# 测试准备：

1. Ramdisk制作4G的Z盘，建立Z:\Test目录
2. C盘建立C:\Test目录
3. 直接运行io.exe

代码路径：<https://10.10.20.204:8080/svn/pet/trunk/nonTOF-PET/Software/TestIO>

# 测试场景一

一次性写入2个1G文件，再顺序读2个文件，读完即删除。

顺序写入a1,a2，再读出a1，删除a1，读出a2，删除a2

# 测试场景二

写入3个1G文件，顺序写入a1,读出a1，删除a1；再写入a2，读出a2，删除a2…….

# 测试结论

1. 保持在现有buffer写读磁盘文件机制+Ramdisk，300ms左右

写读完文件后，及时删除文件。

1. 改变现有buffer写读为内存映射方式+磁盘，写和读在1.5s内。
2. 使用直接IO写读文件+ Ramdisk，效率稳定， 写和读都在300ms左右。
3. 使用直接IO写读文件+磁盘，效率稳定，写和读平均都在8秒左右
4. 保持在现有buffer写读磁盘文件机制， 写和读都在1-50s左右

写读完文件后，及时删除文件。

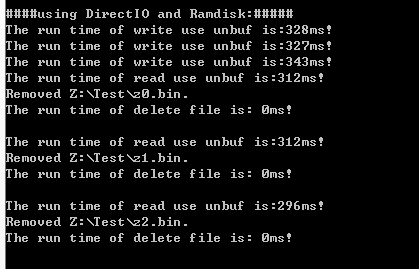
1. 改变现有buffer写读为内存映射方式+Ramdisk，写和读在900ms左右。

写读完文件后，及时删除文件。

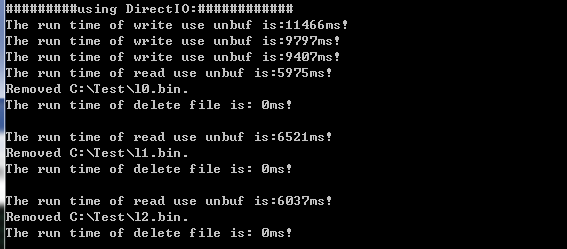
建议使用1方式，其次2,3。

# 测试场景一结果

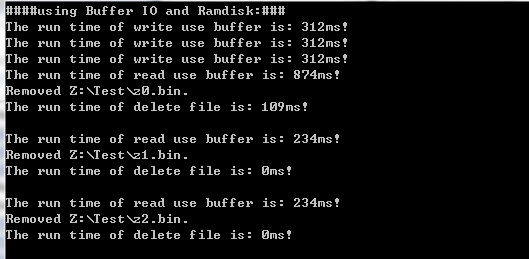
## 使用直接IO写读文件+ Ramdisk



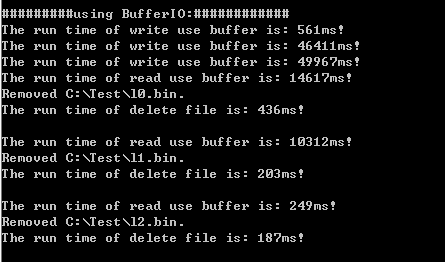
## 使用直接IO写读文件+磁盘



## buffer写读文件机制+Ramdisk

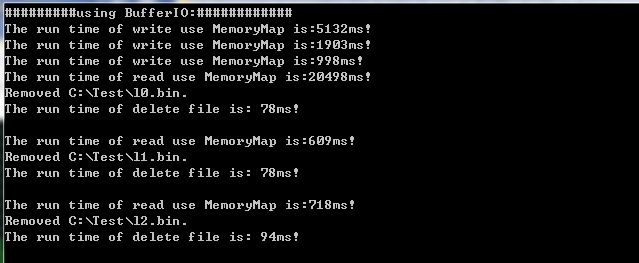


## buffer写读磁盘文件机制

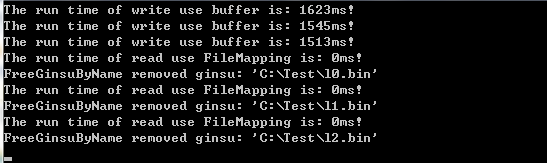


## 内存映射写读磁盘文件机制

使用自定义CMemoryMapWriter, CMemoryMapReader进行写读



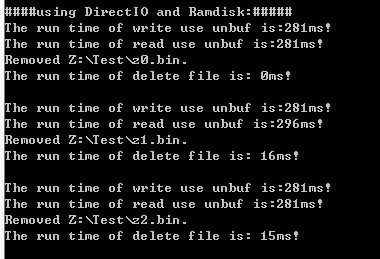
使用Ginsu进行内存映射进行写读



# 测试场景二结果：

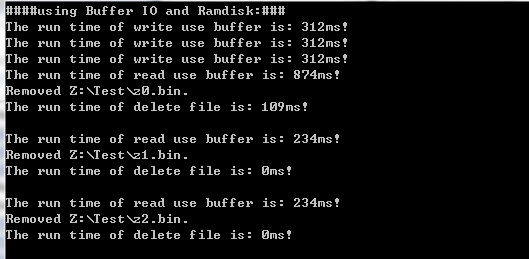
## 直接IO写读磁盘文件或者写读文件+ramdisk

不管写读文件后，是否删除文件，结果相同，读写都在300ms左右

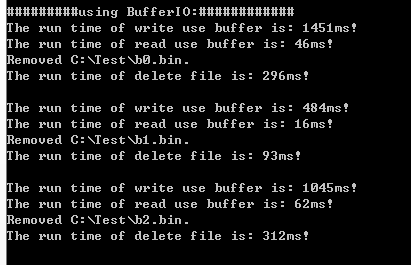


## buffer写读文件机制+Ramdisk

不管写读文件后，是否删除文件，结果相同

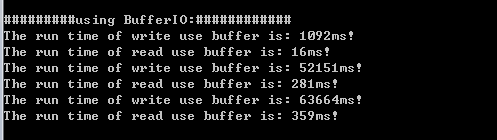


## buffer写读磁盘文件机制，写读文件后，删除文件



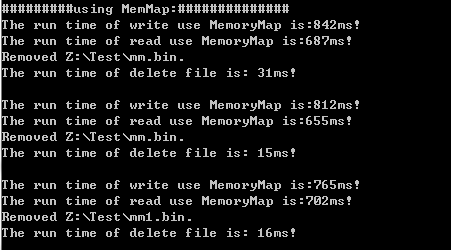
## buffer写读磁盘文件机制，写读文件后，不删除文件

系统缓存换出换入影响到接下来的文件的写和读的效率



## 内存映射写读+Ramdisk

不管写读文件后，是否删除文件，结果相同



## 内存映射写读磁盘

