一、作业名称

编写一个带缓存的文件操作类

二、作业要求

编写一个类，实现对linux api中open，write，read，lseek和close的封装。并在类内部采用buffer来进行缓冲。

1. read:
    1. ret：
        1. 0：If the file offset is at or past the end of file, no bytes are read, and read() returns zero.
        2. -1：错误
        3. > 0：正常读出
2. write:
    1. ret：if count is greater than SSIZE_MAX, the result is implementation-defined;
        1. -1：error
        2. > =0：正常

三、设计与实现

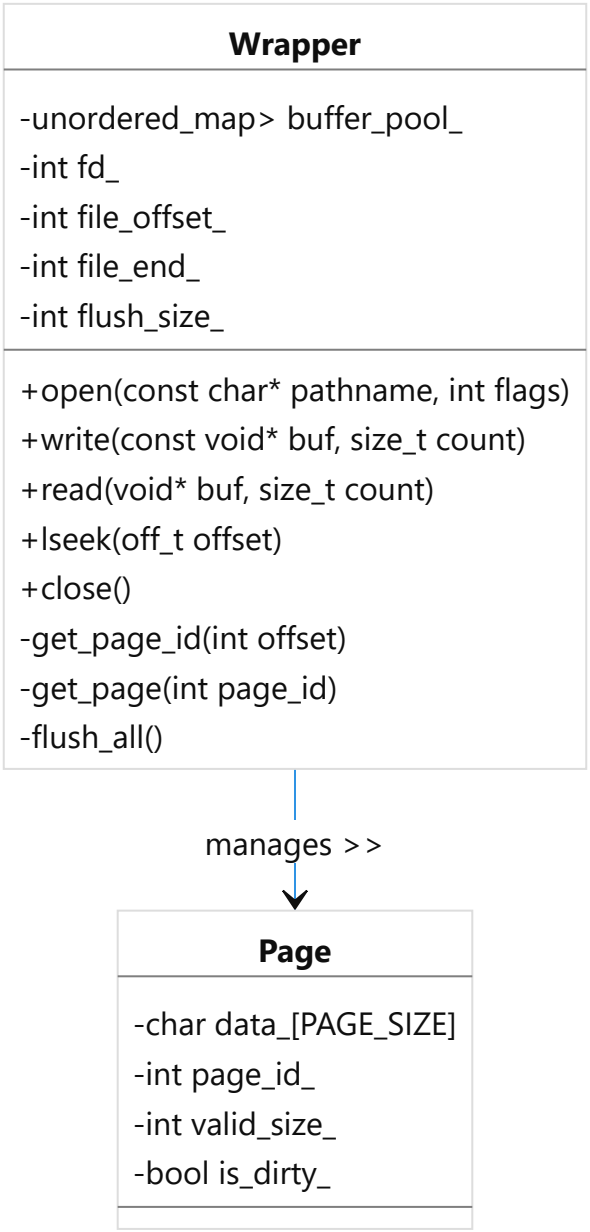
3.1 整体设计

项目 "homework1" 的主要组成部分包括以下几个文件：

1. Page.h: 定义了一个名为 `Page` 的结构体，用于表示内存中的页面。它包含数据数组 `data_`，页面ID `page_id_`，有效数据大小 `valid_size_`，以及一个表示页面是否被修改的标志 `is_dirty_`。
2. Wrapper.h 和 Wrapper.cpp: 这两个文件定义了 `Wrapper` 类，它封装了文件操作的基本功能，如打开、读取、写入、移动文件指针和关闭文件。此外，它还管理一个页面缓冲池 `buffer_pool_`，用于存储和管理内存中的页面。
3. test.cpp: 包含了一系列测试用例，用于验证 `Wrapper` 类的功能。这些测试用例使用了 Google Test 框架。

写入或者读取文件时，先通过缓冲池来进行缓冲，之后再

3.2 类图



3.3 缓冲池的实现

缓冲池是通过 `Wrapper` 类实现的，主要用于管理内存中的页面（`Page` 对象）。这个缓冲池的作用是减少对磁盘的直接读写操作，提高文件操作的效率。下面是缓冲池工作原理的详细解释：

缓冲池的作用

1. **减少磁盘I/O操作**: 通过在内存中暂存数据，减少直接对磁盘的读写次数。
2. **提高数据访问速度**: 内存的访问速度远高于磁盘，因此缓冲池可以加快数据处理速度。
3. **支持数据的临时修改**: 允许在内存中修改数据，而不是每次更改都直接写入磁盘。

缓冲池的工作流程

1. **读取数据**:
   - 当 `Wrapper` 类的 `read` 方法被调用时，它首先检查所需数据的页面是否已经在缓冲池中。
   - 如果页面在缓冲池中，直接从缓冲池中读取数据。
   - 如果页面不在缓冲池中，从磁盘读取页面到缓冲池，然后再从缓冲池中读取数据。
2. **写入数据**:
   - 当 `write` 方法被调用时，数据首先被写入到缓冲池中的相应页面。
   - 如果写入的数据跨越多个页面，会涉及到多个页面的更新。
   - 这些更改暂时只在内存中进行，不会立即写入磁盘。
3. **页面管理**:
   - 缓冲池的大小是有限的。当达到一定的大小或条件时（例如，缓冲池满了或文件关闭时），缓冲池中的页面会被“刷新”到磁盘，即将更改过的数据写回磁盘。
   - 这个过程由 `flush_all` 方法管理。
4. **关闭文件**:
   - 在关闭文件时（调用 `close` 方法），所有修改过的页面（标记为脏页）会被写回磁盘，确保所有更改都被保存。

缓冲池在 "homework1" 项目中扮演着重要角色，它通过在内存中暂存和管理页面来优化文件的读写操作。这种方法减少了对磁盘的直接访问，提高了数据处理的效率，并允许在内存中进行数据的临时修改。

四、测试

使用gtest进行了如下的测试

1. 基础的单页读写操作：创建文件并写入指定的buf，重定位文件头并读取buf大小的字节，看写入和读取是否一致
2. 多页的读写：写入跨多页的buf，重定位到文件头并读取buf大小的数据，看跨页情况下数据是否一致
3. 读取的数据量大于文件的size：返回文件大小的size，而不是我们让其读取的数据量。
4. 多页读写：测试了`size=2<<20`大小的数据量读写
5. 文件空洞：在读取文件时，如果遇到空洞，会自动将其填充为0x00。

```
wyy:build$ ./homework1_test
[==========] Running 7 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 7 tests from WrapperTest
[ RUN      ] WrapperTest.BasicReadWriteTest
[       OK ] WrapperTest.BasicReadWriteTest (0 ms)
[ RUN      ] WrapperTest.ReadWriteCrossPage
[       OK ] WrapperTest.ReadWriteCrossPage (0 ms)
[ RUN      ] WrapperTest.ReadBelowFileEnd
[       OK ] WrapperTest.ReadBelowFileEnd (0 ms)
[ RUN      ] WrapperTest.ReadBiggerThanFileEnd
[       OK ] WrapperTest.ReadBiggerThanFileEnd (0 ms)
[ RUN      ] WrapperTest.Cross3Pages
[       OK ] WrapperTest.Cross3Pages (0 ms)
[ RUN      ] WrapperTest.BigReadWrite
[       OK ] WrapperTest.BigReadWrite (1867 ms)
[ RUN      ] WrapperTest.FileHole
[       OK ] WrapperTest.FileHole (0 ms)
[----------] 7 tests from WrapperTest (1868 ms total)

[----------] Global test environment tear-down
[==========] 7 tests from 1 test suite ran. (1868 ms total)
[  PASSED  ] 7 tests.
```