一、 目的

1. 熟悉类UNIX系统的I/O设备管理
2. 熟悉MINIX 块设备驱动
3. 熟悉MINIX RAM盘

二、实验过程

修改/usr/src/minix/drivers/storage/memory/memory.c ，增加默认的用户RAM盘数：RAMDISKS=7

重新编译内核，重启reboot。

创建设备mknod /dev/myram b 1 13，查看设备是否创建成功输入 ls /dev/ | grep ram。

实现buildmyram.c，将KB单位修改成MB。

```
#define MFACTOR 1048576
```

编译buildmyram.c文件，然后执行命令： ./buildmyram 700 /dev/myram。创

建一个700M的RAM盘。

在ram盘上创建内存文件系统，mkfs.mfs /dev/myram。

将ram盘挂载到用户目录下，mount /dev/myram /root/myram,查看是否

挂在成功：输入df。

三、 内容与设计思想

编写测试文件

为了减小主机操作系统的缓存机制造成的误差，文件总大小越大越好。选用300MB的文件大小

lseek函数定义为off_t lseek(int filde,off_t offset ,int whence);

可以采用lseek重新定位文件指针到一个随机的位置；在主函数中设置srand((**unsigned**)time(**NULL**));在默认情况下随机种子来自系统时钟。如果想在一个程序中生成随机数序列，需要至多在生成随机数之前设置一次随机种子。

顺序读写时，默认文件指针自动移动，当读到文件末尾时，可以用lseek返回文件头。

SEEK_SET表示从文件开头位置往后移动offset个字节

write_file函数如下,read_file函数与其类似

```
void write_file(int block, bool israrand, char *filepath)
{
    fp=open(filepath, O_CREAT | O_RDWR | O_SYNC, 0755);

      int i = 0;
      for (i=0; i < writetime; i++)
      {
          if((res=write(fp, examtext, block))!=block) {
              printf("Error writing to the file.\n");
          }
          //
printf("Wrote %d bytes to the file.\n", res);
          if (israrand) //如果是随机写
              lseek(fp, rand() % filesize, SEEK_SET);
      }
      lseek(fp, 0, SEEK_SET);
}
```

打开文件设置为

```
 ramfp = open("/root/myram/read.txt", O_CREAT | O_RDWR |
O_SYNC, 0755);
       diskfp = open("/usr/read.txt", O_CREAT | O_RDWR |
O_SYNC, 0755);
```

O_RDWR 以可读写方式打开文件

O_SYNC 以同步的方式打开文件.利用O_SYNC参数使得write/read操作为同步模式。

O_CREAT 若欲打开的文件不存在则自动建立该文件.

0755，4位分别代表全部用户（all），文件用户（user)、同组用户(group)和其他用户(other)的权限。
0对应的是对全部用户的权限分配。

7 （十进制）= 111 （二进制）（可以读， 可以写， 可以执行）
5 （十进制）= 101 （二进制）（可以读， 不可以写， 可以执行）
2 （十进制）= 010 （二进制）（不可以读， 可以写， 不可以执行）
0 （十进制）= 000 （二进制）（不可以读， 不可以写， 不可以执行）

0755中的0，相当于对全部用户的缺省权限赋为： 不可以读， 不可以写， 不可以执行
00777就是对文件有读写和执行的权限，也就是对文件是全部控制的。

给每个进程分配独立的文件来读写

```
 for (i = 0; i <=Fork; i++){
        if (fork() == 0)
              {
         //随机写
         //write_file(block, true, filepathDN[i]);
         // write_file(block, true, filepathRN[i]);
        //顺序写
         // write_file(block, false, filepathDN[i]);
          write_file(block, false, filepathRN[i]);
         //随机读
         //read_file(block,true,filepathDN[i]);
         //read_file(block,true,filepathRN[i]);
          //顺序读
          // read_file(block,false,filepathDN[i]);
          // read_file(block,false,filepathRN[i]);
          exit(1);
         }
       }
//等待所有子进程结束
           while (wait(NULL) != -1);
```

没有最后一行代码，就会不等进程结束，这样算出来的时间就有较大偏差

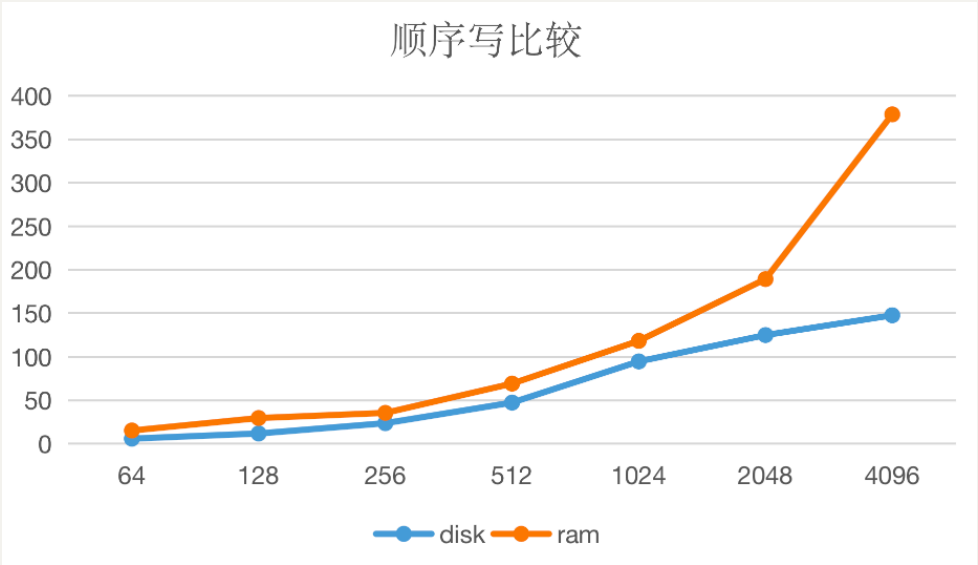
计算速度时，先利用gettimeofday得到的endtime和starttime计算时间差，精确到us

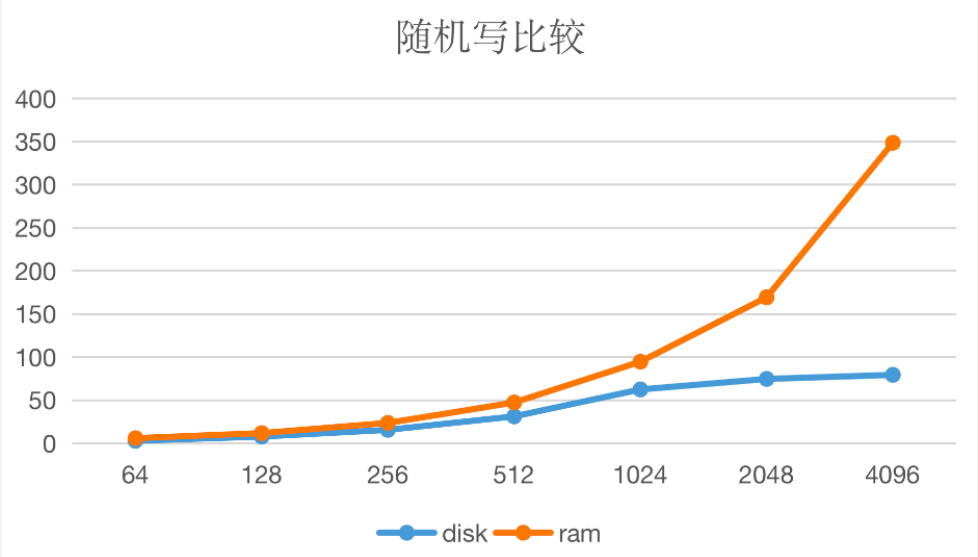
然后计算平均每个进程所用的时间，并转到s级

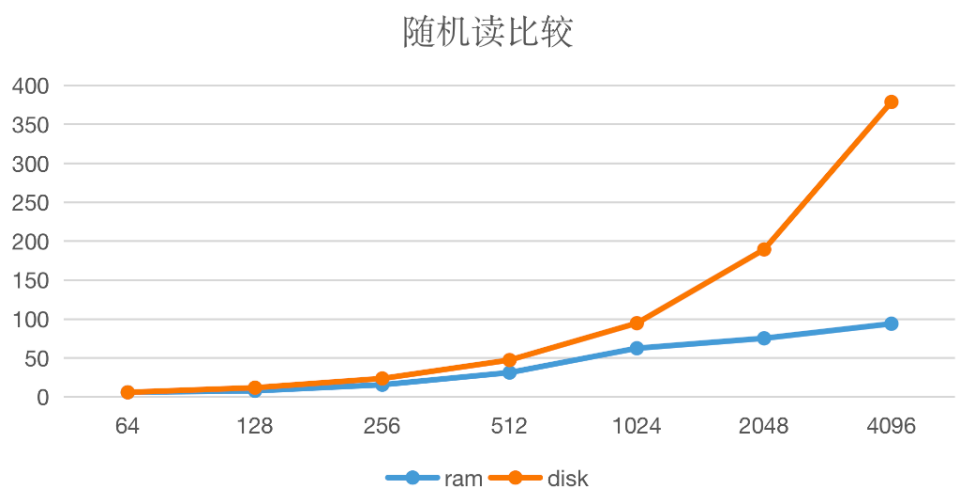
读写速度以MB/s为单位，所以还要对文件大小进行转化

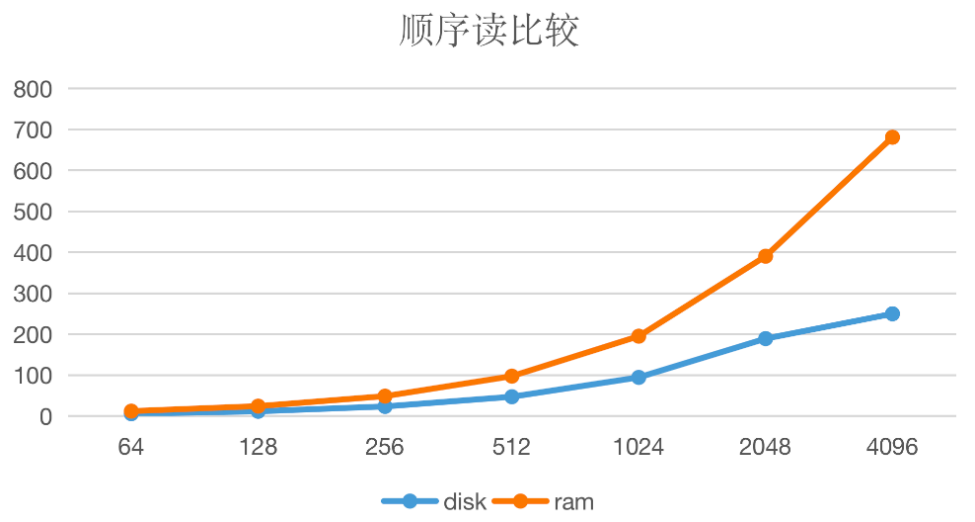
```
 long alltime=(endtime.tv_sec-starttime.tv_sec)*1000+
(endtime.tv_usec-starttime.tv_usec)/1000;
   double eachtime = (alltime ) / (double)writetime /
Fork;
  double timeuse = eachtime / 1000;
   double block_kB = (double)block / 1024.0;
   //MB/s
   double readspeed = block_kB / timeuse/1024.0;
    double writespeed = block_kB / timeuse/1024.0;
```

四、结果









五、总结

本次实验中，我熟悉了类UNIX系统的I/O设备管理，熟悉了MINIX 块设备驱动，熟悉了MINIX RAM盘。通过测试在RAM盘和disk盘下的随机读，顺序读，随机写，顺序写速度，对比在不同的blocksize下的性能，可以发现RAM盘性能高于DISK盘，特别是

随机读写性能。实验过程中，每组的读写需要反复持续一段时间，采用多进程并发执行，每次读写独立的文件，取平均值来计算读写速度。