open, read, write这一套系统api

read 函数 在缺页的情况下，操作系统 内核层 将数据从硬盘中读取到内存，在内核层将数据拷贝至应用层 buffer，

write 函数 将应用层buffer 拷贝至 内核层 ，再做处理

优化思路

1. 最大化利用页缓存
2. 减少系统api调用次数

第一点很容易理解，尽量让每次IO操作都命中页缓存，这比操作磁盘会快很多，第二点提到的系统api主要是read和write，由于系统调用会从用户态进入内核态，并且有些还伴随这内存数据的拷贝，因此在有些场景下减少系统调用也会提高性能

1：对于思路1

readahead

readahead是一种非阻塞的系统调用，它会触发操作系统将文件内容预读到页缓存中，并且立马返回，函数原型如下

```
ssize_t readahead(int fd, off64_t offset, size_t count);
```

说明：在 硬盘 io 不频繁的时候，调用 readahead 由内核层 预先将数据读取到缓存，一段时间后，再使用 read 就会快一点

2：对于思路2

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，函数原型如下

```
void *mmap(void *addr, size_t length, int prot, int flags, int fd,
off_t offset);
```

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，而系统会自动回写脏页面到对应的文件磁盘上，即完成了对文件的操作而不必再调用

read,write等系统调用函数。如下图所示