. 文件的保护

(1) 文件主及其标识符

UNIX 系统将用户分为三类。对任一文件，用户可能是该文件的文件主用户，或者是该文件主用户的同组用户，或者是其它用户。这三类用户被分别授予不同的访问权限，实施不同的文件保护，这就是文件三级保护的概念。

一般情况下，文件主就是建立该文件的用户。每个文件还应属于某一个用户组，它被称为该文件的用户组。相应地，这个用户主(用户组)标识符称为文件主(文件的用户组)标识符。它们都是可以改变的，但只有超级用户和该文件主才能作这种改变。

(2) 文件模式和访问权限

一个文件的访问权限存放在该文件 i 节点的 di_mode 域中 。di_mode 称作文件模式，是一个十六位的整数。文件模式 di_mode 的 0 — 8 位表示文件主、用户组和其它用户对该文件