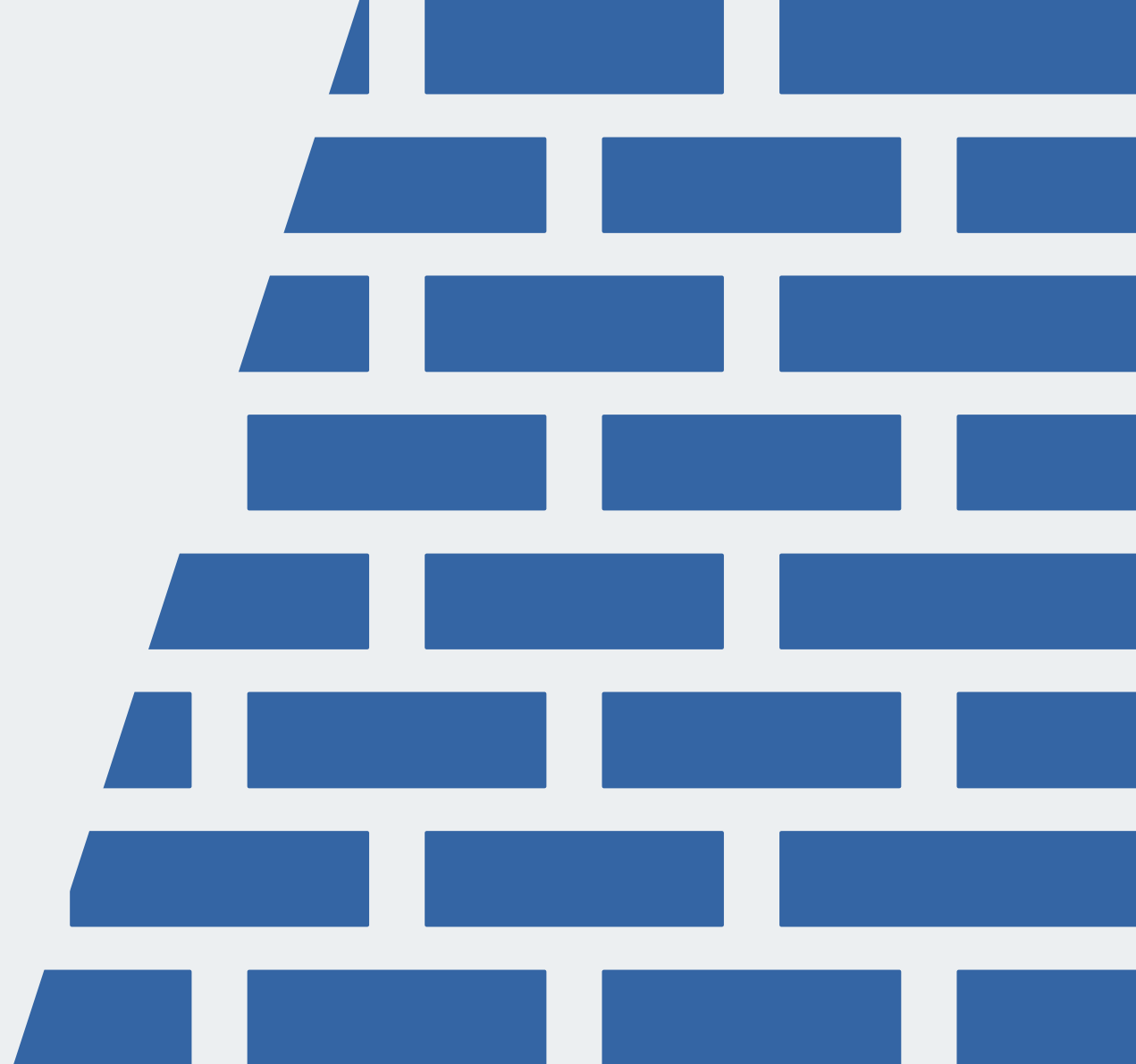


2024 OS 小作业 III: 文件系统

梁亚伦

2024 年 4 月 17 日



前情提要

Unix v6 \rightarrow xv6

只支持一种文件系统, 在作业中我们将其称为 xv6fs

任务：

在 xv6 中实现类似于 Linux 的
虚拟文件系统(VFS)机制

为什么需要 VFS ?

在同一个操作系统中同时使用多个文件系统；
文件系统实现中有很多共性的、与具体文件系统无关的部分；
「LLVM 前端 / 后端」

为什么写 VFS ?


- 现代 Linux 中重要的一部分
- 经典的抽象方法
- 实现起来有一定的难度

什么是 VFS ?

- Virtual File System / Virtual Filesystem Switch / 虚拟文件系统
- 把系统上同时存在的所有文件系统包装起来
- 暴露像一个文件系统一样的接口(「虚拟」)
- 负责接收所有文件系统类的 syscall 并转发给具体文件系统

我还是不理解 VFS 是啥……

Read The Friendly Manual



The Linux Kernel

6.9.0-rc4

Quick search

Contents

- Development process
- Submitting patches
- Code of conduct
- Maintainer handbook
- All development-process docs
- Core API
- Driver APIs
- Subsystems**
- Core subsystems

Overview of the Linux Virtual File System

Original author: Richard Gooch <rgooch@atnf.csiro.au>

- Copyright (C) 1999 Richard Gooch
- Copyright (C) 2005 Pekka Enberg

Introduction

The Virtual File System (also known as the Virtual Filesystem Switch) is the software layer in the kernel that provides the filesystem interface to userspace programs. It also provides an abstraction within the kernel which allows different filesystem implementations to coexist.

VFS system calls `open(2)`, `stat(2)`, `read(2)`, `write(2)`, `chmod(2)` and so on are called from a process context. Filesystem locking is described in the document [Locking](#).

Directory Entry Cache (dcache)

The VFS implements the `open(2)`, `stat(2)`, `chmod(2)`, and similar system calls. The pathname argument that is passed to them is used by the VFS to search through the directory entry cache (also known as the dentry cache or dcache). This provides a very fast look-up mechanism to translate a pathname (filename) into a specific dentry. Dentries live in RAM and are never saved to disc: they exist only for performance.

The dentry cache is meant to be a view into your entire filesystem. As most computers cannot fit all dentries in the RAM at the same time, some bits of the cache are missing. In order to resolve your pathname into a dentry, the VFS may have to resort to creating dentries along the way, and then loading the inode. This is done by looking up the inode.

<https://www.kernel.org/doc/html/latest/filesystems/vfs.html>

好, 我们究竟要干啥?

写一个 VFS 需要……

1. 设计一个 VFS 接口
2. 把相关 syscall 换成 VFS 接口
3. 把既有文件系统 port 到 VFS 上
4. 调试
5. 调试
6. 调试

写一个 VFS 需要……

1. ~~设计一个 VFS 接口~~ (我们提供)
2. 把相关 syscall 换成 VFS 接口
3. 把既有文件系统 port 到 VFS 上
4. 调试
5. 调试
6. 调试

下发代码

<https://git.sjtu.edu.cn/lyl/xv6-riscv>

- 删除了部分不必要的特性
- 将文件系统相关代码移动到 `/kernel/fs/`
- 将 `xv6fs` 相关代码移动到 `/kernel/fs/xv6fs/`
- 将 `xv6fs` 相关 `struct` 重命名为 `xv6fs_XXX`
- 添加 `/kernel/fs/vfs.h`

其余部分与 MIT 版本 `xv6-riscv` 相同。

此次作业的流程 (暂定)

1. 修改 syscall handler 等函数, 使 /kernel/fs/xv6fs/ 外不出现「xv6fs」的字样 (即替换为 VFS 实现);
2. 修改 xv6fs, 接入 VFS 接口 (暂时不用考虑特殊文件的处理);
3. 修改路径解析逻辑, 处理 mount / umount;
4. 加入对设备文件和管道文件的处理;
5. 实现一个 (非常朴素、无需落盘的) 自定义文件系统。

> 三 ✎

~: bat — Konsole

⤴ ⤵ ⤶ ✕

📄 新建标签页(N) 📄 拆分视图 ▾

📄 复制(C) 📄 粘贴(P) 🔍 查找(F) ☰

	File: /proc/mounts
1	proc /proc proc rw,nosuid,nodev,noexec,relatime 0 0
2	sys /sys sysfs rw,nosuid,nodev,noexec,relatime 0 0
3	dev /dev devtmpfs rw,nosuid,relatime,size=20054144k,nr_inodes=5013536,mode=755,inode64 0 0
4	run /run tmpfs rw,nosuid,nodev,relatime,mode=755,inode64 0 0
5	efivarfs /sys/firmware/efi/efivars efivarfs rw,nosuid,nodev,noexec,relatime 0 0
6	/dev/mapper/volumngroup-root / ext4 rw,relatime 0 0
7	securityfs /sys/kernel/security securityfs rw,nosuid,nodev,noexec,relatime 0 0
8	tmpfs /dev/shm tmpfs rw,nosuid,nodev,inode64 0 0
9	devpts /dev/pts devpts rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000 0 0
10	cgroup2 /sys/fs/cgroup cgroup2 rw,nosuid,nodev,noexec,relatime,nsdelegate,memory_recursivepr ot 0 0
11	pstore /sys/fs/pstore pstore rw,nosuid,nodev,noexec,relatime 0 0
12	bpfs /sys/fs/bpf bpfs rw,nosuid,nodev,noexec,relatime,mode=700 0 0
13	systemd-1 /proc/sys/fs/binfmt_misc autofs rw,relatime,fd=37,pgrp=1,timeout=0,minproto=5,maxp roto=5,direct,pipe_ino=13741 0 0
14	mqueue /dev/mqueue mqueue rw,nosuid,nodev,noexec,relatime 0 0
15	hugetlbfs /dev/hugepages hugetlbfs rw,nosuid,nodev,relatime,pagesize=2M 0 0
16	debugfs /sys/kernel/debug debugfs rw,nosuid,nodev,noexec,relatime 0 0
17	tracefs /sys/kernel/tracing tracefs rw,nosuid,nodev,noexec,relatime 0 0
18	fusectl /sys/fs/fuse/connections fusectl rw,nosuid,nodev,noexec,relatime 0 0
19	configfs /sys/kernel/config configfs rw,nosuid,nodev,noexec,relatime 0 0
20	tmpfs /tmp tmpfs rw,nosuid,nodev,size=20144040k,nr_inodes=1048576,inode64 0 0
21	/dev/nvme0n1p1 /boot vfat rw,relatime,fmask=0022,dmask=0022,codepage=437,iocharset=ascii,sho rtname=mixed,utf8,errors=remount-ro 0 0

:

时间安排 (暂定)

第 9 周： 熟悉 xv6 中的文件系统

第 10 周–第 12 周： VFS 实现

第 13 周–第 14 周： mount 和特殊文件

第 15 周： 自定义文件系统

你可能想问……

- 可以改 VFS 接口定义吗？

可以, 但需要向助教申请。合理的修改均会通过。

- 给的接口不是很明确啊？

你应当自己明确接口的具体含义。(例如, 调用时是否需要加锁?)

- 给的 VFS 接口看不懂怎么办？

请参考 Linux 中对应接口的说明。如果看过之后还是不明白, 请联系助教。

- 会有中期检查吗？

会。近期我会发一个具体的任务说明。

Q&A