# 基于目标码的XENIX核心源程序提取*

孙玉方 吴 健 李有志 牛光远 梁志辉

(中国科学院软件研究所,北京 100080)

**摘要** 近年来,基于Intel 80386/80486 CPU的计算机,特别是微型机,在国内大量引进并逐渐国产化.XENIX操作系统作为这些计算机上配备的基本多用户系统已被越来越多的国内用户所接受.但是要充分发挥XENIX系统的作用,还需要对此系统作一些改造、扩充或进一步的开发以适应国内普通用户的需要.为了适应国内对XENIX移植开发的需要,经过几年的努力,我们成功地以可运行的目标码为基础,在计算机辅助之下提取了其核心部分的源程序.本文讲述基于目标码的核心源程序提取的方法、有关工具的使用及具体的提取过程.

**关键词** 目标码,XENIX操作系统,核心,计算机辅助.

这些年国内引进或自行生产了大量计算机,特别是微型机.这些计算机上配备的基本多用户系统是XENIX操作系统.为了普及推广这些计算机的应用,需要对此系统作一些改造、扩充或进一步的开发以适应国内普通用户的需要,真正发挥计算机的作用.这些改造扩充包括增加汉字处理能力,增加专用的外部设备驱动程序等等.但是随机带来的只是XENIX系统可运行的机器码(目标码),因此即使系统提供了反汇编程序,可以得到在功能上与源码系统等同的汇编语言程序,但修改和扩充却十分困难.原因是,此时得到的反汇编代码虽然形式上是一条条汇编指令,但实质上是不可以直接修改并重新汇编的.此时的反汇编工具,比如adb,提供了对反汇编程序修改的手段,但真正修改时打入系统的仅仅是机器码,虽然显示出来的为汇编指令形式,这与DOS系统完全不同.

为了适应国内对XENIX移植开发的需要,我们重点研究了以可运行的目标码作为基础,在计算机辅助之下提取其核心源程序问题.经过几年的努力,取得了成功.下面讲述程序分析方法、有关工具的使用及具体的提取过程.

## 1. 程序分析

程序分析包括功能分析、结构分析、数据流分析和接口关系分析,分析的方法主要采用

* 本文1991年3月7日收到,1991年8月30日定稿

作者孙玉方,47岁,研究员,主要研究领域为计算机软件,体系结构,中文信息处理.吴健,工程师,主要研究领域为计算机软件,中文信息处理.李有志,工程师,主要研究领域为计算机软件,中文信息处理.牛光远,工程师,主要研究领域为计算机软件.梁志辉,工程师,主要研究领域为计算机软件.

本文通讯联系人:孙玉方,北京 100080,中国科学院软件研究所

的是静态模拟和动态追踪或者两者结合的手法.

### 1.1 功能分析

从宏观上来说,一个软件包的功能可以用几句话来概括.不过这种概括对我们所说的功能分析并没有十分重要的意义.我们所需要的是较详尽的准确的从总的到每一子系统或部分的功能说明.比如,若把整个 XENIX 核心系统作为一个整体,则必须知道它具有什么样的功能,有几个子系统,各有什么功能.如有可能还可以逐渐细化到掌握模块(文件)甚至有关函数的功能.举例来说,XENIX 核心其主要功能就是进行系统硬件、软件资源的管理并且为 shell 和外层软件提供服务.其中主要有五个子系统,其功能分别是:(1)存储管理部分,实行对内存及硬盘的一部分进行实际资源的管理,并提供虚实存储空间的转换处理;(2)CPU 和进程管理部分,进行进程控制、进程调度和进程通信,有效地利用 CPU 资源;(3)设备管理部分,主要对各种外设实施有效的控制,提供输入/输出服务;(4)文件管理部分,主要对用户和系统的信息(以文件形式存放在系统中)进行安全、可靠的储存和检索处理;(5)中断管理部分,完成硬件输入/输出或例外事件的处理,也通过它完成用户态到核心态的转接.

了解整个系统的功能的办法首先是通过使用,对于某个子系统的了解更是如此.比如,要深入了解外设输入/输出的功能就要通过终端输入、打印输出以及各种屏幕操作的控制来了解.不过如要更为深入地了解就要具体分析其代码了.内部分析代码与外部使用相结合将能深入地了解其功能,而对功能的透彻了解将为以后的提取打下基础.

### 1.2 结构分析

系统结构反映在系统中各子系统各模块之间的相互关系上,对于重点模块还应当深入了解其内部结构.

核心层从功能上讲可以分成五大部分,但从结构上讲这几大部分之间并不是相互独立的成分,而是互相有调用关系.构成整个核心的程序约有一百七十个模块,可以分属于上述五大部分.每一模块(在 UNIX 系统中称文件)又往往可以分成若干函数.而按 UNIX 核心的构成,是无序模块结构.这些模块的函数之间互有调用关系,无法分成层次或构成树状关系.

但是就某一个部分,比如,块设备输入/输出子系统,如果相对于其它模块独立来看,其主要部分的结构还是比较清晰的,基本上呈现一种层次结构.

结构关系清晰了解有助于各模块功能和接口的了解,当然有助于源程序提取和进一步的开发.

### 1.3 接口分析

接口关系包括各模块和函数之间的调用关系以及数据交换格式.

核心层的程序是在核心态下运行,其余层是在用户态下运行.除核心层外,其余层的程序在操作时总有某种情况下会转入核心层,如请求核心层完成某些输入/输出服务,或程序执行时来了正常中断(如输入/输出完成)也可能是系统发生了故障而导致中断,这些都要转入核心去处理.外层程序转入核心态通常是通过系统调用,比如,读文件时先用系统调用 open 打开文件,然后用系统调用 read 读所需的文件.系统调用就是一种接口.但是一旦进入核心,就不能再使用系统调用了.在核心内部各模块(函数)之间通过调用来完成不同的功能,这时的接口关系是函数调用关系.

模块各部分之间特别是函数调用时都有一定的数据格式要求,简要说起来就是输入参数和返回值.由于系统各部分关系错综复杂,函数调用的层次嵌套很深,又有各种例外情况的处理,所以了解这部分特别困难.

### 1.4 数据流分析

从本质上讲目前的计算机主要就是处理数据的,当然这里所讲的数据是宏观意义上的说法,包括程序指令、普通数据、状态及正文等.

比如,当从键盘上输入时,键入的代码进入系统,在系统内部得以正确流通,并进行内外存交换,必要时又可送出去进行显示或打印.这就是正文入/出的基本数据流程.

除了系统整体的数据流向需要搞清楚以外,在具体实现还必须了解有关的模块间的数据流向.

上面所讲的几种分析工作是相互关联的,孤立地完成某一种工作是不可能至少是很不完善和彻底的.

### 1.5 分析方法

由于我们的分析工作都是基于目标码系统的,所以掌握正确的分析方法至关重要.分析方法主要分两类:静态模拟和动态追踪.

对于数据流及数据格式变换及复杂的程序调用关系的了解光靠静态模拟往往是不够的甚至是极其困难的.此时我们就采用动态跟踪模拟的方法.许多 UNIX(XENIX) 版本中,adb 程序提供了跟踪手段,可以通过它设置断点,运行被分析程序,在断点处查看程序控制流向以及数据变换情况以判定问题之所在.

但这种追踪手段对于核心无效.核心程序虽然从本质上讲也是一种程序,但是由于它要求的环境与用户态程序不一样,所以不能通过跟踪进行动态模拟.此时唯一的办法就只能是静态分析了.

## 2 分析工具及结果

为了更有效地分析和提取核心目标码的结构,我们开发了一些分析和提取工具,其中最主要的是目标码结构分析程序 corestract 和调用关系分析程序 calllist.利用自行开发的工具及系统提供的一些工具,我们对核心的结构以及接口和调用关系进行了分析.

### 2.1 XENIX 核心结构

1. make 与系统系统总体组成

make 是 XENIX(UNIX)系统中极为有效的程序生成和维护的工具.make 从用户定义的 makefile 中读入命令,自动创建和生成中大型程序.

makefile 中列出了要创建的文件,创建每个文件所需的命令以及根据那些文件来创建指定的文件.当用 make 创建程序时,它检查程序所依赖的文件是不是最新的版本,然后执行给定的命令以创建程序.若一个文件不是最新版本,则 make 在创建指定的程序前先更新此文件.make 通过执行显式命令或内部命令来更新程序.

根据 makefile,我们分析得到 XENIX 核心结构组成,见图 1.

这里,CONF 和 CFG 是与核心有关几个模块(.c 文件),H 是系统使用的全部变量文

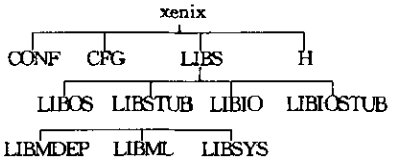


图1 XENIX 核心总体组成

件,而其余的 LIB××则是构成 XENIX 核心的基本程序所组成的库,共有五个.

2. 系统库模块

ar 和 ranlib 是库维护工具,它们可把有用的 C 和汇编语言组成的函数及程序组织成库形式. 其中,ar 是归档程序,可用来把可重定位的目标文件建成库. ranlib 是随机库生成程序,可把归档库转换成随机库,并把相应的库目录表放在各个库的前面.

图 1 中的 LIB××都是以库形式存放的可重定位目标文件. 为了进一步了解其结构,我们利用 ar"打开"这个库从而得到了各个库中的目标模块(.o 文件). 这样,我们就得到了 xenix 核心的结构组成(表 1).

表 1 XENIX 核心组成模块

| CONF | CFG | H | LIBIO | LIBIOSTUB | LIBOS | | | 总计模块数 |
|---|---|---|---|---|---|---|---|---|
| | | | | | LIBMDEP | LIBML | LIBSYS | |
| 5 | 2 | 103 | 50 | 23 | 39 | 14 | 38 | 274 |

整个 XENIX 的核心是由 103 个".h"全局变量文件、171 个".c"和".s"文件(在 XENIX 系统中,一个文件即是一个模块)组成的.

## 2.2 接口及调用关系

1. XENIX 核心系统的目标系统 xenix 结构

在 XENIX 系统中每一个完整的程序的可执行代码(XENIX 的核心可执行代码 xenix 也是如此)的结构主要分成文件头、正文、常量(初始化)数据和非初始化数据、符号表等,这可以从全局变量文件 a.out.h 中了解.

使用 codestruct 工具我们就知道一个程序的正文、数据及符号表的长度,程序的入口地址,字节/字的次序,重定位信息等等. 具体结构见图 2.

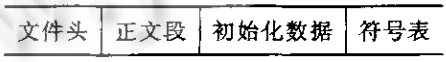


图 2 目标文件 xenix 的组成

2. 变量及函数调用关系

利用 nm 工具可以得到 xenix 中全部函数名和变量名. 进一步可利用 grep 过滤程序把变量名和函数名分离出来. 还可以用 sort 对变量和函数排序,并且可得到其单元地址和入口地址(相对地址).

利用 calllist 工具可以列出函数的调用关系,参照由 ar 列出的".o"文件,即可得到整个系统核心的所有函数的调用关系表及模块的接口关系.

## 3　源程序提取

明确了函数的调用关系及变量意义之后，下一步就是提取和还原源程序.

### 3.1　函数的结构及源码改写

经过分析我们明确了函数的通常结构分三部分：前部为参数传递和信息保留，然后是函数执行部分，最后是恢复及返回. 由于第一和第三部分几乎是规则的，所以重点是放在函数的执行部分. 而函数的执行部分主要是由语句组成的.

C 语言语句中，控制语句的结构比较复杂，其余赋值、比较等语句结构比较简单，这样我们又把重点放在控制语句上. 而控制语句中 switch 语句结构最为清晰，其次是 if 语句，循环语句 for、while 和 do－while. do－while 用得很少，所以我们重点放在 for 和 while 上. for 和 while 的结构是可以可换的，没有本质区别. 在 for、while 语句中基本的又是赋值语句、if 语句等. 这样逐步求精就可以把一个函数的结构具体到语句上. 好在 C 语言的风格是编写小函数，多个函数组合成一个模块而不是一个模块只含一两个大函数，因此这种分析方法不多几步就可穷尽. 为此，我们设计了一些工具判别和分离相应的语句. 在此基础上我们用人工参与的方法改写成 C 语句.

对于表 1 中所列的 LIBIO 和 LIBSYS 中的许多模块在早期的源程序中有类似的结构，这样就大大地便于我们分析、比较和改写.

某些函数和模块比较复杂，我们通过结构和数据流的反复分析才得到正确的结果. 必要时甚至把提取的模块修改到其汇编语句与原系统的完全一致.

### 3.2　系统的还原

系统原来有生成系统的库(见表 1)，而且可以从中得到一个个可重位的目标“.o”模块. 我们在提取源码时就充分利用这一重要特性.

具体做法是，先提取一个模块的源码，然后部分编译，解决显然的句法错误和矛盾之处. 然后把此基本正确的“.o”模块替代库中原有的目标模块，与其它“.o”模块连接成一个系统，然后测试. 首先保证系统功能的正确. 如有错误，则修改提取的模块，再编译并连接成整个系统再测试.

在一个模块提取并测试后再用另外提取的模块替换系统中对应的模块. 再测试，这样一个一个地替换，直至整个库的模块都得到源码. 然后提取其它库中的模块. 这样，一个个模块，一个个库地解决，最终得到一个完整的系统源程序. 最后的工作就是以提取的源程序重新生成一个系统，对此系统进行测试，以保证提取的代码的正确性.

### 3.3　测试和更新

一个软件的功能是否达到了预定的目标，是否有问题都需要经过测试. 对于从目标码上进行改造和扩充而得到的系统来说更是如此. 这种测试可分以下几个步骤：

* 静态检查. 这主要是检查接口的正确性. 一般来说这个工作在开发期间而不是系统完成后再进行.

* 通过特定的模拟和测试程序进行检查. 这个工具是在我们多年开发工作的基础上逐渐形成的. 它可以比较明确地对接口及修改系统进行检查，以确定所加成分的正确性及对系

统其它部分的影响.

* 及时修改和更新. 一旦发现可疑之点,在确定之后即可进行修改,然后再行测试. 另外我们还特别注意版本之间的兼容性及版本的更新. 在不少情况下所出现的问题主要来自于版本的不兼容和硬件之间的差异.

在1988年底我们得到了AT&T UNIX System V R3.1之后,我们参照这一版本核心源码对我们提取的XENIX核心源码进行了进一步比较、改写和调试. 随着XENIX版本的更新及扩充,我们也不断地更新和扩充提取的源码.

系统核心源码的提取工作前后经历了三年(包括版本更新). 因为核心系统运行关系极为复杂,特别牵涉到并发处理等,所以有些隐患可能在很长时间里都发现不了. 由于我们有一批科研人员在UNIX(XENIX)上已有多年的工作经验,一直在维护这一系统,因此隐患一旦发现还是比较容易排除的.

至于系统的合成,我们采用的是make工具,这里涉及库的维护,库中模块的加入,模块的部分编译,连接装配等,关键是编写makefile文件.

**结束语**:我们提取的整个系统包括:".h"文件13 000行,".c"文件31 400行,".s"文件3 200行左右. 生成的可运行核心程序约320KB.

以后我们在XENIX系统核心上的扩充、修改都是以我们提取的源码为基础进行的. 比如,瞬间取时实时响应功能的扩充,I/O多用户智能卡驱动程序的开发与连接,图形处理子系统的扩充和移植,主控监视器中文输入/输出(显示)子系统的扩充等等.

从目标码提取源码的过程是分析、研究、开发的过程. 在这一过程中,我们不仅进一步掌握了XENIX核心的结构、算法和组成,而且深入了解了库的维护、构造,系统的合成、连接的全过程. 现在系统核心的扩充、修改和移植对我们来说是较为容易的事情. 这些技术、方法和经验不但足以应付系统的开发和扩充,而且为系统软件国产化提供了极其重要的技术条件. 提取的源程序在我们完成国家和科学院的攻关课题,以及为国内外的公司开发系统时提供了良好的基础. 这对今后国产化系统软件的开发也有相当的参考价值.

## 参考文献

1 孙玉方. 基于目标码的软件改造与扩充. 计算机用户,1988;(21).

2 刘日升,孙玉方. UNIX操作系统原理与应用. 北京:能源出版社,1987.

# EXSTRACT THE SOURCE CODE FROM THE XENIX KERNEL OBJECT CODE

Sun Yufang, Wu Jian, Li Youzhi, Niu Guangyuan and Liang Zhihui

(*The Institute of Software, The Chinese Academy of Sciences, Beijing* 100080)

**Abstract** Most of Microcomputers based on Intel 80386/486 CPU can run the XENIX Operating System. But the XENIX kernel have to be modified, expanded, and exploited to

meet the need of users in China. Even since 1979, the authors study and develop UNIX and UNIX－like systems, and have more experiences. The authors have exstracted the XENIX kernel source code from the object code, by the computer—aided tools. In this paper, the exstraction method, the tools and the details for exstraction procedure are described.

**Key words** Object code, XENIX operating system, kernel, computer—aided.