没有弄清楚semop对信号量集进行操控时，第三个参数需要指定整个信号量集的大小（错误！实际上是操作集的大小）

semctl使用SETALL功能时，没有注意到arg.val需要指向赋值来源的数组

然后咋改都不行……

后来采用小步快跑

1、实验二

重拾

2、多线程使用共享内存

完成；这期间，由于局部变量和全局变量sharedmemory同名且冲突，因此出现了段错误

3、多进程使用共享内存

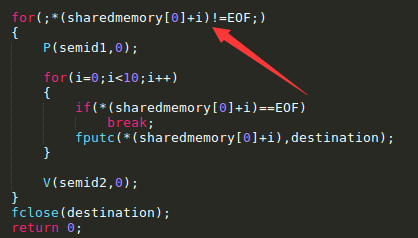
完成；这期间，因为主进程没有waitpid等待子进程，导致子进程没有完成任务便被回收

4、多进程使用共享内存读写文件（main2.c）

完成；中间发现文件的EOF是不用自己手动写入的，而是用fclose就能被系统自动添加

5、共享内存改成大空间（main3）

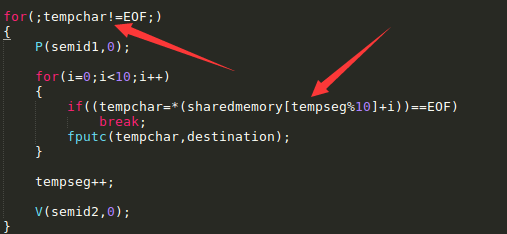
完成；中间对for循环的控制上出现了问题



一开始没有加上这句，导致了问题的出现

6、共享内存改成循环缓冲（main4）

完成



一开始没有引入tempchar，而是在最外层for循环的判断部分使用了指针，导致i的值在初始化为0和跳出内层循环后保留原值间难以区分，使用更优雅的办法解决，当然也可以进一步改进这波循环

7、使用System V改进信号灯集合，实现快速环形缓冲（main5）

完成；中间有些不细致的错误

但是后来发现问题8恰恰出现在7之上，因为两个信号灯的文件能拷贝大文件

8、解决超过一百字后文件从头开始写的bug（main7）

使用控制变量法发现共享内存的总大小是多少，文件能写的字数上限就是多少（也不可能是信号灯的问题），因此改用链表实现循环缓冲，看看是否与之前的取模操作有关

使用链表实现循环缓冲之后，情况照旧（main6）

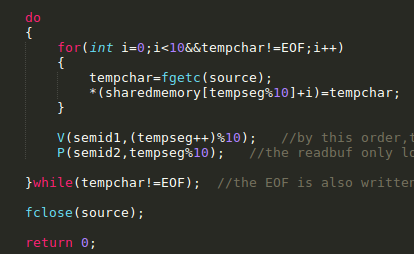
再次实验，与缓冲区是否为2的幂无关，与文件的打开方式无关

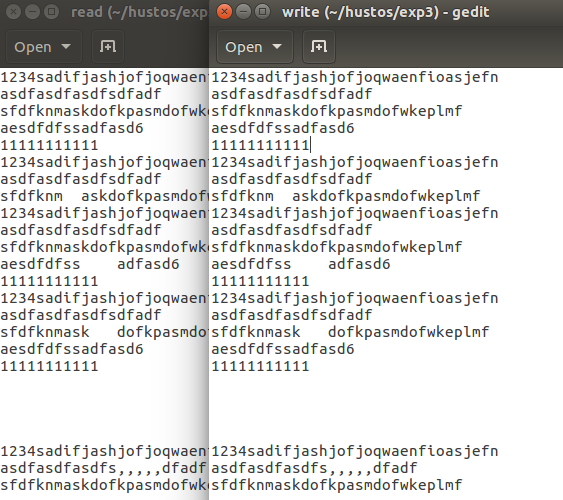
再次实验，与使用的读写函数，信号灯的初值无关

使用小步快跑的思想，发现错误出现在main4到main5的迭代过程中

最后仔细检查接口，发现semop函数的参数3使用错误，误以为是semid集合的大小，实则是操作结构的大小

修改后，发现出现随机的读写错误，经过推导发现是信号灯顺序的问题，仍可以保留初值为1的信号灯，调整顺序后结果正确

  
早先的顺序，会导致writebuf没来得及读取第一遍的写入时，就重新写入并阻塞



成功后的拷贝效果

后续可以根据老师要求，使用exec()，读取特定文件名输入等方式改进

修缮的lab3

加入了读写二进制文件的功能，并且输入输出文件名可以在控制台配置，由于二进制文件中值为-1的字节不一定是EOF，因此改用feof(filepointer)的方法从写内存的部分判断是否已经到达末尾

交由老师检查时发现了几个问题，第一次是通过diff命令发现两个图片不完全相等，仔细检查后发现是由于读写部分执行速度不相同的问题，单纯使用一个标志位，容易出现读还未执行完，就收到执行完的写修改好的标志位，导致提前退出，对比文件大小也会发现写出来的文件少了一些

然后，使用了大量的标志变量，既可以标识文件结束所写到的内存节，又可以标识内存节的具体位置，还可以标识这个位置是否为0

修缮小bug之后交给老师检查，发现了要求多进程但自己在晚上脑抽从原来的多进程改写成了多线程的笑话

然后，修改成多进程交给老师检查，这一次很顺利，但检查时发现自己并没有为标志变量设置共享内存，也就是标志变量只是不同的进程私有，但程序正常退出了，检查发现是自己在移植过程中，main函数的命令行参数没有设置好，导致实质上读写进程都整了空文件，而自己并没有设置打开文件失败时的提示信息，加上原来的输出图片没有删除，产生了修改成功的错觉

重新改写后，采用分块移植的方法，最终多线程移植到多进程成功