提交评测

提交

更新

Lab 5-2 Extra

准备工作：创建并切换到 `lab5-2-extra` 分支

请在**自动初始化分支后**，在开发机依次执行以下命令：

```
$ cd ~/学号
$ git fetch
$ git checkout lab5-2-extra
```

初始化的 `lab5-2-extra` 分支基于课下完成的 `lab5` 分支，并且在 `tests` 目录下添加了 `lab5_mode` 样例测试目录。

题目背景与描述

在 Linux 下文件有着权限这一概念，在 Lab0 中我们也已经学习过使用 `chmod` 命令来修改文件的权限。

本次实验中，你将实现一个最为基础的文件权限系统。由于我们的 MOS 操作系统并未提供用户、用户组的概念，因此对于一个文件，我们只考虑使用一个 3 位的二进制来表示其读、写、执行的权限。你需要实现文件权限的获取与修改，并在打开文件或遍历文件夹时检查相应的权限。

题目要求

1. 在 `user/include/fs.h` 中新增字段 `uint32_t f_mode` ，在通过 `fsformat` 创建磁盘时，将待写入文件（文件夹）的当前用户权限（3 位的二进制数 `000b` ~ `111b` ）写入 `File` 结构体的新增字段 `f_mode` 中。在 `fs/fs.c` 的 `file_create` 创建文件时设置新文件的默认权限为 `111b` 。
2. 在 `user/lib/file.c` 中提供一个用户态函数 `int chmod(const char *path, u_int mode, int type)` 修改 `path` 路径对应文件的权限。
   - 当 `type` 为 0 时，表示设置权限，直接设置文件权限为 `mode` 。
   - 当 `type` 为 1 时，表示添加权限，将 `mode` 指定的权限添加至文件当的权限中。
   - 当 `type` 为 2 时，表示移除权限，将 `mode` 指定的权限从文件当前的权限中移除。

提交评测

4. 添加错误码定义 `#define E_PERM_DENY 14` 。在函数 `serve_open` 、 `dir_lookup` 、 `file_create` 和 `file_remove` 中进行对应的权限检查，若权限检查不通过返回 `-E_PERM_DENY` 。
    - `serve_open` 时需检查待打开的文件是否具有 `struct Fsreq_open *rq` 中请求的所有权限。**注意**： `rp->req_omode` 的取值为 `O_RDONLY` , `O_WRONLY` , `O_RDWR` （分别为 0，1，2），与我们 `f_mode` 中的三位二进制表示不同。
    - `dir_lookup` 时需检查文件夹是否具有执行（ `FMODE_X` ）的权限。
    - `file_create` 时需检查创建文件时的**目标目录**是否具有写（ `FMODE_W` ）权限，同时需要将新创建的文件权限设置为默认的 `FMODE_ALL` 。
    - `file_remove` 时需检查删除文件时文件所在的**直接父目录**是否具有写（ `FMODE_W` ）权限。

参考实现

1. 在 `user/include/fs.h` 中添加以下定义。

```
#define FMODE_R 0x4
#define FMODE_W 0x2
#define FMODE_X 0x1
#define FMODE_RW 0x6
#define FMODE_ALL 0x7

#define STMODE2FMODE(st_mode) (((st_mode) >> 6) & FMODE_ALL)
```

   并在 `struct File` 中添加字段 `uint32_t f_mode` 并修改 `f_pad` 以保证 `sizeof(struct File)` 仍为 `FILE_STRUCT_SIZE` 。
2. 修改 `tools/fsformat.c` , 在函数 `write_file` 与 `write_directory` 中为 `create_file` 创建的文件结构体设置 `f_mode` 的值为 `STMODE2FMODE(stat_buf.st_mode)` , 其中 `stat_buf` 可使用如下代码获取。在函数 `init_disk` 内将根目录的 `f_mode` 置为 `FMODE_ALL` 。

```
struct stat stat_buf;
assert(stat(path, &stat_buf) == 0);
```

3. 在 `user/include/fsreq.h` 中添加 `chmod` 请求的结构体如下，并添加文件系统的请求类型 `FSREQ_CHMOD` 。

```
struct Fsreq_chmod {
    char req_path[MAXPATHLEN];
    u_int req_mode;
```

提交评测

4. 在 `user/include/lib.h` 中添加函数声明 `int fsipc_chmod(const char *, u_int, int);` 与 `int chmod(const char *path, u_int mode, int type);` 。在 `user/lib/fsipc.c` 中仿照现有 fsipc 请求实现 `fsipc_chmod` 。在 `user/lib/file.c` 中实现 `chmod` 函数。
5. 在 `fs/serv.c` 中实现 `serve_chmod` 函数，并添加至 `serve_table` 。在 `serve_chmod` 中，使用 `file_open` 打开请求路径的文件，若打开失败的错误码为 `r` ，则调用 `ipc_send(envid, r, 0, 0);` 并返回。打开文件成功后依据 `type` 和 `mode` ，修改其 `f_mode` ，并调用 `file_close` 关闭文件，写回文件权限，成功后则调用 `ipc_send(envid, 0, 0, 0);` 返回 0。
6. 在 `user/include/fd.h` 的 `struct Stat` 中添加字段 `u_int st_mode` ，并在 `user/lib/file.c` 的 `file_stat` 函数中设置 `st->st_mode` 的值。
7. 在 `fs/fs.c` 的 `file_create` 函数中，为新创建的文件 `f` 设置默认权限为 `FMODE_ALL` 。
8. 在 `include/error.h` 中添加定义 `#define E_PERM_DENY 14` 。在 `fs/serv.c` 的 `serve_open` 函数中，检查打开的文件是否具有请求中的权限 `rq->req_omode` ，若缺少请求的权限，调用 `ipc_send(envid, -E_PERM_DENY, 0, 0);` 并返回。在 `fs/fs.c` 的 `dir_lookup` 函数中，遍历文件夹前首先检查 `dir` 是否具有权限 `FMODE_X` ，若没有则返回 `-E_PERM_DENY` 。在 `file_create` 和 `file_remove` 函数中，检查所在目录 `dir` 是否具有权限 `FMODE_W` ，若没有则返回 `-E_PERM_DENY` 。

本地测试说明

在测试样例中，文件系统仅在根目录下存在一个 `/newmotd` 文件，仅涉及对该文件的权限修改、打开与关闭测试。具体测试逻辑请参见 `tests/lab5_mode/serv_check.c` 。

你可以使用：

- `make test lab=5_mode && make run` 在本地测试上述样例（调试模式）
- `MOS_PROFILE=release make test lab=5_mode && make run` 在本地测试上述样例（开启优化）

若你正确实现了所有功能逻辑，在本地测试中你将看到以下输出，看到 `File mode test passed!` 即表明通过测试。

```
OK, can open /newmotd with O_RDWR
Removed write permisson of /newmotd
OK, can not open /newmotd with O_WRONLY
OK, can not open /newmotd with O_RDWR
Set /newmotd mode to write only
```

提交评测

```
OK, granted all permissions of /newmotd
File mode test passed!
```

提交

提交评测

更新

请在开发机中执行下列命令后，**在课程网站上提交评测。**

```
$ cd ~/学号/
$ git add -A
$ git commit -m "message" # 请将 message 改为有意义的信息
$ git push
```

评测说明

具体要求和分数分布如下：

测试点序号	评测说明	分值
1	样例	10
2	仅检查 `chmod` 的返回值	15
3	文件权限修改测试	25
4	目录权限修改测试	30