国外计算机科学教材系列

操作系统:设计与实现

(第2版)

(下册)

OPERATING SYSTEMS

Design and Implementation (Second Edition)

ANDREW S. TANENBAUM
ALBERT S. WOODHULL
著

王 鹏 尤晋元
朱 鹏 敖青云
译校

随书附光盘

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
PRENTICE HALL 出版公司

国外计算机科学教材系列

操作系统：设计与实现

（第2版）

（下册）

Operating Systems

Design and Implementation

second edition

Andrew S. Tanenbaum, Albert S. Woodhull 著

王 鹏 尤晋元 朱 鹏 敖青云 译校

PRENTICE HALL 出版公司

電子工業出版社

内 容 提 要

本书共6章，涵盖了操作系统课程的所有内容。即传统上的进程管理、存储器管理、文件管理和设备管理，同时又包含线程、基于消息传递的系统的构造模型、日志结构文件系统、安全保护机制、RAM及CD-ROM盘等，且以Pentium CPU作为实例。这样，既能学习操作系统的经典内容，又能了解当前最新技术。

本书为第二版，其第一版于1987年出版时，曾引发了操作系统课程教学的一场小变革。因为，在那以前多数教材只讲理论，而本教材却是基于理论与具体实例(MINIX)的结合。这对于掌握操作系统的设计与实现是大有裨益的。

本书分为上、下两册。上册为正文部分；下册为三个附录及随书光盘。

丛 书 名：国外计算机科学教材系列
原 书 名：OPERATING SYSTEMS: Design and Implementation (second edition)
书　　名：**操作系统：设计与实现（第2版）（下册）**
著　　者：Andrew S. Tanenbaum, Albert S. Woodhull 著
译 校 者：王 鹏 尤晋元 朱 鹏 敖青云
责任编辑：邓又强
印 刷 者：顺义县天竺颖华印刷厂印刷
出版发行：电子工业出版社出版、发行　URL: http://www.phei.com.cn
北京市海淀区万寿路173信箱　邮编　100036　发行部电话 68214070
经　　销：各地新华书店经销
开　　本：787×1092 1/16　印张：22.75　字数：538千字
版　　次：1998年8月第1版　1998年8月第1次印刷
印　　数：7000册
书　　号：ISBN 7-5053-4924-4/TP·2415
定　　价：48.00元
著作权合同登记号　图字：01-98-0965
凡购买电子工业出版社的图书，如有缺页、倒页、脱页者，本社发行部负责调换

出版说明

计算机科学的迅速发展是20世纪科学发展史上最伟大的事件之一。从1946年第一台笨重而体积庞大的计算机的发明至今，仅仅半个多世纪，计算机已经变得小巧无比却又能力非凡。它的应用已经渗透到了社会的各个方面，成为当今所谓的信息社会的最显著的特征。

处于世纪之交科技进步的大潮中，我国正在加强计算机科学的高等教育，着眼于为下一世纪培养高素质的计算机人才，以适应信息社会加速度发展的需要。当前，全国各类高等院校已经或计划在各专业基础课程规划中增加计算机科学的课程内容，而作为与计算机科学密切相关的计算机、通信、信息等专业，更是在酝酿着教学的全面革新，以期规划出一整套面向21世纪的、具有中国高校计算机教育特色的课程计划和教材体系。值此，我们不妨借鉴并引进国外具有先进性、实用性和权威性的大学计算机教材，洋为中用，以更好地服务于国内的高校教育。

美国Prentice Hall出版公司是享誉世界的高校教材出版商，自1913年公司成立以来，即致力于教育图书的出版。它所出版的计算机教材在美国为众多大学所采用，其中有不少是专业领域中的经典名著。许多蜚声世界的教授学者成为该公司的资深作者，如：道格拉斯·科默(Douglas Comer)，安德鲁·坦尼伯姆(Andrew Tanenbaum)，威廉·斯大林(William Stallings)……几十年来，他们的著作教育了一批批不同肤色的莘莘学子，使这些教材同时也成为全人类的共同财富。

为了保证本系列教材翻译出版的质量，电子工业出版社和Prentice Hall出版公司共同约请北京地区的清华大学、北京大学、北京航空航天大学，上海地区的上海交通大学、复旦大学，南京地区的南京大学、解放军通信工程学院等全国著名的高等院校的教学第一线的几十位教师参加翻译工作。这中间有正在讲授同类教材的年轻教师和博士，有积累了几十年教学经验的教授和博士生导师，还有我国著名的计算机科学家。他们的辛勤劳动保证了本系列丛书得以高质量地出版面世。

如此大规模地引进计算机科学系列教材，在我们还是第一次。除缺乏经验之外，还由于我们对计算机科学的发展，对中国高校计算机教育特点认识的不足，致使在选题确定、翻译、出版等工作中，肯定存在许多遗憾和不足之处，恳请广大师生和其他读者提出批评、建议。

电子工业出版社

URL：http://www.phei.com.cn

Prentice Hall出版公司

URL：http://www.prenhall.com

译者序

坦尼鲍姆教授是国际知名的计算机科学家和教育家。他在操作系统、分布式系统以及计算机网络领域都有很深的造诣。自80年代以来，他已先后出版了一系列面向大学生和研究生的教材性质的专著，并被世界各国的许多大学广泛采用。这本书就是他的最新专著之一。

操作系统是计算机系统中最核心和最底层的软件，对操作系统的深入学习关系到对整个系统运作机制的全面理解，因此一本好教材也显得愈发重要。本书的英文版出版于1997年，其中涵盖了操作系统课程的所有内容，即传统上的进程管理、存储器管理、文件管理和设备管理。同时其中又包含了许多新内容，如线程、基于消息传递的系统构造模型、日志结构文件系统、安全和保护机制、RAM盘及CD-ROM设备等，而用作例子的CPU则为Interl Pentium。这使得读者一方面能够学习操作系统的经典内容，另一方面又能够了解和跟踪当前的最新技术和研究成果。

本书的另一个特点是基本原理与具体实例，即MINIX紧密结合。第2到第5章的前半部分讲述原理，后半部分则详细地解释这些原理在MINIX的设计和实现中的应用。通过阅读这些部分能够把握MINIX源代码的组织方式，并理解那些很关键或者很难懂的代码。这部分内容非常翔实，有时甚至逐行地解释附录中所列的源程序。对操作系统课程多年的授课经验以及相关的科研工作使我们认识到：详细地剖析一个像MINIX这样的操作系统对于掌握操作系统设计与实现的精髓是大有裨益的。

正因为上述原因，我们真切地感受到将这本书翻译、介绍给国内读者将是一件非常有意义的事，衷心希望我们付出的劳动能对国内的操作系统教学和实践有所帮助和促进。

本书的第1章，第2章，第3章由王鹏翻译，刘福岩和陆宁也参加了部分工作；第四章由朱鹏翻译；第五章由敖青云翻译。全书由尤晋元教授审校并统稿。

在整个翻译过程中，上海交通大学计算机系系统软件研究室的师生给予了许多帮助。并且在计算机系95级本科生的操作系统课程中进行了试用，许多学生提出了很好的建议，在此向他们表示衷心的感谢。

特别要感谢本书的责任编辑邓又强编审，本书的顺利出版与他的辛勤劳动和热情支持是分不开的。

虽然在翻译过程中我们尽力恪守“信，达，雅”的准则，但不当和疏漏之处在所难免，敬请读者提出宝贵建议。

译　者

上海交通大学计算机科学与工程系

1998.4

前　言

多数操作系统教材都重理论而轻实践，本书希望在这二者之间求取较好的平衡。本书详细论述了操作系统的所有基本概念，包括进程、进程间通信、信号量、管程、消息传递、调度算法、输入/输出、死锁、设备驱动程序、存储器管理、页面调度算法、文件系统设计、安全与保护机制等。同时，本书也详细讨论了 MINIX——一个与 UNIX 兼容的操作系统，并提供了完整的源代码供学习之用。这样的安排使读者不仅学习到理论，而且能够理解它们如何应用在一个实际的操作系统之中。

本书第一版在 1987 年出版时，曾引发了操作系统课程教学的一场小小的变革。在此之前多数课程都只讲理论。随着 MINIX 的出现，许多学校开始增加实验环节以使学生了解实际的操作系统是如何运作的。我们认为这种趋势是可取的，并希望通过本书第二版能进一步加强这种趋势。

MINIX 在其出现以来的十年间发生了许多变化，最初的代码是为基于 8088 芯片、256K 内存和两个软驱的 IBM PC 机型编写的，它基于 UNIX 版本 7。随着时间的推移，MINIX 在许多方面有所发展，比如当前版本可运行在众多机型上，从 16 位实模式的 PC 机到配有大容量硬盘的奔腾机(32 位保护模式)，而且它不再基于 UNIX 版本 7，而是基于国际上的 POSIX 标准(POSIX 1003.1 和 ISO9945-1)。与此同时，有许多新特征被添加到 MINIX 中，在我们看来，所增加的特征可能已经太多了，但有些人则认为还不够，这最终导致了 LINUX 的诞生。MINIX 还被移植到许多其他平台上，包括 Macintosh、Amiga、Atari 和 SPARC。本书只涉及 MINIX2.0，到目前为止，该版本只能运行于基于 80x86 的机器，或者可模拟此类 CPU 的机器，以及 SPARC 机器。

与第一版相比，第二版有许多变化，原理性部分基本都被修改过，同时增加了大量新内容。最主要的变化是新的基于 POSIX 的 MINIX，以及对其源代码的剖析。另外，每本书都附带一张 CD－ROM，它包含了全部 MINIX 源代码，以及在 PC 上安装 MINIX 的说明(见 CD－ROM 主目录下的 README.TXT 文件)。

在一台 80x86 的 PC 机上安装 MINIX 很方便。它需要一个至少 30MB 的硬盘分区，然后按照 CD-ROM 上 README.TXT 文件中的步骤进行即可。在打印 README.TXT 文件之前，先启动 MS－DOS(若运行 WINDOWS，则双击 MS-DOS 图标)，然后键入

```
copy readme.txt prn
```

即可。该文件也可以用 edit、wordpad、notepad 等任何可以处理 ASCII 正文的编辑器进行浏览。

对于没有 PC 机的学校和个人，有两种解决办法，即 CD-ROM 上提供的两个模拟程序。一个由 Paul Ashton 为 SPARC 机器编写，它作为用户程序在 Solaris 上运行，此时 MINIX 被编译成 SPARC 上的可执行文件。在这种模式下，MINIX 不再是一个操作系统，而只是一个用户程序，所以必须对其底层作一些修改。

另一个模拟程序由 Bochs 软件公司的 Kevin P. Lawton 编写，它解释 Intel 80386 的指令集以及足以使 MINIX 运行所需的 I/O 指令。显然，在解释器层次上运行使性能有所下降，但这使得学生更容易进行调试。该模拟程序运行在所有支持 M. I. T 的 X-Window 的系统上，更详细的信息请参看 CD-ROM 上的有关文件。

MINIX 仍在继续发展，本书和 CD -ROM 中的内容仅仅反映了本书出版时的情况，有关 MINIX 的最新动态请访问 MINIX 的主页：http://www.cs.vu.nl/~ast/minix.html。MINIX 也有 USENET 中的新闻组：comp.os.minix，读者可以订阅该新闻组。对于仅有 Email 的读者可通过以下步骤来加入 MINIX 的邮件用户通信组。给 listserv@listserv.nodak.edu 发一封信，其中只需一行字："subscribeminix-1＜您的完整用户名＞"，此后你便会通过 E-mail 获得很多的信息。

讲授本课程的教师可以从 Prentice Hall 出版公司获得一份习题解答手册。从 WWW 地址 http://www.cs.vu.nl/ast/沿着"Software and supplementary material"链接可以获得一些有用的 PostScript 文件，其中包含本书中所有的图表，可供需要时使用。

在 MINIX 的开发项目中我们有幸得到了许多人的帮助。首先要感谢 Kees Bot 在 MINIX 标准化和软件发布中所作的大量工作，没有他的帮助，我们不可能完成这件工作。他自己编写了大量的代码(如 POSIX 终端 I/O)并修正了一些数年来一直存在的错误，他还整理了其他的代码。

这些年来 Bruce Evans, Philip Homburg, Will Rose 和 Michael Temari 为 MINIX 的开发做了大量的工作。有几百人通过新闻组对 MINIX 作出了贡献，他们人数众多，所作出的贡献也各不相同，在此谨向他们一并表示感谢。

John Casey, Dale Grit, Frans Kaashoek 等人阅读了本书的部分手稿并提出了宝贵建议，在此向他们表示谢意。

Vrije 大学的许多学生测试了 CD-ROM 中 MINIX 的 β 版本，他们是：Ahmed Batou, Goran Dokic, Peter Gijzel, Thomer Gil, Dennis Grimbergen, Roderick Groesbeek, Wouter haring, Guido Kollerie, Mark Lassche, Raymond Ris, Frans ter Borg, Alex van Ballegooy, Ries van der Velden, Alexander Wels 以及 Thomas Zeeman。我们对他们细致的工作和详尽的报告致以衷心的感谢。

阿尔伯特·S·伍德豪尔向他从前的几位学生表示感谢，特别是 Hampshire 学院的 Peter W. Young，Nacional Autonoma de Nicaragua 大学的 Maria Isabel Sanchez 和 William Puddy Vargas。

最后要向我们的家庭成员表示感谢。Suzanne 已是第十次在我埋头写作时给我支持，对 Barbara 是第九次，Marvin 是第八次，甚至小 Bram 也是第四次了。他们的支持和爱心对我非常重要。(坦尼鲍姆)

至于阿尔伯特的 Barbara，这倒是第一次，假如没有她的支持，耐心和幽默，我们是不可能完成这一工作的，对我的儿子 Gordon 而言，由于在编写本书时，他大部分时间都不在家中，而是在大学学习，因此是非常幸运的。但是他的理解和关心深深吸引着我从事本书的编写工作，有这样一个儿子是令人非常愉快的。(伍德豪尔)

安德鲁·S·坦尼鲍姆
阿尔伯特·S·伍德豪尔

作者简介

安德鲁·坦尼鲍姆分别在麻省理工学院和加州大学伯克利分校获得学士和博士学位。他现任位于荷兰阿姆斯特丹市的 Vrije 大学计算机科学教授并领导着一个计算机系统研究小组。同时他还任一个研究并行、分布及图像系统的校际研究生院 - 计算机与图像高级学院的院长。

坦尼鲍姆先前的研究领域包括编译器、操作系统、网络和局域分布式系统，他现在的研究主要集中在可扩展到数百万用户的广域分布式系统。对这些课题的研究使他在学报和会议上发表了 70 余篇论文，并出版了五部专著。

坦尼鲍姆教授同时还主持开发了大量的软件。他是 Amsterdam 编译工具箱的总设计师，该工具箱被广泛地用来开发可移植的编译器，同时还用于 MINIX 的开发。他和他的博士研究生及程序员们一起设计了一个基于微内核的高性能分布式操作系统 - Amoeba。现在，以教学和研究为目的的用户可以从 Internet 上免费获得 MINIX 和 Amoeba 软件。

坦尼鲍姆的许多博士研究生在获得学位后都取得了非常丰硕的成果，这令坦尼鲍姆非常自豪，因为这是他诲人不倦的结果。

坦尼鲍姆教授同时还是 ACM 的会士、IEEE 高级会员、荷兰皇家艺术和科学院院士，他曾获得 1994 年 ACM Karl V. Karlstrom 杰出教育奖和 1997 年 ACM/SIGCSE 计算机科学教育杰出贡献奖。他被列入 Internet 上的 Who's Who in the World 名单，他在 WWW 上的主页地址为：http://www.cs.vu.nl/~ast/。

阿尔伯特·伍德豪尔分别在麻省理工学院和华盛顿大学获得学士和博士学位。他进麻省理工学院本来是想成为一名电气工程师，可是后来却成了生物学家。从 1973 年起他开始在位于麻省 Amherst 的 Hampshire 自然科学学院工作。当微型计算机慢慢多起来的时候，作为使用电子检测仪器的生物学家，他开始使用微型计算机。他给学生开设的检测仪器方面的课程逐渐演变为计算机接口和实时程序设计。

伍德豪尔博士对教学和科学技术的发展有浓厚的兴趣，在进入研究生院之前他曾在尼日利亚教过两年中学，近年来他曾几次利用自己的假期在尼加拉瓜教授计算机科学。

他对计算机作为电子系统，以及计算机与其他电子系统的相互配合很感兴趣。他最喜欢讲授的课程有计算机体系结构、汇编语言程序设计、操作系统和计算机通信。他还为开发电子器件及相关软件担当顾问。

在学术之外，伍德豪尔有不少兴趣，包括各种户外运动，业余无线电制作和读书。他还喜欢旅游和学习别国语言。他的 WWW 主页就存在一台运行 MINIX 的机器上，地址是：http://minix1.hampshire.edu/asw/。

关于本书所附 CD-ROM 的说明

该 CD-ROM 包含了全部 MINIX 源代码，以及在 PC 上安装 MINIX 的说明。关于 CD-ROM 的安装和使用，请参阅本书前言。

附录 A　MINIX 源代码表

++
include/ansi.h
++

```
00000   /* The <ansi.h> header attempts to decide whether the compiler has enough
00001    * conformance to Standard C for Minix to take advantage of. If so, the
00002    * symbol _ANSI is defined (as 31415). Otherwise _ANSI is not defined
00003    * here, but it may be defined by applications that want to bend the rules.
00004    * The magic number in the definition is to inhibit unnecessary bending
00005    * of the rules. (For consistency with the new "#ifdef _ANSI" tests in
00006    * the headers, _ANSI should really be defined as nothing, but that would
00007    * break many library routines that use "#if _ANSI".)
00008
00009    * If _ANSI ends up being defined, a macro
00010    *
00011    *          _PROTOTYPE(function, params)
00012    *
00013    * is defined. This macro expands in different ways, generating either
00014    * ANSI Standard C prototypes or old-style K&R (Kernighan & Ritchie)
00015    * prototypes, as needed. Finally, some programs use _CONST, _VOIDSTAR etc
00016    * in such a way that they are portable over both ANSI and K&R compilers.
00017    * The appropriate macros are defined here.
00018    */
00019
00020   #ifndef _ANSI_H
00021   #define _ANSI_H
00022
00023   #if __STDC__ == 1
00024   #define _ANSI                31459    /* compiler claims full ANSI conformance */
00025   #endif
00026
00027   #ifdef __GNUC__
00028   #define _ANSI                31459    /* gcc conforms enough even in non-ANSI mode */
00029   #endif
00030
00031   #ifdef _ANSI
00032
00033   /* Keep everything for ANSI prototypes.                    */
00034   #define _PROTOTYPE(function, params)    function params
00035   #define _ARGS(params)                   params
00036
00037   #define _VOIDSTAR          void *
00038   #define _VOID              void
00039   #define _CONST             const
00040   #define _VOLATILE          volatile
00041   #define _SIZET             size_t
00042
00043   #else
00044
00045   /* Throw away the parameters for K&R prototypes. */
00046   #define _PROTOTYPE(function, params)    function()
00047   #define _ARGS(params)                   ()
00048
00049   #define _VOIDSTAR          void *
00050   #define _VOID              void
00051   #define _CONST
00052   #define _VOLATILE
```

```
00053   # define _SIZET          int
00054
00055   # endif /* _ANSI */
00056
00057   # endif /* ANSI_H */
```

++

include/limits.h

++

```
00100   /* The <limits.h> header defines some basic sizes, both of the language types
00101    *      (e.g., the number of bits in an integer), and of the operating system (e.g.
00102    * the number of characters in a file name.
00103    */
00104
00105   # ifndef _LIMITS_H
00106   # define _LIMITS_H
00107
00108   /* Definitions about chars (8 bits in MINIX, and signed). */
00109   # define CHAR_BIT          8     /* # bits in a char */
00110   # define CHAR_MIN       -128     /* minimum value of a char */
00111   # define CHAR_MAX        127     /* maximum value of a char */
00112   # define SCHAR_MIN      -128     /* minimum value of a signed char */
00113   # define SCHAR_MAX       127     /* maximum value of a signed char */
00114   # define UCHAR_MAX       255     /* maximum value of an unsigned char */
00115   # define MB_LEN_MAX 1/* maximum length of a multibyte char */
00116
00117   /* Definitions about shorts (16 bits in MINIX). */
00118   # define SHRT_MIN      (-32767-1)     /* minimum value of a short */
00119   # define SHRT_MAX          32767     /* maximum value of a short */
00120   # define USHRT_MAX         0xFFFF    /* maximum value of unsigned short */
00121
00122   /* _EM_WSIZE is a compiler-generated symbol giving the word size in bytes. */
00123   # if _EM_WSIZE == 2
00124   # define INT_MIN       (-32767-1)     /* minimum value of a 16-bit int */
00125   # define INT_MAX           32767     /* maximum value of a 16-bit int */
00126   # define UINT_MAX          0xFFFF    /* maximum value of an unsigned 16-bit int */
00127   # endif
00128
00129   # if _EM_WSIZE == 4
00130   # define INT_MIN   (-2147483647-1)     /* minimum value of a 32-bit int */
00131   # define INT_MAX   2147483647          /* maximum value of a 32-bit int */
00132   # define UINT_MAX  0xFFFFFFFF          /* maximum value of an unsigned 32-bit int */
00133   # endif
00134
00135   /* Definitions about longs (32 bits in MINIX). */
00136   # define LONG_MIN  (-2147483647L-1) /* minimum value of a long */
00137   # define LONG_MAX  2147483647L      /* maximum value of a long */
00138   # define ULONG_MAX 0xFFFFFFFFL      /* maximum value of an unsigned long */
00139
00140   /* Minimum sizes required by the POSIX P1003.1 standard (Table 2-3). */
00141   # ifdef _POSIX_SOURCE/* these are only visible for POSIX */
00142   # define _POSIX_ARG_MAX      4096    /* exec() may have 4K worth of args */
00143   # define _POSIX_CHILD_MAX       6    /* a process may have 6 children */
00144   # define _POSIX_LINK_MAX        8    /* a file may have 8 links */
00145   # define _POSIX_MAX_CANON     255    /* size of the canonical input queue */
00146   # define _POSIX_MAX_INPUT     255    /* you can type 255 chars ahead */
00147   # define _POSIX_NAME_MAX       14    /* a file name may have 14 chars */
00148   # define _POSIX_NGROUPS_MAX     0    /* supplementary group IDs are optional */
00149   # define _POSIX_OPEN_MAX       16    /* a process may have 16 files open */
00150   # define _POSIX_PATH_MAX      255    /* a pathname may contain 255 chars */
00151   # define _POSIX_PIPE_BUF      512    /* pipes writes of 512 bytes must be atomic */
00152   # define _POSIX_STREAM_MAX      8    /* at least 8 FILEs can be open at once */
00153   # define _POSIX_TZNAME_MAX      3    /* time zone names can be at least 3 chars */
00154   # define _POSIX_SSIZE_MAX   32767    /* read() must support 32767 byte reads */
```

```
00155
00156   /* Values actually implemented by MINIX (Tables 2-4, 2-5, 2-6, and 2-7). */
00157   /* Some of these old names had better be defined when not POSIX. */
00158   #define_NO_LIMIT                 100     /* arbitrary number; limit not enforced */
00159
00160   #define NGROUPS_MAX                 0     /* supplemental group IDs not available */
00161   #if_EM_WSIZE > 2
00162   #define ARG_MAX                 16384     /* # bytes of args + environ for exec() */
00163   #else
00164   #define ARG_MAX                  4096     /* args + environ on small machines */
00165   #endif
00166   #define CHILD_MAX_NO_LIMIT/* MINIX does not limit children */
00167   #define OPEN_MAX                   20     /* # open files a process may have */
00168   #define LINK_MAX                  127     /* # links a file may have */
00169   #define MAX_CANON                 255     /* size of the canonical input queue */
00170   #define MAX_INPUT                 255     /* size of the type-ahead buffer */
00171   #define NAME_MAX                   14     /* # chars in a file name */
00172   #define PATH_MAX                  255     /* # chars in a path name */
00173   #define PIPE_BUF                 7168     /* # bytes in atomic write to a pipe */
00174   #define STREAM_MAX                 20   /* must be the same as FOPEN_MAX in stdio.h */
00175   #define TZNAME_MAX                  3     /* maximum bytes in a time zone name is 3 */
00176   #define SSIZE_MAX                32767     /* max defined byte count for read() */
00177
00178   #endif /* _POSIX_SOURCE */
00179
00180   #endif /* _LIMITS_H */
```

++

include/errno.h

++

```
00200   /* The <errno.h> header defines the numbers of the various errors that can
00201    *      occur during program execution. They are visible to user programs and
00202    * should be small positive integers. However, they are also used within
00203    * MINIX, where they must be negative. For example, the READ system call is
00204    * executed internally by calling do_read(). This function returns either a
00205    * (negative) error number or a (positive) number of bytes actually read.
00206    *
00207    * To solve the problem of having the error numbers be negative inside the
00208    * the system and positive outside, the following mechanism is used. All the
00209    * definitions are are the form:
00210    *
00211    *      #define EPERM (_SIGN 1)
00212    *
00213    * If the macro_SYSTEM is defined, then_SIGN is set to "-", otherwise it is
00214    * set to "". Thus when compiling the operating system, the macro_SYSTEM
00215    * will be defined, setting EPERM to (-1), whereas when when this
00216    * file is included in an ordinary user program, EPERM has the value (1).
00217    */
00218
00219   #ifndef_ERRNO_H               /* check if <errno.h> is already included */
00220   #define_ERRNO_H               /* it is not included; note that fact */
00221
00222   /* Now define_SIGN as "" or "-" depending on_SYSTEM. */
00223   #ifdef_SYSTEM
00224   #  define_SIGN          -
00225   #  define OK             0
00226   #else
00227   #  define_SIGN
00228   #endif
00229
00230   extern int errno;                     /* place where the error numbers go */
00231
00232   /* Here are the numerical values of the error numbers. */
00233   #define_NERROR                  70     /* number of errors */
00234
```

```
00235   # define EGENERIC        (_SIGN 99)  /* generic error */
00236   # define EPERM           (_SIGN 1)   /* operation not permitted */
00237   # define ENOENT          (_SIGN 2)   /* no such file or directory */
00238   # define ESRCH           (_SIGN 3)   /* no such process */
00239   # define EINTR           (_SIGN 4)   /* interrupted function call */
00240   # define EIO             (_SIGN 5)   /* input/output error */
00241   # define ENXIO           (_SIGN 6)   /* no such device or address */
00242   # define E2BIG           (_SIGN 7)   /* arg list too long */
00243   # define ENOEXEC         (_SIGN 8)   /* exec format error */
00244   # define EBADF           (_SIGN 9)   /* bad file descriptor */
00245   # define ECHILD          (_SIGN 10)  /* no child process */
00246   # define EAGAIN          (_SIGN 11)  /* resource temporarily unavailable */
00247   # define ENOMEM          (_SIGN 12)  /* not enough space */
00248   # define EACCES          (_SIGN 13)  /* permission denied */
00249   # define EFAULT          (_SIGN 14)  /* bad address */
00250   # define ENOTBLK         (_SIGN 15)  /* Extension: not a block special file */
00251   # define EBUSY           (_SIGN 16)  /* resource busy */
00252   # define EEXIST          (_SIGN 17)  /* file exists */
00253   # define EXDEV           (_SIGN 18)  /* improper link */
00254   # define ENODEV          (_SIGN 19)  /* no such device */
00255   # define ENOTDIR         (_SIGN 20)  /* not a directory */
00256   # define EISDIR          (_SIGN 21)  /* is a directory */
00257   # define EINVAL          (_SIGN 22)  /* invalid argument */
00258   # define ENFILE          (_SIGN 23)  /* too many open files in system */
00259   # define EMFILE          (_SIGN 24)  /* too many open files */
00260   # define ENOTTY          (_SIGN 25)  /* inappropriate I/O control operation */
00261   # define ETXTBSY         (_SIGN 26)  /* no longer used */
00262   # define EFBIG           (_SIGN 27)  /* file too large */
00263   # define ENOSPC          (_SIGN 28)  /* no space left on device */
00264   # define ESPIPE          (_SIGN 29)  /* invalid seek */
00265   # define EROFS           (_SIGN 30)  /* read-only file system */
00266   # define EMLINK          (_SIGN 31)  /* too many links */
00267   # define EPIPE           (_SIGN 32)  /* broken pipe */
00268   # define EDOM            (_SIGN 33)  /* domain error (from ANSI C std) */
00269   # define ERANGE          (_SIGN 34)  /* result too large (from ANSI C std) */
00270   # define EDEADLK         (_SIGN 35)  /* resource deadlock avoided */
00271   # define ENAMETOOLONG    (_SIGN 36)  /* file name too long */
00272   # define ENOLCK          (_SIGN 37)  /* no locks available */
00273   # define ENOSYS          (_SIGN 38)  /* function not implemented */
00274   # define ENOTEMPTY       (_SIGN 39)  /* directory not empty */
00275
00276   /* The following errors relate to networking. */
00277   # define EPACKSIZE       (_SIGN 50)  /* invalid packet size for some protocol */
00278   # define EOUTOFBUFS      (_SIGN 51)  /* not enough buffers left */
00279   # define EBADIOCTL       (_SIGN 52)  /* illegal ioctl for device */
00280   # define EBADMODE        (_SIGN 53)  /* badmode in ioctl */
00281   # define EWOULDBLOCK     (_SIGN 54)
00282   # define EBADDEST        (_SIGN 55)  /* not a valid destination address */
00283   # define EDSTNOTRCH      (_SIGN 56)  /* destination not reachable */
00284   # define EISCONN         (_SIGN 57)  /* all ready connected */
00285   # define EADDRINUSE      (_SIGN 58)  /* address in use */
00286   # define ECONNREFUSED    (_SIGN 59)  /* connection refused */
00287   # define ECONNRESET      (_SIGN 60)  /* connection reset */
00288   # define ETIMEDOUT       (_SIGN 61)  /* connection timed out */
00289   # define EURG            (_SIGN 62)  /* urgent data present */
00290   # define ENOURG          (_SIGN 63)  /* no urgent data present */
00291   # define ENOTCONN        (_SIGN 64)  /* no connection (yet or anymore) */
00292   # define ESHUTDOWN       (_SIGN 65)  /* a write call to a shutdown connection */
00293   # define ENOCONN         (_SIGN 66)  /* no such connection */
00294
00295   /* The following are not POSIX errors, but they can still happen. */
00296   # define ELOCKED         (_SIGN 101) /* can't send message */
00297   # define EBADCALL        (_SIGN 102) /* error on send/receive */
```

```
00298
00299   /* The following error codes are generated by the kernel itself. */
00300   #ifdef _SYSTEM
00301   #define E_BAD_DEST      -1001   /* destination address illegal */
00302   #define E_BAD_SRC       -1002   /* source address illegal */
00303   #define E_TRY_AGAIN     -1003   /* can't send -- tables full */
00304   #define E_OVERRUN       -1004   /* interrupt for task that is not waiting */
00305   #define E_BAD_BUF       -1005   /* message buf outside caller's addr space */
00306   #define E_TASK          -1006   /* can't send to task */
00307   #define E_NO_MESSAGE    -1007   /* RECEIVE failed: no message present */
00308   #define E_NO_PERM       -1008   /* ordinary users can't send to tasks */
00309   #define E_BAD_FCN       -1009   /* only valid fcns are SEND, RECEIVE, BOTH */
00310   #define E_BAD_ADDR      -1010   /* bad address given to utility routine */
00311   #define E_BAD_PROC      -1011   /* bad proc number given to utility */
00312   #endif /* _SYSTEM */
00313
00314   #endif /* _ERRNO_H */
```

++

include/unistd.h

++

```
00400   /* The <unistd.h> header contains a few miscellaneous manifest constants. */
00401
00402   #ifndef _UNISTD_H
00403   #define _UNISTD_H
00404
00405   /* POSIX requires size_t and ssize_t in <unistd.h> and elsewhere. */
00406   #ifndef _SIZE_T
00407   #define _SIZE_T
00408   typedef unsigned int size_t;
00409   #endif
00410
00411   #ifndef _SSIZE_T
00412   #define _SSIZE_T
00413   typedef int ssize_t;
00414   #endif
00415
00416   /* Values used by access(). POSIX Table 2-8. */
00417   #define F_OK            0       /* test if file exists */
00418   #define X_OK            1       /* test if file is executable */
00419   #define W_OK            2       /* test if file is writable */
00420   #define R_OK            4       /* test if file is readable */
00421
00422   /* Values used for whence in lseek(fd, offset, whence). POSIX Table 2-9. */
00423   #define SEEK_SET        0       /* offset is absolute */
00424   #define SEEK_CUR        1       /* offset is relative to current position */
00425   #define SEEK_END        2       /* offset is relative to end of file */
00426
00427   /* This value is required by POSIX Table 2-10. */
00428   #define _POSIX_VERSION 199009L      /* which standard is being conformed to */
00429
00430   /* These three definitions are required by POSIX Sec. 8.2.1.2. */
00431   #define STDIN_FILENO    0       /* file descriptor for stdin */
00432   #define STDOUT_FILENO   1       /* file descriptor for stdout */
00433   #define STDERR_FILENO   2       /* file descriptor for stderr */
00434
00435   #ifdef _MINIX
00436   /* How to exit the system. */
00437   #define RBT_HALT        0
00438   #define RBT_REBOOT      1
00439   #define RBT_PANIC       2       /* for servers */
00440   #define RBT_MONITOR     3       /* let the monitor do this */
00441   #define RBT_RESET       4       /* hard reset the system */
00442   #endif
```

```
00443
00444   /* NULL must be defined in <unistd.h> according to POSIX Sec. 2.7.1. */
00445   #define NULL ((void *)0)
00446
00447   /* The following relate to configurable system variables. POSIX Table 4-2. */
00448   #define _SC_ARG_MAX                1
00449   #define _SC_CHILD_MAX              2
00450   #define _SC_CLOCKS_PER_SEC         3
00451   #define _SC_CLK_TCK                3
00452   #define _SC_NGROUPS_MAX            4
00453   #define _SC_OPEN_MAX               5
00454   #define _SC_JOB_CONTROL            6
00455   #define _SC_SAVED_IDS              7
00456   #define _SC_VERSION                8
00457   #define _SC_STREAM_MAX             9
00458   #define _SC_TZNAME_MAX             10
00459
00460   /* The following relate to configurable pathname variables. POSIX Table 5-2. */
00461   #define _PC_LINK_MAX               1       /* link count */
00462   #define _PC_MAX_CANON              2       /* size of the canonical input queue */
00463   #define _PC_MAX_INPUT              3       /* type-ahead buffer size */
00464   #define _PC_NAME_MAX               4       /* file name size */
00465   #define _PC_PATH_MAX               5       /* pathname size */
00466   #define _PC_PIPE_BUF               6       /* pipe size */
00467   #define _PC_NO_TRUNC               7       /* treatment of long name components */
00468   #define _PC_VDISABLE               8       /* tty disable */
00469   #define _PC_CHOWN_RESTRICTED       9       /* chown restricted or not */
00470
00471   /* POSIX defines several options that may be implemented or not, at the
00472    * implementer's whim. This implementer has made the following choices:
00473    *
00474    * _POSIX_JOB_CONTROL          not defined:     no job control
00475    * _POSIX_SAVED_IDS            not defined:     no saved uid/gid
00476    * _POSIX_NO_TRUNC             defined as -1:   long path names are truncated
00477    * _POSIX_CHOWN_RESTRICTED     defined:         you can't give away files
00478    * _POSIX_VDISABLE             defined:         tty functions can be disabled
00479    */
00480   #define _POSIX_NO_TRUNC            (-1)
00481   #define _POSIX_CHOWN_RESTRICTED    1
00482
00483   /* Function Prototypes. */
00484   #ifndef _ANSI_H
00485   #include <ansi.h>
00486   #endif
00487
00488   _PROTOTYPE( void _exit, (int _status)                                          );
00489   _PROTOTYPE( int access, (const char *_path, int _amode)                        );
00490   _PROTOTYPE( unsigned int alarm, (unsigned int _seconds)                        );
00491   _PROTOTYPE( int chdir, (const char *_path)                                     );
00492   _PROTOTYPE( int chown, (const char *_path, Uid_t _owner, Gid_t _group)         );
00493   _PROTOTYPE( int close, (int _fd)                                               );
00494   _PROTOTYPE( char *ctermid, (char *_s)                                          );
00495   _PROTOTYPE( char *cuserid, (char *_s)                                          );
00496   _PROTOTYPE( int dup, (int _fd)                                                 );
00497   _PROTOTYPE( int dup2, (int _fd, int _fd2)                                      );
00498   _PROTOTYPE( int execl, (const char *_path, const char *_arg, ...)              );
00499   _PROTOTYPE( int execle, (const char *_path, const char *_arg, ...)             );
00500   _PROTOTYPE( int execlp, (const char *_file, const char *arg, ...)              );
00501   _PROTOTYPE( int execv, (const char *_path, char *const _argv[])                );
00502   _PROTOTYPE( int execve, (const char *_path, char *const _argv[],
00503   char *const _envp[])                                                           );
00504   _PROTOTYPE( int execvp, (const char *_file, char *const _argv[])               );
00505   _PROTOTYPE( pid_t fork, (void)                                                 );
00506   _PROTOTYPE( long fpathconf, (int _fd, int _name)                               );
```

```
00507   _PROTOTYPE( char * getcwd, (char * _buf, size_t _size)                    );
00508   _PROTOTYPE( gid_t getegid, (void)                                         );
00509   _PROTOTYPE( uid_t geteuid, (void)                                         );
00510   _PROTOTYPE( gid_t getgid, (void)                                          );
00511   _PROTOTYPE( int getgroups, (int _gidsetsize, gid_t _grouplist[])          );
00512   _PROTOTYPE( char * getlogin, (void)                                       );
00513   _PROTOTYPE( pid_t getpgrp, (void)                                         );
00514   _PROTOTYPE( pid_t getpid, (void)                                          );
00515   _PROTOTYPE( pid_t getppid, (void)                                         );
00516   _PROTOTYPE( uid_t getuid, (void)                                          );
00517   _PROTOTYPE( int isatty, (int _fd)                                         );
00518   _PROTOTYPE( int link, (const char * _existing, const char * _new)         );
00519   _PROTOTYPE( off_t lseek, (int _fd, off_t _offset, int _whence)            );
00520   _PROTOTYPE( long pathconf, (const char * _path, int _name)                );
00521   _PROTOTYPE( int pause, (void)                                             );
00522   _PROTOTYPE( int pipe, (int _fildes[2])                                    );
00523   _PROTOTYPE( ssize_t read, (int _fd, void * _buf, size_t _n)               );
00524   _PROTOTYPE( int rmdir, (const char * _path)                               );
00525   _PROTOTYPE( int setgid, (Gid_t _gid)                                      );
00526   _PROTOTYPE( int setpgid, (pid_t _pid, pid_t _pgid)                        );
00527   _PROTOTYPE( pid_t setsid, (void)                                          );
00528   _PROTOTYPE( int setuid, (Uid_t _uid)                                      );
00529   _PROTOTYPE( unsigned int sleep, (unsigned int _seconds)                   );
00530   _PROTOTYPE( long sysconf, (int _name)                                     );
00531   _PROTOTYPE( pid_t tcgetpgrp, (int _fd)                                    );
00532   _PROTOTYPE( int tcsetpgrp, (int _fd, pid_t _pgrp_id)                      );
00533   _PROTOTYPE( char * ttyname, (int _fd)                                     );
00534   _PROTOTYPE( int unlink, (const char * _path)                              );
00535   _PROTOTYPE( ssize_t write, (int _fd, const void * _buf, size_t _n)        );
00536
00537   # ifdef _MINIX
00538   _PROTOTYPE( int brk, (char * _addr)                                       );
00539   _PROTOTYPE( int chroot, (const char * _name)                              );
00540   _PROTOTYPE( int mknod, (const char * _name, Mode_t _mode, Dev_t _addr)    );
00541   _PROTOTYPE( int mknod4, (const char * _name, Mode_t _mode, Dev_t _addr,
00542   long _size)                                                               );
00543   _PROTOTYPE( char * mktemp, (char * _template)                             );
00544   _PROTOTYPE( int mount, (char * _spec, char * _name, int _flag)            );
00545   _PROTOTYPE( long ptrace, (int _req, pid_t _pid, long _addr, long _data)   );
00546   _PROTOTYPE( char * sbrk, (int _incr)                                      );
00547   _PROTOTYPE( int sync, (void)                                              );
00548   _PROTOTYPE( int umount, (const char * _name)                              );
00549   _PROTOTYPE( int reboot, (int _how, ...)                                   );
00550   _PROTOTYPE( int gethostname, (char * _hostname, size_t _len)              );
00551   _PROTOTYPE( int getdomainname, (char * _domain, size_t _len)              );
00552   _PROTOTYPE( int ttyslot, (void)                                           );
00553   _PROTOTYPE( int fttyslot, (int _fd)                                       );
00554   _PROTOTYPE( char * crypt, (const char * _key, const char * _salt)         );
00555   # endif
00556
00557   # endif / * _UNISTD_H * /
```

++

include/string.h

++

```
00600   / * The <string.h> header contains prototypes for the string handling
00601    * functions.
00602    * /
00603
00604   # ifndef _STRING_H
00605   # define _STRING_H
00606
00607   # define NULL((void * )0)
```

```
00608
00609     # ifndef _SIZE_T
00610     # define _SIZE_T
00611     typedef unsigned int size_t;/ *  type returned by sizeof  * /
00612     # endif / * _SIZE_T  * /
00613
00614     / *  Function Prototypes.  * /
00615     # ifndef _ANSI_H
00616     # include <ansi.h>
00617     # endif
00618
00619     _PROTOTYPE( void * memchr, (const void * _s, int_c, size_t_n)                    );
00620     _PROTOTYPE( int memcmp, (const void * _s1, const void * _s2, size_t_n)           );
00621     _PROTOTYPE( void * memcpy, (void * _s1, const void * _s2, size_t_n)              );
00622     _PROTOTYPE( void * memmove, (void * _s1, const void * _s2, size_t_n)             );
00623     _PROTOTYPE( void * memset, (void * _s, int_c, size_t_n)                          );
00624     _PROTOTYPE( char * strcat, (char * _s1, const char * _s2)                        );
00625     _PROTOTYPE( char * strchr, (const char * _s, int_c)                              );
00626     _PROTOTYPE( int strncmp, (const char * _s1, const char * _s2, size_t_n)          );
00627     _PROTOTYPE( int strcmp, (const char * _s1, const char * _s2)                     );
00628     _PROTOTYPE( int strcoll, (const char * _s1, const char * _s2)                    );
00629     _PROTOTYPE( char * strcpy, (char * _s1, const char * _s2)                        );
00630     _PROTOTYPE( size_t strcspn, (const char * _s1, const char * _s2)                 );
00631     _PROTOTYPE( char * strerror, (int_errnum)                                        );
00632     _PROTOTYPE( size_t strlen, (const char * _s)                                     );
00633     _PROTOTYPE( char * strncat, (char * _s1, const char * _s2, size_t_n)             );
00634     _PROTOTYPE( char * strncpy, (char * _s1, const char * _s2, size_t_n)             );
00635     _PROTOTYPE( char * strpbrk, (const char * _s1, const char * _s2)                 );
00636     _PROTOTYPE( char * strrchr, (const char * _s, int_c)                             );
00637     _PROTOTYPE( size_t strspn, (const char * _s1, const char * _s2)                  );
00638     _PROTOTYPE( char * strstr, (const char * _s1, const char * _s2)                  );
00639     _PROTOTYPE( char * strtok, (char * _s1, const char * _s2)                        );
00640     _PROTOTYPE( size_t strxfrm, (char * _s1, const char * _s2, size_t_n)             );
00641
00642     # ifdef _MINIX
00643     / *  For backward compatibility.  * /
00644     _PROTOTYPE( char * index, (const char * _s, int_charwanted)                      );
00645     _PROTOTYPE( char * rindex, (const char * _s, int_charwanted)                     );
00646     _PROTOTYPE( void bcopy, (const void * _src, void * _dst, size_t_length)          );
00647     _PROTOTYPE( int bcmp, (const void * _s1, const void * _s2, size_t_length)        );
00648     _PROTOTYPE( void bzero, (void * _dst, size_t_length)                             );
00649     _PROTOTYPE( void * memccpy, (char * _dst, const char * _src, int_ucharstop,
00650      size_t_size)                                                                    );
00651     / *  BSD functions  * /
00652     _PROTOTYPE( int strcasecmp, (const char * _s1, const char * _s2)                 );
00653     # endif
00654
00655     # endif / * _STRING_H  * /
```

++

include/signal.h

++

```
00700     / *  The <signal.h> header defines all the ANSI and POSIX signals.
00701      * MINIX supports all the signals required by POSIX.  They are defined below.
00702      *  Some additional signals are also supported.
00703      * /
00704
00705     # ifndef _SIGNAL_H
00706     # define _SIGNAL_H
00707
00708     # ifndef _ANSI_H
00709     # include <ansi.h>
00710     # endif
00711
```