第26卷 第6期 电子科技大学学报 Vol.26 No.6
1997年12月 Journal of UEST of China Dec. 1997

UNIX中对多种文件系统类型的支持*

林晓东** 刘心松
（电子科技大学计算机系 成都 610054）

【摘要】 介绍了UNIX下通过虚拟文件系统接口实现多文件系统类型支持的技术，分析了AIX操作系统中虚拟文件系统接口的实现。在此基础上对支持用户实现自己的文件系统存在的问题与困难进行了分析与研究，提出了一种较方便实现新的文件系统的方法。该方法利用了已有文件系统的现成代码，大大减少了开发工作的难度与工作量。

关 键 词 虚拟文件系统； 对象； 抽象； Vnode/Vfs接口； 虚拟I节点
中图分类号 TP316

早期的UNIX虽然开发了性能很好的文件系统S5FS、FFS[1]，适合通常的分时应用，但是它们不能满足下列情形的要求：(1) 需要较好事务支持的数据库环境。(2) 基于局部性的顺序、只读存储环境。(3) 对非UNIX文件系统(FS)支持的需要。(4) 网络上对机器间文件共享的需要，分布式文件系统的开发。

所有这些要求UNIX改变自身的结构以支持多个FS类型，因而出现了AT &T的的文件系统开关结构，SUN Microsystem的Vnode/Vfs接口[2]，DEC的Gnode结构[3]等技术。最后AT &T集成SUN的Vnode/Vfs和NFS技术到SVR4中，使Vnode/Vfs接口成为事实上的多文件系统支持标准。

本文分析了IBM的AIX4.1.1中的Vnode/Vfs实现及对新增加的文件系统支持的实现研究。

1 Vnode/Vfs接口

Vnode/Vfs接口设计应达到下列目的：同时支持多个FS类型；不同的磁盘分区可以包含不同的文件系统类型，但对用户是透明的；完全支持通过网络的文件共享；开发者应能够开发自己的文件系统类型并以模块化的方法加到核心。其主要任务就是在核心提供一个文件访问与管理框架以及核心其他模块与FS之间的统一接口。

Vnode/Vfs的思想来源于UNIX系统中的设备开关表，原先的UNIX中虽然只有一个FS类型，但却有多种文件类型，为了支持对它的访问和管理，定义了一设备开关表结构[1]，核心维护着此结构一个的数组，每种类型的文件各占一项。

利用面向对象的技术，同样的原理被现代UNIX用于实现支持多个FS类型的Vnode/Vfs接口[4]。Vnode/Vfs接口中，Vnode被抽象成代表核心的活动文件，Vfs代表文件系统，两者均被定义成虚基类，由它们可以派生出实现各种FS的子类。图1说明了Vnode类的结构，其数据域包含了

1997年12月11日收稿，1997年3月21日修改定稿
* 国家智能计算机研究开发中心基金资助
** 男 28岁 博士生

独立于 FS 类型的信息,成员函数被分成两类,一类是定义依赖于 FS 类型的虚函数集;另一类定义被别的核心模块调用来管理文件的例程,最终调用依赖于 FS 类型的函数执行底层操作。

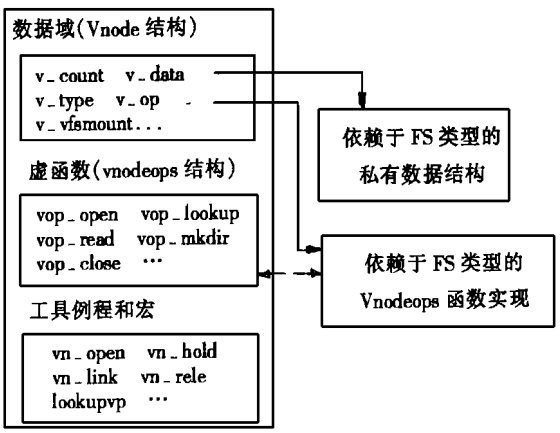

图 1　Vnode 类抽象

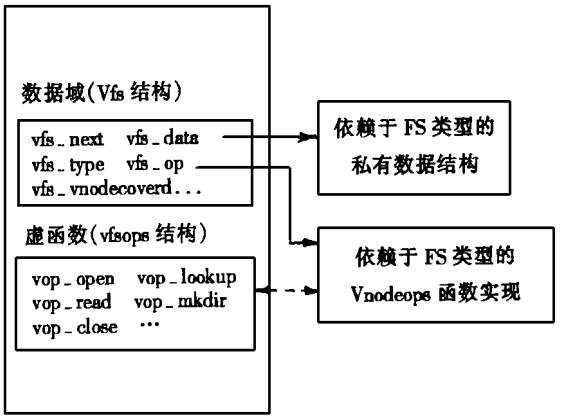

图 2　VFS 类抽象

Vnode 虚基类中有两个字段与实现的子类有关,一个是指向包含特定 FS 类型的私有数据结构的指针 v_data,在传统的 UNIX 文件系统中即是 inode 结构;另一个是指向包含 Vnode 操作函数集的结构指针 v_op,当独立于 FS 的代码调用任意 Vnode 的虚函数时,核心都引用 v_op 指针调用相对应的 FS 函数。如核心调用虚函数 VOP_OPEN,实际执行时被翻译成下列代码:

#define VOP_OPEN(vp, …)(*(vp)->v_op->vop_open))(vp, …)

同样,Vfs 虚基类也有两个字段分别指向依赖于文件系统的数据和操作函数集,参看图 2。

为了正确访问某文件系统对应的 Vnode/Vfs 操作,核心通过使用一种 Vfs 开关的机制来访问每类文件系统的接口函数,这是一个每类文件系统都有一条对应记录的全局表或数组。

2　AIX 中 Vnode/Vfs 接口的实现

2.1　AIX 的 FS 体系结构

AIX 文件系统主要有三部分[5]:

逻辑文件系统(LFS):为用户提供对各种文件操作的统一接口,对核心提供一致的 FS 外观。

物理文件系统(PFS):各种具体 FS 实现,如 UFS、NFS、FFS 等。

Vnode/Vfs 接口:通过对 PFS 的抽象,提供了支持对多个 FS 实现的并发访问的一致接口;允许 LFS 不区分访问的 FS 类型;它可以是物理的/逻辑的、远程/局部的或不附属存贮设备的 FS。

2.2　接口中的主要对象

2.2.1　虚拟节点 Vnode——包含 Vfs 中一个对象的信息

Vnode 代表通向的一个 Vfs 对象的道路,这是寻径的结果,它使每个 Vfs 知道的路径名与一个 Vfs 对象联系,当用路径名访问文件时,将被创建或再次使用(定位文件)。它是 OS 引用文件的主要手段,没有关于所代表文件的数据存贮信息。由核心服务 vn_get 创建,vn_free 释放,通过 lookupvp 核心服务获得某文件的 Vnode,基于 Vnode 的操作被定义为 vnodeops 结构。

2.2.2　通用节点 Gnode——包含 PFS 中一个对象的信息

gnode 代表 FS 实现中的对象,与其是一对一关系,并直接指向一个文件。实现时常被镶入到内存 inode 中,根据需要由 FS 特定的代码创建(vn-get)。当所指文件的内存 inode 除去时,由核心除去。它被用作 LFS 与 PFS 之间的接口,在 gn-type, gn-ops 域提供有用的 PFS 信息给 LFS,是

PFS 中唯一标识某对象的结构。

2.2.3 虚拟文件系统 Vfs——包含安装系统中的单个FS 实例的信息

当前安装到系统中的每个FS 实例都有一个Vfs 结构对象,在安装时被 vmount 创建,下载时被 uvmount 删除,它是载入的 FS 的中心,提供对所属某 Vnode 的访问途径、FS 的安装信息和回到 gfs 的方法,基于 Vfs 主要操作函数被定义为结构 vfsops:

```
struct vfsops{
    int vfs_mount(vfsp, crp);
    int vfs_unmount(vfsp, flags, crp);
    int vfs_root(vfsp, vpp, crp);
    …          }
```

2.2.4 Gfs——包含一类 FS 实现的信息

AIX 核心有一个记录各种 FS 信息的全局结构数组,完成 Vfs 开关功能,其定义如下:

```
struct gfs{
    struct vfsops  *go_ops;/ *Vfs 接口函数入口表指针  */
    struct vnodeops  *gn_ops;/ *Vnode 接口函数入口表指针 */
    int gn_type;
    char gfs_name[16] ;
    int( *gfs_init)(); / *初始例程函数 */
    ……          }
```

每个系统中实现的 FS 类型都有一个 gfs 结构记录,包括名称,标识号,Vfs/Vnode 接口函数入口表指针,初始化例程指针等。当配置一个新的文件系统类型到核心时,调用 gfsdd 核心服务插入一个 gfs 项,调用 gfsdel 删除,每个 gfs 可对应多个 Vfs 项,所有的 Vfs 在核心构成一个表头由核心全局变量 ROOTVFS 指向的链表,Vfs 接口函数表指针被存在 gfs 结构中,因而 Vfs 结构中无 Vfs_op 域,而是一个指向 gfs 结构的指针。Vnode 被定义成一个双向链表,它与所属的 Vfs 对象互相指向。AIX 中 v_data 字段被通用 inode 的 gnode 所代替,在实现时被镶嵌到内存 inode 中。各个对象间的关系如图 3 所示。

2.3 文件系统的创建与安装的实现

为了支持多 FS 类型,很多 FS 命令没有包括同单个 FS 通信的代码,而是由命令收集不依赖于特定 FS 的参数,FS 名,及其他信息,然后传递给一个 back—end 程序处理,此程序被称为 HELPER。它懂得相关 FS 的特定信息,完成与 FS 通信的具体工作。

AIX 中 FS 的创建通过 mkfs 命令调用一个 FS HELPER 用户空间程序,由此程序实现在介质上创建 FS。FS 的安装通过 mount 命令调用安装 HELPER 程序。由其调用 vmount 系统调用来完成,vmount 首先调用 lookunnp()获得安装点的 Vnode 并检查没有别的 FS 安装在上面,接着搜索 gfs 结构数组中与待安装的FS 类型相对应的记录,激发它指向的gfs_init()初始化函数完成必需的数据结构和资源分配,而后分配一个新的 Vfs 结构并进行下列初始化工作:(1) 把新 Vfs 结构加到由 ROOTVFS 指向的链上;(2) 用 gfs 中的vfsops 指针设置 Vfs_op 指针;(3) 用安装点的 Vnode 指针设置 Vfs_mntdover 指针。

核心接着在安装点Vnode 的v_mvfsp 域存放此 Vfs 结构的地址,最后调用 Vfs_MOUNT 执行依赖于FS 的处理。除了检查权限和初始化工作外,对于一个本地文件系统,典型的操作就是从磁盘上读取FS 结构数据建立 FS 内存映象;对于分布式文件系统则是发出一个安装请求给服务节点。

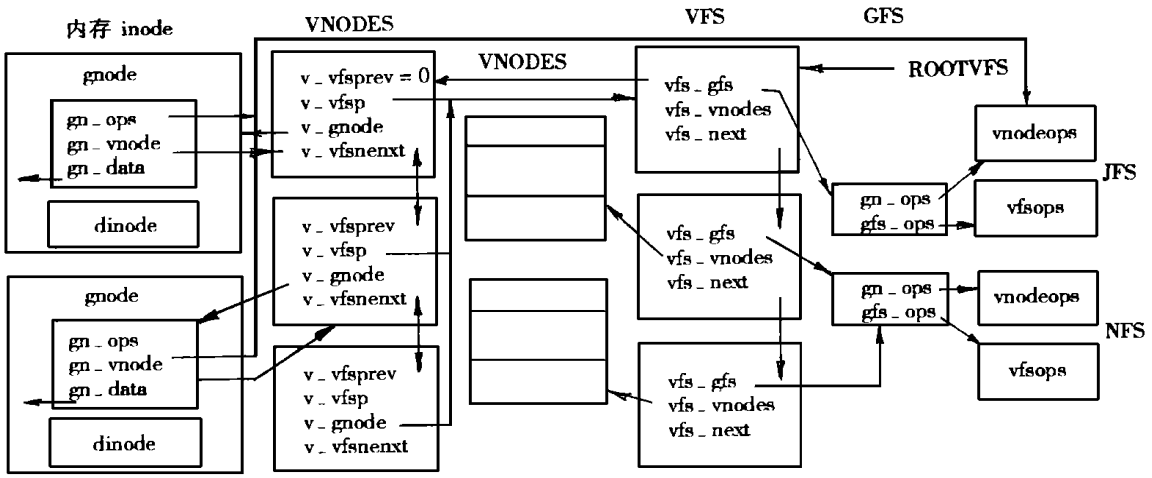

图 3　Vfs/ Vnode 接口中各对象间的关系

3　新的文件系统实现及问题

AIX 系统核心可动态扩展[5], 允许新的扩展功能模块被动态地加载到核心, 新的文件系统代码即作为核心扩展模块处理。实现新的文件系统所作的工作包括:

(1)实现要求的 Vfs/V node 接口函数, 创建与安装 FS 的 HELPER 程序, 编译成可装载的核心扩展目标代码。这部分的工作量很大, 而且要求对操作系统非常熟悉。

(2) 用 sysconfig 系统调用将生成的目标代码加载到核心。系统专门动态扩展核心提供了一个系统调用 sysconfig。

(3) 修改有关的系统配置文件。

然而, AIX4.11 下的 Vfs/ Vnode 接口实现并没有圆满地达到预期的目的。虽然允许通过模块化的方法实现新的 FS。但是几乎不可能在没有操作系统源代码的情况下写新的 FS。这是因为:

(1) FS 同内存管理存在密切的联系, 必须理解内存管理的方法才能实现相关的接口函数。

(2) 不同的 UNIX 系统和版本接口不尽相同。

(3) 接口不是完全透明的。核心直接访问 Vnode 结构中的很多字段, 而不是通过过程调用接口访问。难以在改变 Vnode 结构的时候同先前的版本保持二进制兼容性。

(4) 该接口不支持继承。新的 FS 不能继承已有的 FS 的功能。

(5) 该接口不能扩展。一个 FS 不能增加一个新的功能或改变已经存在的操作的语义。如用户不想开发整个 FS 而只想增加文件复制、加密和解密功能等。现有的接口都不支持。

为了解决困难, 现在提出了一种称为堆栈式分层 FS 结构[7, 8]。在此结构中。每个文件被一 V node 栈代表, 栈中每个 FS 对应一 Vnode。用户发出文件操作时, 核心将它传到栈顶的 Vnode, 并可能完成下列一种或两种操作: 一是执行完操作并返回结果到调用者; 二是可能作些处理, 把操作传到栈中的下层 V node。这样, 操作可以传遍所有的层, 每个 Vnode 有作些额外处理的机会。该结构允许对 FS 的增量式开发, 如实现文件加密、解密和复制等功能。

为了在现有的 AIX 中实现新功能 FS(如可备份的 FS), 同时避开重写全部 FS 代码的困难, 通过分析它的 FS 实现, 找到了一个较简便的方法来实现新的 FS。在新的 FS 实现中, 使用与现有 FS 相同的组织方式, 通过获取它的 Vfs/ Vnode 接口函数指针, 利用此指针调用所需要的接口函数, 通

过修改部分接口函数来实现新的 Vfs 和 VnodeFS，修改的方法是借用已有的接口函数而不全部自己编写实现代码。

首先，为了获得现成的 FS 的接口函数指针，在系统的根目标(FS 类型为 JFS)下创建一个不被删除的空文件(称为“钉子”)，其作用好象设在别的 FS 下的“领事馆”，并被用作获取现有 FS 的接口函数指针，方法如下：

```
……
struct vnode  *anchor;/ *用于指向“钉子”文件的 vnode */
struct vnodeops  *oldvnop;
struct vfsops  *oldvfsop;
lookupvp(pathname, flag,  &anchor, …); / *获取“钉子”文件的 vnode */
oldvnop=anchor->v-gnode>gn-ops;
oldvfsop=anchor->v-vfsp>Vfs->gfs-gfs-ops;
……
```

然后，在实现自己的接口函数时，使用自己的参数来调用对应的原有 FS 的接口函数。同时，利用系统提供的核心服务，还可以实施额外的操作，以增加新的功能。如实现可复制 FS 的 VN-WRITE()操作时，同时还要把修改写到备份的文件中，此时需要对有关的调用参数和该调用涉及到的系统别的结构进行修改。

采用此方法，如果不打算改变原有 FS 的语义机制，实现起来是行之有效的，可以大大缩短开发周期与难度。只需要处理好参数问题和弄清有关的数据结构，完成扩展功能代码、用户空间的初始化程序和配置程序。

4 结 论

本文分析了 AIX4.1.1 下对多文件系统类型支持——Vfs/Vnode 接口的实现，该接口下原系统实现了日志文件系统 JFS、网络文件系统 NFS、只读光盘文件系统 CDRFS，但是在该接口下开发新的 FS 却存在着诸多困难，如不支持继承、功能扩展等，为此提出了一种较方便实现新的文件系统的方法。通过获得现成的文件系统的接口函数地址，利用原有的 FS 实现代码和可动态扩展核心的特点，只需完成增加的功能代码和少量的用户空间程序，即可较方便地实现新的文件系统。该方法的缺点是新实现的 FS 必须与基于的 FS 据有相同的磁盘空间组织方式和语义。

参 考 文 献

1 Bach M J. The design of the UNIX operating system NJ:Prentice-hall, Englewood Cliffs, 1986

2 Kleiman S R. Vnodes:An architecture for multiple file system type in sun UNIX. Proceedings of Summer 1986 USENIX Technical Conference, 1986, 238~247

3 John A. Dynamic Vnodes-design and implementation. Proceedings of Winter 1995 USENIX Technical Conference, 1995, 11~23

4 Mckusick M K. The virtual filesystem interface in 4.4BSD. Computing System, 1995, 8(1):3~25

5 IBM Crop. AIX 4.1.1 Info handbook. IBM Crop, 1994

6 Margo Seltzer, Keith Bostic, Marshall Kirk Mckusick et al. An implementation of a Log-structured file system for UNIX. Proceding of the Winter 1993 USENIX Technical Conference, 1993, 25~29:1~18

7 Rosenthal D S H. Requirements for a "Stacking" Vnode/Vfs interface. UNIX International Document SF-01-92-N014, NJ:Parsippany, 1992

8 Rosenthal D S H, Rosenthal D S H. Evolving the Vnode interface. Proceeding of the Summer 1990 USENIX Technical Conference, 1990：107～118

9 Bershad B N, Pinkerton C B. Wacthdogs-extending the UNIX file system. Computing System, 1988, 1(2)：169～188

Technique for Supporting Serveral Types of File System Under UNIX

Lin Xiaodong　　Liu Xinsong

(Dept. of Computer Science, UEST of China　Chengdu　610054)

Abstract　Many existing types of FS developed to address different application cases demand the OS to support concurrent accessing to different type of FS instance. This paper discribes the technique for supporting serveral of types file system under UNIX operating system—the vnode/vfs interface, and analyzes its implementation in AIX 4.1.1. There are some problems and difficulties for vender to implement his own file system with new functions. In this paper, an easy method is presented to reslove these problems by using existing file system code to simplify the implemetnation of new FS.

Key words　virtual file system;　object;　abstract;　Vnode/Vfs interface;　virtual inode

编辑　徐安玉

·科研成果介绍·

野战地域网动态网络优化设计系统

主研人员：郭　伟　余敬东　张　薇　洪福明　张国平　廖仕珍 等

野战地域网动态网络优化设计系统首次提出了动态网络拓朴优化设计模型，在此基础上创造性地提出了确定网络拓朴结构的原理和算法(等效 KCL 算法和网络模糊连通度算法)。该算法在确保网络具有高的抗毁性能和服务质量的条件下，能快速给出野战地域网的动态优化设计结果；在理论上建立了全新的野战地域通信网的抗毁性和可靠性评估模型，能对网络的抗毁性和可靠性进行快速定量评估；能完成抗毁性和可靠性指标的分配和优化设计；开展了野战地域网初始规模优化设计研究，提出了渐进寻优的递推启发式算法，能满足网络设计条件下快速给出网络初始规划优化设计结果。

雷达信号相位噪声测试仪

主研人员：张有正　李玉柏　王竹红　周正中　蔡竞业　袁家樾 等

雷达信号相位噪声测试仪是以微处理机为核心的智能仪器，用于测量雷达信号的相位噪声，是一种通用的测量仪器。其主要技术指标如下：

1) 频率范围：1～18 GHz。

2) 脉冲宽度：1 μs。

3) 脉冲重复频率：10 kHz，100 kHz。

4) 灵敏度：(1) 单边带相位噪声功率谱 $L_p(f)$：

$L_p(f) \leqslant -110$ dBC/Hz，$f \geqslant 30$ kHz；$V_R = 100$ kHz，$V_0 = 1$ GHz；

$L_p(f) \leqslant -100$ dBC/Hz，$f \geqslant 30$ kHz；$V_R = 100$ kHz，$V_0 = 10$ GHz。

(2) 改善因子：

$I_1 \geqslant 80$ dB，$V_R = 100$ kHz，$V_0 = 1$ GHz；

$I_2 \geqslant 60$ dB，$V_R = 100$ kHz，$V_0 = 10$ GHz。

·科　卞·