lab9-inode

本次实验你们将实现 xv6 文件系统的中的 inode 层。请将 `src/fs/inode.c` 中的所有 TODO 完成。

参考资料：

1. 实验课 slides
2. The xv6 book，第 8 章，“File system”

建议在动手前先阅读 `src/fs/` 目录下的所有头文件，对整体的代码框架有个大概的了解。

测试

本次实验通过标准如下：

1. `make kernel` 编译通过
2. 通过文件系统的单元测试 `inode_test`

运行 `inode_test` 的方法如下：

```
cd src/fs/test
mkdir build
cd build
cmake ..
make && ./inode_test
```

如果没有显示任何 `(error)` 或 `(fatal)` 字样，则表明通过测试。

如果遇到问题，请阅读测试代码 `src/fs/test/inode_test.cpp` 并参考实验课 slides 中的调试指南进行调试。

警告：我们不保证测试足够全面，请不要过于依赖于我们提供的测试。如果你的代码存在问题，可能会影响到后续实验。

提示

推荐完成顺序及相关提示（不搞猜谜，没说清楚来问助教）

先行阅读inode.h与slides相关部分。

建议在理解清楚以下问题后开始写代码：

- inode的生命周期是怎样的？（ppt16或许有些谜语）
    - inode是在内存上的，get\put 是对存储inode的数据结构的操作。
- inode目前主要有2种类型，常规文件和目录文件。他们分别对应的数据是如何布局的？(ppt19,20)

```
u32 addrs[INODE_NUM_DIRECT];  // direct addresses/block numbers.
u32 indirect;                 // the indirect address block
```

- inode_map的功能（ppt21）
- 需要注意避免带锁睡眠（但是，如果保证在第一时间能拿到睡眠锁，并马上释放掉自旋锁，这种做法是ok的,比如下面的情况）

```
init sleeplock
acquire spinlock
acquire sleeplock
release spinlock
release sleeplock
```

需要完成的代码建议完成顺序及分组

- lock, unlock——inode的访问控制
- alloc, get, sync, share, put, clear——内存中的inode、磁盘上的inode_entry的管理
- read, write——文件类型的inode的内容管理接口
    - inode_map：read、write的辅助函数
- insert, lookup, remove——目录类型的inode的内容管理接口

思考题

本部分不计分，可以不作答。

1. 阅读 `src/fs/defines.h` 中的 `InodeEntry` 结构体。我们代码中的一个 inode 能存放多少字节的二进制数据？如果我们想在一个 inode 中存储更多的二进制数据，需要对 inode 的机制和 `InodeEntry` 做怎样的修改？