说明文档

## 思路

1. 在打开目标文件的时候要做两个判断：目标文件是否存在；若存在，是文件还是目录。因为如果是目录的话，写操作的对象需要改变一下。
2. 判断目标文件是否存在可以利用open()的flag---O\_CREAT来实现，如果不存在则会创建新文件，使用O\_CREAT同时需要设置第三个参数---文件访问权限。
3. 此时我发现目标文件为目录时，open()一直出错返回-1，一直找不到原因，后来在网上才找到解释“If the named file already exists as a directory, then if O\_CREAT is specified, open() must fail because of the semantics of O\_CREAT”大概就是说用了O\_CREAT如果是打开目录就会报错。后来学会利用errno来看错误原因，并且直接利用errno来判断是否是目录再做不同的操作。
4. 写tcp.c这个文件的时候看了老师在ch3/ex01里面给的源码。发现里面的读写操作比较复杂，嵌套了两个循环，思路是如果一次写入没有达到符合的字节数，就继续读剩下的字节直到写入符合的字节数，然后再这样读写直到文件读完。
5. 我自己写的时候则是设计成：每次如果没有写够读到的字节数，则设置offset到写之前的位置，重新再写，直到写入符合的字节数。
6. 写tcpm.c前面打开文件的操作和tcp.c一样。后面利用内存映射一片区域读写则是才看了ch3/ex01里面给的源码。大致思路是：先用fstat()获得源文件大小，lseek()为目标文件分配与源文件大小相同的空间，然后用mmap()分别为源文件和目标文件映射一片内存，用void \*类型的指针start、end来分别标识内存区域，用memcpy复制，最后用munmap关闭映射。这里不太理解的是，为什么用lseek()为目标文件分配空间时分配sb.st\_size-1，然后后面又写了一个字节进去呢？不能直接分配sb.st\_size吗？

## 总结

unix编程用到很多系统调用，这些函数已经是很原子级的操作了，因此以往调试的方法不太奏效，应该多利用函数返回值，和errno来确定出错原因。需求帮助可以多看源码，多用命令行命令man command 来获得帮助信息。

## 运行截图



