Lab5挑战性任务

题目背景与描述

在Lab5的实验中，我们实现了一个简单的文件系统。其中的部分操作使用了文件系统IPC机制，这样的机制可以被视为文件系统抽象层（VFS）。文件系统抽象层在Linux等大型操作系统中的应用十分广泛，它使得操作系统能够与多个不同类型的文件系统进行交互，这能够提升文件系统的可用性。

FAT文件系统全称"File Allocation Table"，它是一个经典的文件系统，曾被微软广泛应用于DOS系列和Windows系列操作系统，得益于它足够简单，它成为了许多存储设备所采用的文件系统，且受到大部分电脑系统的支持。

在本任务中，我们希望你能实现一个**FAT文件系统**，与MOS原文件系统并行运行，组成一个简单的**多文件系统**，在此过程中体会FAT文件系统的设计精髓和多文件系统的思想。你可以根据个人兴趣和能力，选择FAT12、FAT16、FAT32文件系统中的任意一个进行实现，此处不作具体要求。具体步骤如下：

1. 构建FAT格式磁盘镜像

要建立FAT文件系统，我们首先需要FAT格式的磁盘镜像，这里提供两种方法思路：

- **使用Linux相关命令（推荐）**
  - 以创建FAT12磁盘镜像为例：

```
# 创建磁盘镜像
dd if=/dev/zero of=fs.img bs=16k count=1024
mkfs.vfat -F 12 fs.img
# 将磁盘镜像挂载到Linux下，向磁盘镜像写入测试用文件
mkdir test_fat
mount fs.img test_fat
cp testfile test_fat
```

  - 如果遇到权限不足的问题，可在本地虚拟机内先构建并写入测试程序，再提交到远程仓库，来达到创建带有测试文件的磁盘镜像的目的。
- **编写代码自行构建镜像**
  - 参考已有的`fs/fsformat.c`相关代码，根据FAT文件系统格式要求，创建磁盘镜像
  - 你需要在创建磁盘镜像的同时向其中写入相应的测试文件
  - 采用此方式构建磁盘镜像，请在实验报告内写上向镜像写入测试文件的关键代码。此方式不一定需要严格遵守FAT文件系统的标准格式，可自行发挥。
  - 注：`fs/fsformat.c`文件并不依赖MOS操作系统的环境，它使用的Linux下的C语言标准库

磁盘镜像制作完成后，我们需要将其挂载到MOS操作系统下，与原文件系统磁盘镜像并行运行。使用`gxemul -E testmips -C R3000 -M 64 -d 0:gxemul/fs.img -d 1:gxemul/fs2.img elf-file`可以挂载多个文件系统镜像，其中0，1指定了不同的虚拟磁盘的磁盘ID。（指令请自行修改来满足MOS系统正常运行的要求）

2. 实现FAT文件系统

- 创建`fs/fatfs.c`文件及相关头文件，参考`fs/fs.c`内的代码逻辑，结合FAT文件系统格式，实现FAT文件系统的基本功能。在此过程中，你需要新的结构体来适应FAT文件系统的文件格式，以及针对性的函数来适应FAT文件系统的文件组织形式。

- 你需要实现文件操作相关的基本功能，能够打开或关闭文件、读取或写入数据到已打开文件所使用的某个簇、调整文件大小、刷新缓冲区等。
- 由于实现的功能有限，对于文件的相关操作可能较为麻烦，可以参考`fs/test.c`编写测试程序，但应比其更加详细。

3. 兼容文件系统用户接口

为了能够同时访问两个不同的文件系统，我们需要对原有的文件系统用户接口做出一些调整，具体步骤如下：

- 创建`fs/fatserve.c`文件，编写相关函数，用于启动FAT文件系统进程，与用户接口进行交互。
- 用户在使用时可以指定需要交互的文件系统进程，一种可行的思路是通过调整路径来指定需要交互的文件系统进程。你也可以采取其他思路进行实现，保证正常交互即可。此过程需要修改`user/fsipc.c`或`user/file.c`文件，来实现不同文件系统进程的选取。
  - 例如，`open("/root1/...", ...)`访问的是原文件系统的内容，即该用户需要和MOS原文件系统进程进行交互；`open("/root2/...", ...)`访问的是FAT文件系统的内容，需要和FAT文件系统进程进行交互。
- `user/file.c`相关操作用到了步骤2不曾创建/使用的函数，需要进一步编写完善。如`open`函数在打开文件的同时，将文件的所有内容映射到某个虚拟地址区域，需要实现新的map函数支持FAT文件系统完成该操作。
- 测试程序应包含对FAT文件系统用户接口功能的完整测试
  - 对于新建、打开、读取、写入、删除等功能，保证文件的数据符合预期（可以在不同的操作后输出文件内容）。
  - 保证磁盘镜像内有足够大的测试文件能够用到文件的第二个及之后的簇，证明功能的完整性。
  - 保证测试程序包含对原文件系统的足够调用，验证多文件系统的可用性

提交要求

- 请自行建立lab5-challenge，在该分支完成代码后，push到个人的远程仓库。代码内需要包含对于功能的详细测试程序，测试程序本身及运行测试程序得到的运行结果应具有足够的可读性。

```
git checkout lab5
git add .
git commit -m "xxxxx"
git checkout -b lab5-challenge
# 完成代码
git push origin lab5-challenge:lab5-challenge
```

- 实验报告请提交至spoc系统，在书写**实验报告**和准备**申优答辩**时，请加入以下内容：
  - 对于任务的实现思路，并配合关键代码进行说明。
  - 对于功能的详细**测试程序**，以及运行测试程序得到的运行结果。
  - 完成挑战性任务过程中**遇到的问题及解决方案**。