文件系统基本接口

该部分分为三个板块，目的是帮助基本掌握fs中的三个基本接口的使用。

mmap()

> 将磁盘文件映射到内存

Task1: 通过mmap修改文件

mmap是一种内存映射文件的方法，即将一个文件或者其它对象映射到进程的地址空间，实现文件磁盘地址和进程虚拟地址空间中一段虚拟地址的一一对映关系。实现这样的映射关系后，进程就可以采用指针的方式读写操作这一段内存。

在这个任务中，你需要使用mmap来实现通过共享映射方式的文件修改。你只能够按照注释的要求添加代码，而不可以对代码有任何的删减或修改。

当你完成必要的填空之后，你可以通过在task1目录下执行make以编译程序，并且在同一个目录下执行make run以测试你的结果。

本任务不计入分数。

Task2: 通过共享映射实现两个进程之间的通信

进程间通信（Inter-process communication）有许多种方法，其中一种是共享内存，即两个进程读写同一段内存来实现信息交换。

在这个任务中，给出的程序实现了两个进程将同一个文件映射到自己的地址空间，进程A先运行，每隔1秒读取映射区域，5秒后运行进程B，修改映射区域，从进程A中观察映射区域内容变化。

你需要通过在task2目录下执行make以编译程序，并且在同一个目录下执行make run以观察程序的运行结果。

本任务不计入分数。

Task3: 通过匿名映射实现父子进程通信

进程间通信不一定需要写入文件。我们可以使用匿名映射（anonymous mapping）使得多进程运行时访问同一段内存，而不进行任何文件操作。

在这个任务中，你需要实现在当前进程中创建一个子进程，用匿名映射的方法从父进程向子进程传输一条消息，再从子进程向父进程传输一条消息。

你需要通过在task3目录下执行make以编译程序，并且在同一个目录下执行make run以观察程序的运行结果。

本任务不计入分数。

Task4: 共享映射读写与传统系统调用读写 (read, write) 读写速度比较

当进行一般的文件读操作时，磁盘空间的文件需要先拷贝到内核态，再从内核态拷贝到用户态，一共是两次拷贝操作；mmap将文件从用磁盘直接映射到用户态内存，只需要一次拷贝。写操作同理。

在这个任务中，4个函数分别测量了使用共享映射读写/系统调用读写64MB文件所需的时间。而对于这个64MB的读写，我们采用了多个不同的，每次读/写 bytes，来观察不同读写粒度两种读写方式的性能差异。

你可以通过在task4目录下执行make以编译程序，并且在同一个目录下执行make run-write以测试文件写入部分，执行make run-read以测试文件读取部分。

本任务不计入分数。

注：wsl的文件系统和linux可能有所差异，有条件建议在linux环境下测量。

fcntl()

> 在文件使用过程中修改其相关属性

Task5: 利用fcntl复制fd与控制锁

在这个小作业中，你需要熟练掌握 `fcntl` 、 `open` 的使用。在 `main.c` 中，你需要先打开文件 `test.txt` 并且向内写入'hello fcntl!'。完成写入之后，你需要使用 `fcntl` 复制当前的文件标识符，并且通过 `fork` 生成另外一个进程。两个进程需要完成的任务如下：

1. 子进程睡眠2秒后，尝试获取文件的锁定状态，并且向文件追加写入字符串a；
2. 父进程获取文件的锁，向文件追加写入字符串b。完成写入后睡眠3秒，输出文件的内容。

代码中只提供了框架，你需要完成剩余部分。

ioctl()

> 对I/O设备进行控制

Task6: 利用ioctl()来获取网卡信息

在这个小作业中，你需要试图使用ioctl()函数来获取网卡设备的设备名、ipv4地址、广播地址和mtu。

你可能会用到 `SIOCGIFCONF`，`SIOCGIFBRDADDR`，`SIOCGIFMTU` 这些关键字。在 `main.cpp` 中，我们首先用 `ifconf` 结构获取所有的网络设备信息，然后再从中分别获取设备名等信息。

代码中已经完成了大部分工作，你只需要调用ioctl()函数填空即可。

注：该程序可能需要在linux环境下运行。

FUSE：用户态文件系统

目标：实现一个简单的用户态文件系统驱动

在Linux内核中，文件系统处于内核态之中。然而越来越多的低延迟新硬件使得特权级别切换所需要的开销占比越来越大，将部分驱动在用户态实现并且在用户态管理已经成为一种较多人的选择。你的任务是使用Linux给出的Fuse框架实现一个简单的用户态文件系统。由于更主要的精力是放在体验Fuse上，因此并不需要实现一个一般意义的文件系统。

基本实现：70%

实现一个基于FUSE的聊天软件。比如，你可以在你的电脑上挂载两个用户态文件系统（比如分别是/mnt/bot1和/mnt/bot2），执行echo "Hello" > /mnt/bot2/bot1 之后应该在 bot1中找到一个文件叫bot2，并且其中内容是Hello。

- 支持文件系统挂载
- 支持文件的基本读写操作，至少支持echo、cat命令
- 支持目录的ls、mkdir、cd操作
- 自建数据结构对文件和目录节点进行维护
- 实现cd, ls, touch, cat, echo, mkdir, rmdir, rm命令

Review：向我展示你的chatbot及实现 20%

Bonus：完成任意两个 或 仅完成权限检查或文件锁 10%

- 将你的chatbot做的尽量有趣
- 测试你的文件系统效率并和传统文件系统对比
- 支持文件权限检查
- 支持文件锁
- 任何其他的细节实现

参考资料

libfuse库：

https://github.com/libfuse/libfuse

文档和操作手册：

http://libfuse.github.io/doxygen/index.html

https://www.kernel.org/doc/html/latest/filesystems/fuse.html

https://man7.org/linux/man-pages/man4/fuse.4.html

可供参考的博客：

编译安装FUSE：https://www.jianshu.com/p/040bb60aa468