同济大学本科教材出版基金资助

Linux操作系统实现原理

赵 炯 编著

同济大学出版社
TONGJI UNIVERSITY PRESS

Linux 操作系统实现原理

赵　炯　编著

同济大学出版社
TONGJI UNIVERSITY PRESS

内 容 提 要

本书依据内核源代码详细描述 Linux 操作系统内核的基本功能和工作原理。首先介绍 Linux 的发展历史，说明内核的演变改进过程，然后概要介绍运行 Linux 操作系统的 PC 硬件的组成结构以及内核所使用的汇编语言和 C 语言扩展部分，着重介绍和说明 80x86 处理器在保护模式下运行的编程方法，接着详细介绍 Linux 内核源代码的程序文件，每个文件的功能和相关的软硬件知识。在最后一章给出实验内容，读者可以由此对 Linux 操作系统内核进行仿真实验，以获取对 Linux 操作系统工作原理的深入理解。

本书已将注释过的 Linux 内核源代码程序放在网上供读者直接下载。这样编排可大大缩减书本篇幅，更适合作为大专院校本科生或研究生学习操作系统和嵌入式系统的教材，同时也可供一般技术人员作为开发嵌入式系统的参考书籍。

图书在版编目(CIP)数据

Linux 操作系统实现原理 / 赵炯编著. -- 上海：同济大学出版社，2018.9

ISBN 978-7-5608-8056-3

Ⅰ.①L… Ⅱ.①赵… Ⅲ.①Linux 操作系统 Ⅳ.①TP316.85

中国版本图书馆 CIP 数据核字(2018)第 174173 号

Linux 操作系统实现原理

赵 炯 **编著**

策划编辑 朱 勇 **责任编辑** 李小敏 **责任校对** 徐春莲 **封面设计** 陈益平

出版发行	同济大学出版社 www.tongjipress.com.cn (地址：上海市四平路 1239 号 邮编：200092 电话：021-65985622)
经 销	全国各地新华书店
排 版	南京月叶图文制作有限公司
印 刷	江苏凤凰数码印务有限公司
开 本	787 mm×1092 mm 1/16
印 张	27.25
字 数	680 000
版 次	2018 年 9 月第 1 版 2018 年 9 月第 1 次印刷
书 号	ISBN 978-7-5608-8056-3
定 价	79.00 元

序 言

在智能制造和控制大趋势下，Linux 操作系统已经成为当今嵌入式系统中最为重要的操作控制基础平台。本书是一本有关 Linux 操作系统内核基本工作原理的入门教材，主要目标是在较少的篇幅内对完整的 Linux 内核源代码进行解剖，以期对操作系统的基本功能和实现方式获得全方位的理解。做到对 Linux 内核有一个完整而深刻的了解，对 Linux 操作系统的基本工作原理真正入门。

本书读者群定位于大专院校高年级学生，或者是一些知晓 Linux 系统一般使用方法并具有一定的编程基础，但比较缺乏阅读目前最新内核源代码的基础知识，又急切希望能够进一步理解 UNIX 类操作系统工作原理和具体代码实现的爱好者。

本书特点

在写作本书时已有的描述 Linux 内核的书籍，均尽量选用最新 Linux 内核版本（例如 Fedora Core 8 使用的 2.6.24 稳定版等）进行描述，但由于目前 Linux 内核整个源代码的大小已经非常得大（例如 2.2.20版就已具有 268 万行代码），因此这些书籍仅能对 Linux 内核源代码进行选择性的或原理性的说明，许多系统实现细节被忽略。因此很难给予读者对实际 Linux 内核有清晰完整的理解。

Scott Maxwell 先生著的《Linux 内核源代码分析》基本上是面对 Linux 中高级水平的读者，需要较为全面的基础知识才能完全理解。而且可能是由于受篇幅所限，该书并没有对所有 Linux 内核代码进行注释，略去了很多内核实现细节，例如其中内核中使用的各个头文件（*.h）、生成内核代码映像文件的工具程序、各个 make 文件的作用和实现等均没有涉及。因此对于处于初中级水平之间的读者来说阅读该书有些困难。

John Lions 先生著的《莱昂氏 UNIX 源代码分析》一书虽然是一本学习 UNIX 类操作系统内核源代码很好的书籍，但是由于其采用的是 UNIX V6 版，其中系统调用等部分代码是用早已废弃的 PDP-11 系列机的汇编语言编制的，因此在阅读和理解与硬件部分相关的源代码时就会遇到较大的困难。

A. S. Tanenbaum 先生的书《操作系统：设计与实现》是一本有关操作系统内核实现很好的入门书籍，但该书所叙述的 MINIX 系统是一种基于消息传递的内核实现机制，与 Linux 内核的实现有所区别。因此在学习该书之后，并不能很顺利地即刻着手进一步学习较新的 Linux 内核源代码实现。

在使用这些书籍进行学习时会有一种“盲人摸象”的感觉，不容易真正理解 Linux 内核系统具体实现的整体概念，尤其是对那些 Linux 系统初学者或刚学会如何使用 Linux 系统的人在使用那些书学习内核原理时，内核的整体运作结构并不能清晰地在脑海中形成。这在本人多年的 Linux 内核学习过程中也深有体会。1991 年 10 月，Linux 的创始人 Linus Torvalds 先生在开发出 Linux 0.03 版后写的一篇文章中也提到了同样的问题。在这篇题为“LINUX-a free unix-386 kernel”[①]的文章中，他说：“开发 Linux 是为了那些操作系统爱好者和计算机科学系的学生使用、学习和娱乐。”而现今流行的 Linux 系统已经变得庞大和复杂，已不适合作为初学者的入门学习起点。这也是作者基于 Linux 早期内核版本写作本书的动机之一。

为了填补这个空缺，本书的主要目标即是使用尽量少的篇幅或在有限的篇幅内，对完整的 Linux 内核源代码进行全面解剖，以期对操作系统的基本功能和实际实现方式获得全方位的理解。做到对 Linux 内核有一个完整而深刻的理解，对 Linux 操作系统的基本工作原理真正理解和入门。

---

① 原文可参见：http://oldlinux.org/Linus/

阅读早期内核的其他好处

目前，已经出现不少基于 Linux 早期内核而开发的专门用于嵌入式系统的内核版本，如 DJJ 的 x86 操作系统、Uclinux 等，世界上也有许多人认识到通过早期 Linux 内核源代码学习的好处，目前国内也已经有人正在组织人力注释出版类似本文的书籍。因此，通过阅读 Linux 早期内核版本的源代码，的确是学习 Linux 系统的一种行之有效的途径，并且对研究和应用 Linux 嵌入式系统也有很大的帮助。

在对早期内核源代码的注释过程中，作者发现，早期内核源代码几乎就是目前所使用的较新内核的一个精简版本，其中已包括了目前新版本中几乎所有的基本原理的内容。正如《系统软件：系统编程导论》一书的作者 Leland L. Beck 先生在介绍系统程序以及操作系统设计时，引入了一种极其简化的简单指令计算机(SIC)系统来说明所有系统程序的设计和实现原理，既避免了实际计算机系统的复杂性，又能透彻地说明问题。这里选择 Linux 的早期内核版本作为学习对象，其指导思想与 Leland 一致。这对 Linux 内核学习的入门者来说，是理想的选择之一，能够在尽可能短的时间内深入理解 Linux 内核的基本工作原理。

对于那些已经比较熟悉内核工作原理的人，为了能让自己在实际工作中对系统的实际运转机制消除空中楼阁的感觉，因此也有必要阅读内核源代码。选用的 Linux 早期内核版本的不足之处在于其不包含对虚拟文件系统 VFS 的支持、对网络系统的支持，仅支持 a.out 执行文件和对其他一些现有内核中复杂子系统的说明。但由于本书是作为 Linux 内核工作机理实现的入门教材，因此这也正是选择早期内核版本的优点之一。通过学习本书，可以为进一步学习这些高级内容打下扎实的基础。

阅读完整源代码的重要性和必要性

虽然通过阅读一些操作系统原理经典书籍(例如 M. J. Bach 的《UNIX 操作系统设计》)能够对 UNIX 类操作系统的工作原理有一些理论上的指导作用，但实际上对操作系统的真正组成和内部关系实现的理解仍不是很清晰。正如 AST 所说的，“许多操作系统教材都是重理论而轻实践”，“多数书籍和课程为调度算法耗费大量的时间和篇幅而完全忽略 I/O，其实，前者通常不足一页代码，而后者往往要占到整个系统三分之一的代码总量”。内核中大量的重要细节均未提到。因此并不能让读者理解一个真正的操作系统实现的奥妙所在。只有在详细阅读过完整的内核源代码之后，才会对系统有一种豁然开朗的感觉，对整个系统的运作过程有深刻的理解。以后再选择最新的或较新内核源代码进行学习时，也不会碰到大问题，基本上都能顺利地理解新代码的内容。

正如 Linux 系统的创始人 Linus 先生在一篇新闻稿上所说的，要理解一个软件系统的真正运行机制，一定要阅读其源代码(RTFSC)。系统本身是一个完整的整体，具有很多看似不重要的细节存在，但是若忽略这些细节，就会对理解整个系统带来困难，并且不能真正了解一个实际系统的实现方法和手段。

如何选择要阅读的内核代码版本

那么，如何选择既能达到上述要求，又不被太多的内容而搞乱头脑，选择一个适合的 Linux 内核版本进行学习，提高学习的效率呢？作者通过对大量内核版本进行比较和选择后，最终选择了与目前 Linux 内核基本功能较为相近，又非常短小的 0.12 版内核作为入门学习的最佳版本。图 0-1 是对一些主要 Linux 内核版本行数的统计。

目前的 Linux 内核源代码量都在数百万行的数量级上，2.6.0 版内核代码行数约为 592 万行，而 4.12.X 版内核代码量则更为庞大，对这些版本进行完全注释和说明几乎不可能。而 0.12 版内核仅有不超过 2 万行代码量，因此可以在一本书中解释和说明清楚。麻雀虽小，五脏俱全。为了对所研究的系统有感性的了解，并能利用实验来加深对原理的理解，作者还专门重建了基于该内核的可运行的 Linux 0.12系统，并且可以使用该系统中的 GNU gcc 编译环境，做一些实验编程和简单的开发工作。

另外，使用该版本也可以避免使用现有较新内核版本中已变得越来越复杂的各子系统部分的研究(如虚拟文件系统 VFS，ext2 或 ext3 文件系统、网络子系统、新的复杂的内存管理机制等)。

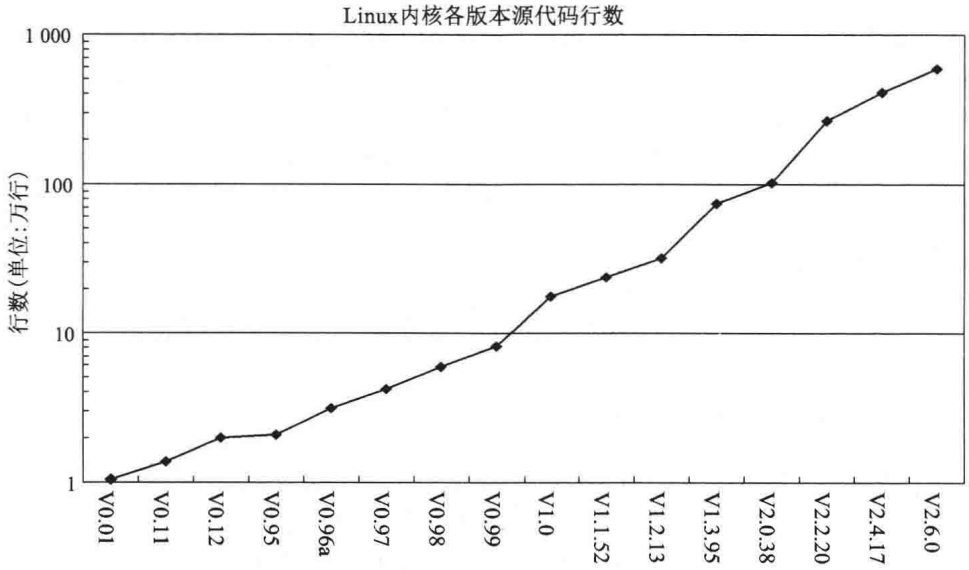

图 0-1 内核版本号

阅读本书需具备的基础知识

在阅读本书时，读者应该具备基本的 C 语言知识和 Intel CPU 汇编语言知识。有关 C 语言最佳的参考资料仍然是 Brain W. Kernighan 和 Dennis M. Ritchie 编写的《The C Programming Language》一书。而汇编语言的资料则可以参考任意一本讲解与 Intel CPU 相关的汇编语言教材。书中也包含有一些嵌入汇编基本的语法说明(第 5.5 节)。

除此之外，还希望读者具备以下一些基础知识或者有相关的参考书籍在身边。其一是有关 80x86 处理器结构和编程的知识或资料，如 80x86 编程手册(INTEL 80386 Programmer's Reference Manual)；其二是有关 80x86 硬件体系结构和接口编程的知识或资料；其三还应具备初级使用 Linux 系统的简单技能。另外，由于 Linux 系统内核实现最早是根据 M. J. Bach 的《UNIX 操作系统设计》一书的基本原理开发的，源代码中许多变量或函数的名称都来自该书，因此在阅读本书时若能适当参考该书，会更易于理解内核源代码。

Linus 在最初开发 Linux 操作系统时，参照了 MINIX 操作系统。例如，最初的 Linux 内核版本完全照搬了 MINIX1.0 文件系统。因此，在阅读本书时，A. S. Tanenbaum 的书《操作系统：设计与实现》也具有较大的参考价值。

使用早期版本是否过时?

表面看来本书对 Linux 早期内核版本注释的内容犹如 Linux 操作系统刚公布时 Tanenbaum 就认为其已经过时的(Linux is obsolete)想法一样，但通过学习本书内容，你就会发现，利用本书学习 Linux 内核，由于内核源代码量短小而精干，因此会有极高的学习效率，能够做到事半功倍，快速入门。并且对继续进一步选择新内核部分源代码的学习打下坚实的基础。在学习完本书之后，你将对系统的运作原理有一个非常完整而实际的概念，这种完整概念能使人很容易地进一步选择和学习新内核源代码中的任何部分，而不需要再去啃读代码量巨大的新内核中完整的源代码。

Ext 文件系统与 MINIX 文件系统

目前 Linux 系统上所使用的 Ext2(或最新的 Ext3)文件系统是在内核 1.x 之后开发的。其功能详尽

并且性能也非常完整和稳固，是目前 Linux 操作系统上默认的标准文件系统。但是，作为对 Linux 操作系统完整工作原理入门学习所使用的部分，原则上是越精简越好。为了达到对一个操作系统有完整的理解，并且能不被其中各子系统中复杂和过多的细节喧宾夺主，在选择学习剖析用的内核版本时，只要系统的部分代码内容能说明实际工作原理，就越简单越好。Linux 内核 0.12 版上仅包含最为简单的 MINIX 1.0 文件系统，对于理解一个操作系统中文件系统的实际组成和工作原理已经足够。这也是选择 Linux 早期内核版本进行学习的主要原因之一。

Linux 内核程序提供形式

为了缩减本书篇幅，现将所有注释过的内核程序均以电子版形式提供给读者。程序下载或在线浏览的网址是

http://oldlinux.org/download/Book-Lite/

除包括 linux 0.12 内核原始源代码外，网上对每个程序都提供有 pdf 和 html 两种格式的文档。表 0-1 是书中程序列表号与内核源代码程序的对照表。上面网站也有此表，同时也提供对应 pdf 和 html 文档的链接。

表 0-1　程序列表标号与内核代码文件对应关系

程序标号	源程序路径名	程序标号	源程序路径名
第 5 章		程序 9-1	linux/kernel/blk_drv/blk.h
程序 5-1	linux/Makefile	程序 9-2	linux/kernel/blk_drv/hd.c
第 6 章		程序 9-3	linux/kernel/blk_drv/ll_rw_blk.c
程序 6-1	linux/boot/bootsect.s	程序 9-4	linux/kernel/blk_drv/ramdisk.c
程序 6-2	linux/boot/setup.S	程序 9-5	linux/kernel/blk_drv/floppy.c
程序 6-3	linux/boot/head.s	第 10 章	
第 7 章		程序 10-1	linux/kernal/chr_drv/keyboard.S
程序 7-1	linux/init/mina.c	程序 10-2	linux/kernal/chr_drv/console.c
第 8 章		程序 10-3	linux/kernal/chr_drv/serial.c
程序 8-1	linux/kernel/asm.s	程序 10-4	linux/kernal/chr_drv/rs_io.s
程序 8-2	linux/kernel/traps.c	程序 10-5	linux/kernal/chr_drv/tty_io.c
程序 8-3	linux/kernel/sys_call.s	程序 10-6	linux/kernal/chr_drv/tty_ioctl.c
程序 8-4	linux/kernel/mktime.c	第 11 章	
程序 8-5	linux/kernel/sched.c	程序 11-1	linux/kernel/math/math-emulate.c
程序 8-6	linux/kernel/signal.c	程序 11-2	linux/kernel/math/error.c
程序 8-7	linux/kernel/exit.c	程序 11-3	linux/kernel/math/ea.c
程序 8-8	linux/kernel/fork.c	程序 11-4	linux/kernel/math/convert.c
程序 8-9	linux/kernel/sys.c	程序 11-5	linux/kernel/math/add.c
程序 8-10	linux/kernel/vsprintf.c	程序 11-6	linux/kernel/math/compare.c
程序 8-11	linux/kernel/printk.c	程序 11-7	linux/kernel/math/get_put.c
程序 8-12	linux/kernel/panic.c	程序 11-8	linux/kernel/math/mul.c
第 9 章		程序 11-9	linux/kernel/math/div.c

（续表）

程序标号	源程序路径名	程序标号	源程序路径名
第 12 章		程序 14-14	linux/include/asm/io. h
程序 12-1	linux/fs/buffer. c	程序 14-15	linux/include/asm/memory. h
程序 12-2	linux/fs/bitmap. c	程序 14-16	linux/include/asm/segment. h
程序 12-3	linux/fs/truncate. c	程序 14-17	linux/include/asm/system. h
程序 12-4	linux/fs/inode. c	程序 14-18	linux/include/linux/config. h
程序 12-5	linux/fs/super. c	程序 14-19	linux/include/linux/fdreg. h
程序 12-6	linux/fs/namei. c	程序 14-20	linux/include/linux/fs. h
程序 12-7	linux/fs/file_table. c	程序 14-21	linux/include/linux/hdreg. h
程序 12-8	linux/fs/block_dev. c	程序 14-22	linux/include/linux/head. h
程序 12-9	linux/fs/file_dev. c	程序 14-23	linux/include/linux/kernel. h
程序 12-10	linux/fs/pipe. c	程序 14-24	linux/include/linux/mm. h
程序 12-11	linux/fs/char_dev. c	程序 14-25	linux/include/linux/sched. h
程序 12-12	linux/fs/read_write. c	程序 14-26	linux/include/linux/sys. h
程序 12-13	linux/fs/open. c	程序 14-27	linux/include/linux/tty. h
程序 12-14	linux/fs/exec. c	程序 14-28	linux/include/sys/param. h
程序 12-15	linux/fs/stat. c	程序 14-29	linux/include/sys/resource. h
程序 12-16	linux/fs/fcntl. c	程序 14-30	linux/include/sys/stat. h
程序 12-17	linux/fs/ioctl. c	程序 14-31	linux/include/sys/time. h
程序 12-18	linux/fs/select. c	程序 14-32	linux/include/sys/times. h
第 13 章		程序 14-33	linux/include/sys/types. h
程序 13-1	linux/mm/memory. c	程序 14-34	linux/include/sys/utsname. h
程序 13-2	linux/mm/page. s	程序 14-35	linux/include/sys/wait. h
程序 13-3	linux/mm/swap. c	第 15 章	
第 14 章		程序 15-1	linux/lib/_exit. c
程序 14-1	linux/include/a. out. h	程序 15-2	linux/lib/close. c
程序 14-2	linux/include/const. h	程序 15-3	linux/lib/ctype. c
程序 14-3	linux/include/ctype. h	程序 15-4	linux/lib/dup. c
程序 14-4	linux/include/errno. h	程序 15-5	linux/lib/errno. c
程序 14-5	linux/include/fcntl. h	程序 15-6	linux/lib/execve. c
程序 14-6	linux/include/signal. h	程序 15-7	linux/lib/malloc. c
程序 14-7	linux/include/stdarg. h	程序 15-8	linux/lib/open. c
程序 14-8	linux/include/stddef. h	程序 15-9	linux/lib/setsid. c
程序 14-9	linux/include/string. h	程序 15-10	linux/lib/string. c
程序 14-10	linux/include/termios. h	程序 15-11	linux/lib/wait. c
程序 14-11	linux/include/time. h	程序 15-12	linux/lib/write. c
程序 14-12	linux/include/unistd. h	第 16 章	
程序 14-13	linux/include/utime. h	程序 16-1	linux/tools/build. c

在完整阅读完本书之后，相信您定会发出这样的感叹："对于 Linux 内核系统，我现在终于入门了！"此时，您应该有十分的把握去进一步学习最新 Linux 内核中各部分的工作原理和过程了。

同济大学

赵炯博士

2018 年 4 月

目 录

第1章

概　述

本章首先回顾 Linux 操作系统诞生、开发和成长的过程，由此理解本书选择 Linux 系统早期版本作为学习对象的一些原因，然后具体说明选择早期 Linux 内核版本进行学习的优点和不足之处以及如何开始进一步学习，最后对各章的内容进行了简要介绍。

1.1　Linux 的诞生和发展

Linux 是 UNIX 操作系统的一种克隆系统。它诞生于 1991 年 10 月 5 日(这是第一次正式向外公布的时间)。此后借助于 Internet，经过全世界各地计算机爱好者的共同努力，现已成为当今世界上使用量最多的一种 UNIX 类操作系统，并且使用数还在迅猛增长。

Linux 操作系统的诞生、发展和成长过程主要依赖于五个方面的重要支柱：UNIX 操作系统、MINIX 操作系统、GNU 计划、POSIX 标准和 Internet 网络。下面根据这五个基本线索来追寻一下 Linux 的开发历程、酝酿过程以及最初的发展经历。首先分别介绍其中的四个基本要素，然后根据 Linux 的创始人 Linus Torvalds 先生从对计算机感兴趣而开始自学计算机知识，到心里酝酿编制一个自己的操作系统，到最初 Linux 内核 0.01 版公布以及从此如何艰难地一步一个脚印在全世界编程爱好者的帮助下最后于 1994 年推出比较完善的 1.0 版本这段时间的发展经过，也即对 Linux 的早期发展历史进行详细介绍。

当然，目前 Linux 内核版本已经开发到了 4.12.x 版本，而大多数 Linux 系统中所使用的比较稳定的 3.x.x 版或 4.x.x 版内核(其中第 2 个数字若是奇数则表示是正在开发的版本，不能保证系统的稳定性)。对于 Linux 的一般发展史，许多文章和书籍都有介绍，这里就不再重复。

1.1.1　UNIX 操作系统的诞生

Linux 是 UNIX 操作系统的一个克隆版本，UNIX 操作系统是美国贝尔实验室 Ken. Thompson 先生和 Dennis Ritchie 先生于 1969 年夏季在 DEC PDP-7 小型计算机上开发的一个分时操作系统。

Ken Thompson 为了能在闲置不用的 PDP-7 计算机上运行他非常喜欢的星际旅行(Space travel)游戏，于是在 1969 年夏天趁夫人回家乡加利福尼亚度假期间，在一个月时间内开发出了 UNIX 操作系统的原型。当时使用的是 BCPL 语言(基本组合编程语言)，后经 Dennis Ritchie 于 1972 年用移植性很强的 C 语言进行了改写，使得 UNIX 系统开始在大专院校得到了推广和广泛传播。

1.1.2　MINIX 操作系统

MINIX 系统是由 Andrew S. Tanenbaum(AST)先生开发的应用于教学的小型操作系统。AST 在荷兰 Amsterdam 的 Vrije 大学数学与计算机科学系统工作，是 ACM 和 IEEE 的资深会员(全世界也只有很少人是两会的资深会员)。共发表了 100 多篇文章，出版了 5 本计算机书籍。

AST 虽出生在美国纽约，但却是荷兰侨民(1914 年他的祖辈来到美国)。他在纽约上的中学、MIT 上的大学、加州大学 Berkeley 分校获得博士学位。在博士后研究工作阶段，他来到了家乡荷兰，从此就与家乡一直有来往。后来就在 Vrije 大学开始教书、带研究生。荷兰首都 Amsterdam 是个常年阴雨绵绵的城市，但对于 AST 来说，这最好不过了，因为在这样的环境下他可以经常待在家中摆弄他的计算机了。

MINIX 是他 1987 年编制成的，主要用于学生学习操作系统原理。到 1991 年时版本是 1.5。目前主要有三个版本在使用：1.5 版、2.0 版和 3.0 版。当时该操作系统可免费在大学内使用，但其他用途则不是。当然现在 MINIX 系统已经是免费的了，可以从许多 FTP 上下载。

对于 Linux 系统，AST 先生后来曾表示对其开发者 Linus 的称赞。但他认为 Linux 的发展很大原因是他为了保持 MINIX 的小型化和易于教学使用，能让学生在一个学期内学完，因而没有接纳全世界许多人对 MINIX 的扩展要求。因此在这样的前提下激发了 Linus 编写 Linux 系统。当然 Linus 也正好抓住了这个好时机。

作为一个操作系统，MINIX 并不是优秀者，但它同时提供了用 C 语言和汇编语言编写的系统源代码。这是第一次使得有抱负的程序员和黑客们(hackers)能够阅读操作系统的源代码。在当时，这种源代码是软件商们一直小心守护着的秘密。

1.1.3 GNU 计划

GNU(是"GNU's Not Unix"的递归缩写，它的发音为"guh-NEW")计划和自由软件基金会 FSF(the Free Software Foundation)是由 Richard M. Stallman 先生于 1984 年一手创办的。旨在开发一个类似 UNIX 并且可以免费使用的完整操作系统：GNU 系统。各种使用 Linux 作为核心的 GNU 操作系统正在被广泛地使用。虽然这些系统通常被称作"Linux"，但是 Stallman 认为，严格地说，它们应该被称为 GNU/Linux 系统。

到 20 世纪 90 年代初，GNU 项目已经开发出许多高质量的免费软件，其中包括有名的 Emacs 编辑系统、bashshell 程序、gcc 系列编译程序、gdb 调试程序等。这些软件为 Linux 操作系统的开发创造了一个合适的环境。这是 Linux 能够诞生的重要基础之一，以至于目前许多人都将 Linux 操作系统称为"GNU/Linux"操作系统。

1.1.4 POSIX 标准

计算机可移植操作系统接口标准 POSIX(Portable Operating System Interface for Computing Systems)是由 IEEE 和 ISO/IEC 开发的一簇标准。该标准是基于现有 UNIX 操作系统的应用实践和经验，描述了操作系统的调用服务接口，用于保证编制的应用程序可以在源代码一级上在多种操作系统上移植和运行。它是在 1980 年早期一个 UNIX 用户组(usr/group)的早期工作基础上取得的。该 UNIX 用户组原来试图将 AT&T 的 System V 操作系统和 Berkeley 分校的 BSD 操作系统调用接口之间的区别重新调和集成。并于 1984 年定制出了/usr/group 标准。

1985 年，IEEE 操作系统技术委员会标准小组委员会(TCOS-SS)开始在 ANSI 的支持下责成 IEEE 标准委员会制定有关程序源代码可移植性操作系统服务接口正式标准。到 1986 年 4 月，IEEE 制定出了试用标准。第一个正式标准是在 1988 年 9 月份批准的(IEEE 1003.1—1988)，也即以后经常提到的 POSIX.1 标准。到 1989 年，POSIX 的工作被转移至 ISO/IEC 社团，并由 15 个工作组继续将其制定成 ISO 标准。到 1990 年，POSIX.1 与已经通过的 C 语言标准联合，正式批准为 IEEE 1003.1—1990(也是 ANSI 标准)和 ISO/IEC 9945—1:1990 标准。

POSIX.1 仅规定了系统服务应用程序编程接口(API)，概括了基本的系统服务标准。因此工作组期望对系统的其他功能也制定出标准。这样 IEEE POSIX 的工作就开始展开了。刚开始有十个批准的计划在进行，有近 300 多人参加每季度为期一周的会议。着手的工作有命令与工具标准(POSIX.2)、测试方法标准(POSIX.3)、实时 API(POSIX.4)等。到了 1990 年上半年已经有 25 个计划在进行，并且有 16 个工作组参与了进来。与此同时，还有一些组织也在制定类似的有与操作系统有关的标准，如 X/Open，AT&T，OSF 等公司和组织。

1991—1993 年，POSIX 标准的制定正在最后投票阶段，那时 Linux 正刚起步。这个 UNIX 标准为 Linux 提供了极为重要的信息，它使得 Linux 能够在标准的指导下进行开发，并能够与绝大多数 UNIX 类操作系统兼容。在最初的 Linux 内核源代码中(0.01 版、0.11 版和 0.12 版)就已经为与 POSIX 标准的兼容性方面做好了准备工作。在 Linux 0.01 版内核的/include/unistd.h 文件中就已定义了几个有关 POSXI 标准要求的符号常数，而且 Linus 先生在程序注释中已写道："OK，这也许是个玩笑，但我正在着手研究它呢"。1991 年 7 月 3 日 Linus 在 comp.os.minix 上发布的消息中就已经提到了正在搜集 POSIX

的资料。其中透露了他正在着手一个操作系统的开发，并且在开发之初已经想到要实现与 POSIX 相兼容的问题了。

1.1.5 Linux 操作系统的诞生

1981 年，IBM 公司推出了享誉全球的微型计算机 IBM PC 机。在 1981—1991 年的十年间，MS-DOS 操作系统一直是微型计算机操作系统的主宰。此时计算机硬件价格虽然逐年下降，但软件价格仍然居高不下。当时 Apple 公司的 MACs 操作系统可以说是性能最好的，但是其天价使得没人能够轻易靠近。

当时的另一个计算机技术阵营就是 UNIX 世界。但是 UNIX 操作系统就不仅是价格昂贵的问题了。为了寻求高利润率，UNIX 经销商们把价格抬得极高，PC 小用户对其根本是望尘莫及。曾经一度得到贝尔实验室许可而能在大学中用于教学的 UNIX 源代码也一直被小心地守护着不许公开。对于广大的 PC 用户，软件行业的大型供应商们始终没有给出有效解决这个问题的方案。

正在此时(20 世纪 80 年代末)，出现了 MINIX 操作系统，并且有一本描述其设计实现原理的书同时发行。由于 AST 先生的这本书写得非常详细，并且叙述得有条有理，于是几乎全世界的计算机初学者和爱好者都开始看这本书，以期能理解操作系统的工作原理。其中也包括 Linux 系统的创始者 Linus Benedict Torvalds 先生。

当时(1991 年)，Linus Benedict Torvalds 仅是赫尔辛基大学计算机科学系的一名二年级学生，是一位善于自学的计算机黑客。这个 21 岁的芬兰年轻人喜欢鼓捣自己的计算机，测试计算机的性能和限制。但当时他所缺乏的就是一个专业级的操作系统。

在同一年间，GNU 计划已经开发出许多工具软件。其中最受期盼的 GNU C 编译器已经出现，但还没有开发出免费的 GNU 操作系统。即使是教学使用的 MINIX 操作系统也开始有了版权，需要购买才能得到源代码。虽然 GNU 的操作系统 HURD 一直在开发之中，但在当时看来不可能在几年内完成。

为了能更好地学习计算机知识，Linus 使用圣诞节的压岁钱和贷款购买了一台 386 兼容电脑，并从美国邮购了一套 MINIX 系统软件。就在等待 MINIX 软件期间，Linus 认真学习了有关 Intel 80386 的硬件知识。为了能通过调制解调器(Modem)拨号连接登录到学校的主机上，他使用汇编语言和 80386 CPU 的多任务特性编制出一个终端仿真程序。此后为了将自己一台老式电脑上的软件复制到新电脑上，他还为软盘驱动器、键盘等硬件设备编制出相应的驱动程序。通过这些编程实践，并在学习过程中认识到 MINIX 系统的诸多限制(MINIX 虽然很好，但只是一个用于教学目的的简单操作系统，而非一个强有力的实用操作系统)，Linus 已经有了一些类似于操作系统硬件设备驱动程序的代码，于是在他脑海中逐步有了编制一个新操作系统的想法。此时 GNU 计划已经开发出许多工具软件，其中最受期盼的 GNU C 编译器已经出现。虽然 GNU 的免费操作系统 HURD 正在开发中，但 Linus 已经等不及了。

从 1991 年 4 月起，通过修改终端仿真程序和硬件驱动程序，他开始编制起自己的操作系统来。刚开始，他的目的很简单，只是为了学习 Intel 386 保护模式运行方式下的编程技术，但后来 Linux 的发展却完全改变了初衷。根据 Linus 在 comp. os. minix 新闻组发布的消息，我们可以知道他逐步从学习 MINIX 系统阶段发展到开发自己的 Linux 系统的过程。

Linus 第 1 次向 comp. os. minix 投递消息是在 1991 年 3 月 29 日。所发帖子的题目是“gcc on minix-386 doesn't optimize”，它是有关 gcc 编译器在 MINIX-386 系统上运行优化的问题(MINIX-386 是一个由 Bruce Evans 先生改进的利用 Intel 386 特性的 32 位的 MINIX 操作系统)。由此可知，Linus 在 1991 年初就已经开始深入研究 MINIX 系统，并在这段时间开始有了改进 MINIX 操作系统的思想。在进一步学习 MINIX 系统之后，该想法逐步演变成重新设计一个基于 Intel 80386 体系结构的新操作系统的构思。

他在回答 MINIX 上某人提出的一个问题时，所说的第一句话就是“阅读源代码”(RTFSC)。他认为答案就在源程序中。这也说明了对于学习系统软件来说，我们不光需要懂得系统的工作原理，还需要结合实际系统，学习系统的实现方法。因为理论毕竟是理论，其中省略了许多枝节，而这些枝节问题虽然没有太多的理论含量，但却是一个系统必要的组成部分，就像麻雀身上的一根羽毛。

从 1991 年 4 月开始，Linus 几乎全身心地投入到研究 MINIX-386 系统中，并尝试着移植 GNU 的软