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

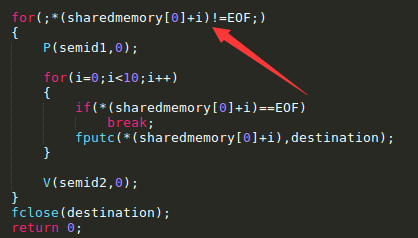
完成；这期间，因为主进程没有waitpid等待子进程，导致子进程没有完成任务便被回收

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）

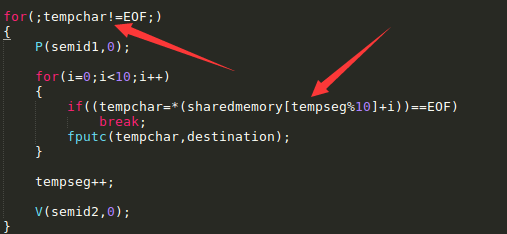
完成；中间对for循环的控制上出现了问题



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）

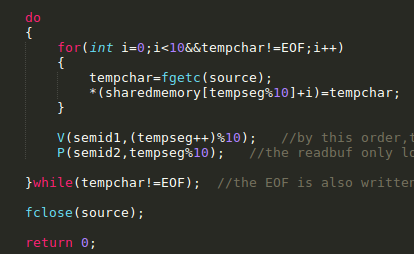
再次实验，与缓冲区是否为2的幂无关，与文件的打开方式无关

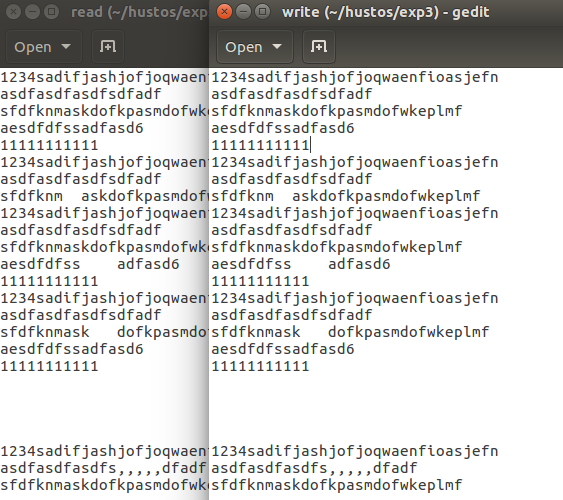
再次实验，与使用的读写函数，信号灯的初值无关

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