考试时间：14:00~16:30

考虑到有可能存在VPN连接不稳定的情况，建议同学们尽量在本地完成，再copy到跳板机相关文件里，避免中途断连影响代码编写。

Lab5-2-exam

创建并切换分支

```
git checkout lab5
git add .
git commit -m "xxxx" --allow-empty
git checkout -b lab5-2-exam
```

题目描述

在现有的文件系统上进行三个扩展，每个扩展各30分，编译正确10分。每个扩展不能影响已有实现的正确性。

1. O_APPEND (30')

目前我们的操作系统在打开一个文件后向其中写入内容时，会从文件起始位置开始，这会覆盖原有文件内容。示例如下：

文件内容如下

```
This is /motd, the message of the day.
Welcome to the MOS kernel, now with a file system!
```

代码如下

```
int r, fdnum, n;
char buf[200];
if ((r = open("/motd", O_RDWR) < 0)) {
    user_panic("open /motd: %d\n", r);
}
fdnum = r;
if (r = fwritef(fdnum, "test append") < 0) {
    user_panic("fwritef %d\n", r);
}
close(fdnum);
if ((r = open("/motd", O_RDWR) < 0)) {
    user_panic("open /motd: %d\n", r);
}
if ((n = read(fdnum, buf, 150)) < 0) {
    writef("read %d\n", n);
```

```
    return;
}
writef("\n%s\n", buf);
```

输出如下

```
test appendtd, the message of the day.
Welcome to the MOS kernel, now with a file system!
```

我们希望你在 `open` 函数的参数中增加一个 `O_APPEND` 选项，使得文件打开后偏移指针为文件的大小，写的内容会添加在文件最后。

即以 `O_APPEND` 打开文件 `motd`（`open("/motd", O_RDWR | O_APPEND)`）后，进行上述写操作后，输出如下:

```
This is /motd, the message of the day.
Welcome to the MOS kernel, now with a file system!
test append
```

tips

- 需要在 `user/lib.h` 中定义 `#define O_APPEND 0x0004`。
- 示例输出删除了多个连续的换行符，实际以本地测试的输出为准。
- 仅需修改 `user/file.c` 中的 `open` 函数

2. O_ALONE (30')

我们在lab4中实现了重要的 `fork` 函数，目前我们的操作系统在父进程 `fork` 了子进程后，两个进程会共享文件描述符表，也同时会共用文件描述符的偏移指针，示例如下:

文件内容如下

```
This is a NEW message of the day!
```

代码如下

```
int r, fdnum, n;
char buf[200];
fdnum = open("/newmotd", O_RDWR);
if ((r = fork()) == 0) {
    n = read(fdnum, buf, 5);
    writef("[child] buffer is \'%s\'\n", buf);
} else {
    n = read(fdnum, buf, 5);
    writef("[father] buffer is \'%s\'\n", buf);
}
```

输出如下

```
[father] buffer is 'This '
[child] buffer is 'is a '
```

我们希望你在`open`函数的参数中增加一个`O_ALONE`选项，使得以此方式打开文件的父进程与子进程不再共享这一文件的文件描述符的偏移指针。

即，以`O_ALONE`打开文件`newmotd`（`open('/newmotd', O_RDWR|O_ALONE)`）后，进行上述的`fork`和`read`，输出如下:

```
[father] buffer is 'This '
[child] buffer is 'This '
```

tips

- 需要在`user/lib.h`中定义`#define O_ALONE 0x0008`。
- 修改文件描述符页面进行映射时的`PTE_LIBRARY`标志位。
- 本部分测试仅调用`open` `read` `write`函数，不会调用`close`函数(无需考虑父子进程写回文件不一致的问题)。

3. O_CREAT (30')

目前我们的操作系统中，若用户进程需要打开的文件不存在时，会返回错误`E_NOT_FOUND`。

我们希望你在`open`函数的参数中增加一个`O_CREAT`选项，使得在用户想要打开的文件不存在时，会创建一个空文件，并正常打开它。

代码如下

```
int fdnum;
char buf[200];
fdnum = open("/created_file", O_RDWR|O_CREAT);
fwritef(fdnum, "test create");
close(fdnum);
fdnum = open("/created_file", O_RDWR);
read(fdnum, buf, 150);
writef("read from new file: %s\n", buf);
```

输出如下

```
test create
```

tips

- `user/lib.h`中对`O_CREAT`已经有了定义，无需修改。
- 需要调用`fs.c`中的`file_create`函数。

本地测试

在init.c中启动文件系统服务进程和文件系统测试进程(与lab5课下本地测试一致)

```
ENV_CREATE(user_fstest);
ENV_CREATE(fs_serv);
```

将随题下发的`fstest_o_xxx.txt`文件内容拷贝覆盖`user/fstest.c`原有内容，即可展开本地测试，期望输出在此文件最后的注释中。

运行命令为：

```
make clean && make && /OSLAB/gxemul -E testmips -C R3000 -M 64 -d gxemul/fs.img
gxemul/vmlinux
```

Lab5-2-Extra

创建并切换分支

注意：需要基于exam分支创建extra分支

```
git checkout lab5-2-exam
git add .
git commit -m "xxxx" --allow-empty
git checkout -b lab5-2-Extra
```

题目描述

在`user/file.c`实现函数

```
int list_dir(const char *path, char* ans)
```

该函数接受以`path`为绝对路径的文件夹，将此文件夹的所有文件的文件名以`空格(space)`为间隔保存在字符串`ans`中。函数执行成功时返回`0`，目录不存在时返回`-1`。

约定：

- 当一个文件夹有多个文件时，返回文件名的顺序和`dir_lookup`遍历的顺序相同。
- 测试时`char* ans`的长度不会超过1024byte。

原始的lab5(仅在根目录下烧录了`newmotd`和`motd`文件)的示例如下：

代码如下

```
int r;
if (r = list_dir("/", name_buf) < 0) { // 列出根目录下的文件名
    writef("cannot list_dir\n");
}
writef("%s\n", name_buf);
```

输出如下

```
motd newmotd
```

实现步骤

1. 在`user/lib.h`中添加函数声明`int list_dir(const char* path, char* ans);`。
2. 回顾课下实现`fs/fs.c`中的`dir_lookup`函数，模仿此函数列出一个目录下的所有文件的文件名。
3. 在文件系统服务进程中增加新的`Fsreq`及相关服务，通过IPC机制将结果回传给用户进程。

tips

- 成功列出根目录下的所有文件的文件名可获得60分，能够列出任意目录下的文件名可获得100分。
- 仅考虑目录直接包含的文件，不考虑递归情况。
- 注意：如果用户进程接受IPC映射的`dst`页面已经被映射，需要调用`syscall_mem_unmap`解除映射关系。

本地测试

在init.c中启动文件系统服务进程和文件系统测试进程(与lab5课下本地测试一致)

```
ENV_CREATE(user_fstest);
ENV_CREATE(fs_serv);
```

将随题下发的`fstest_list_dir.txt`文件内容拷贝覆盖`user/fstest.c`原有内容，即可展开本地测试，期望输出在此文件最后的注释中。运行命令为：

```
make clean && make && /OSLAB/gxemul -E testmips -C R3000 -M 64 -d gxemul/fs.img
gxemul/vmlinux
```