华东师范大学数据学院上机实践报告

课程名称： 操作系统　　**年级：** 大二　　**上机实践成绩：**

指导教师：　　**姓名：** 沈小奇

上机实践名称：　　**学号：** 10185501401　　**上机实践日期：**

上机实践编号：

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、 **内容与设计思想**

测试 RAM 盘和 DISK 盘的文件读写速度，分析其读写速度 差异原因（可用图表形式体现在实验报告中）。

三、 **使用环境**

Minix，mobaxterm

四、 **实验过程**

增加 RAM 盘：

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

```
#define 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 初始化工具（用于分配容量）。

```
# mknod /dev/myram b 1 13
# ls /dev/ | grep ram
myram
ram
ram0
ram1
ram2
ram3
ram4
ram5
#
```

参考/usr/src/minix/commands/ramdisk/ramdisk.c，实现 buildmyram.c，但是需要将 KB 单位修改成 MB。

fprintf(stderr, "usage: %s <size in MB> [device]\n",　　//修改为 MB
#define KFACTOR 1048576　//修改为 2^20

编译 buildmyram.c 文件，然后执行命令： buildmyram <size in MB> /dev/myram。创建一个 RAM 盘。

```
# ./buildmyram 128 /dev/myram
size on /dev/myram set to 128MB
```

创建了 128MB 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。

```
# mount /dev/myram /root/myram
/dev/myram is mounted on /root/myram
# df
Filesystem    512-blocks       Used      Avail %Cap Mounted on
/dev/myram         262144       4144     258000   1% /root/myram
/dev/c0d0p0s0      262144      77512     184632  29% /
none                    0          0          0 100% /proc
/dev/c0d0p0s2    33566464    4880216   28686248  14% /usr
/dev/c0d0p0s1     8114176      84968    8029208   1% /home
none                    0          0          0 100% /sys
#
```

注：重启后用户自定义的 ram 盘内容会丢失，需要重新设置大小，创 建文件系统，并挂载。

在完成上述操作之后即可进行主要代码的编写。

思路：

2 个 for 循环嵌套，最外层循环考虑 block 的增长，我编写的呈 4 倍增长趋势，第二层循环包含并发数的增加，测试随着并发数的增加，内存、磁盘的读写性能有何变化。

在第二层循环中创建子进程，在创建进程前计时，并且令每个子进程读/写不同文件（以避免不同进程操作一个 block，降低了 cache Miss 的命中率）。

调用 read_file 或 write_file 函数，进入函数以后，考虑是顺序的还是随机的，若是随机读/写，则需要随机改变读/写指针，用到了 lseek 函数。

内循环结束后结束计时，计算总时间，再通过数学公式计算延迟、吞吐率等最终打印出来。

代码部分：

```
#include <stdio.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <stdbool.h>
#include <unistd.h>
#include <time.h>
```

```
#include <string.h>
#include <sys/wait.h>


#define Concurrency 17 //并发数
#define writetime 12000
#define readtime 8000
#define Blocksize 4096
#define filesize (50 * 1024 *1024 )//文件大小
#define maxline (100 * 1024+26 ) //大于等于块的大小
#define readbuff (10 * 1024 * 1024)


char examtext[maxline] = "abcdefghijklmnopqrstuvwxyz";//用来写的
struct timeval starttime, endtime, spendtimeSpec;

char buff[maxline];

/*写文件:打开文件，判断返回值，如果正常打开文件就判断是否随机写，进行写操作*/
void write_file(int blocksize, bool isrand, char* filepath)
{
	int fp=open(filepath,O_RDWR | O_CREAT | O_SYNC    , 0755);
	int i=0;

	if(fp >0){
		for(;i<writetime;i++){
		    int x=write(fp,examtext,blocksize);
		    if(x<0){
		       printf("write error!");
		       break;
		   }
		     if(!isrand)
		          lseek(fp,rand()%filesize,SEEK_SET);
		}



	}
	else {printf("open error!");}
	lseek(fp,0,SEEK_SET);

}

/*读文件:打开文件，判断返回值，如果正常打开就判断是否随机读，进行读操作*/
void read_file(int blocksize, bool isrand, char *filepath) {
	//to do....
	int fp=open(filepath,O_RDONLY);

	int i = 0;
```

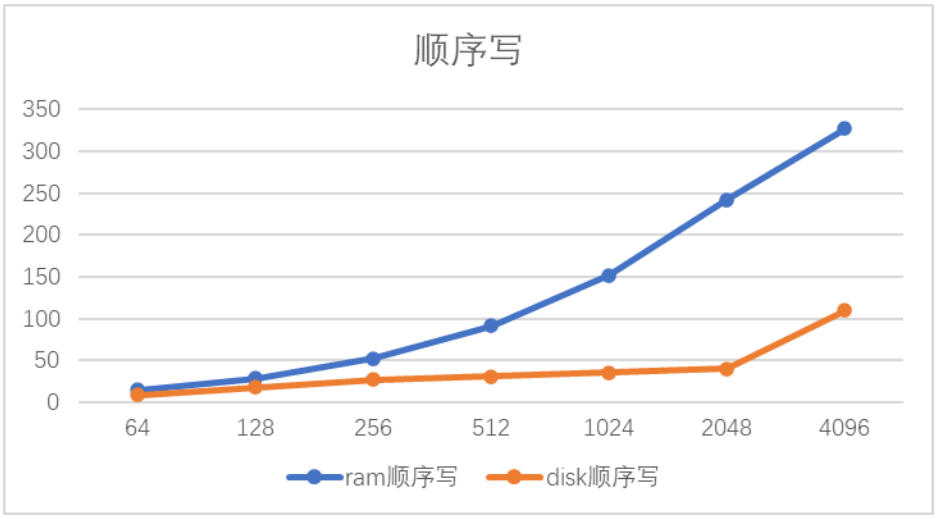
```
            if(fp>0){
               for (; i < readtime; i++) {

                    int x=read(fp, buff, blocksize);
                    if(x<0){
                      printf("read error!\n");
                      break;
                  }
                    if (!isrand)//如果是随机读
                         lseek(fp, rand() % filesize, SEEK_SET);//文件偏移量设为 offset

               }
               }
               else {printf("open error!");}
               lseek(fp, 0, SEEK_SET);//重置指针

      }
      //计算时间差，在读或写操作前后分别取系统时间，然后计算差值即为时间差。
      long get_time_left(struct timeval starttime, struct timeval endtime)
      {
        //to do....
        long spendtime = 1000 * (endtime.tv_sec - starttime.tv_sec) + (endtime.tv_usec -
starttime.tv_usec)/1000; /* ms */
          return spendtime;
      }

      /*主函数：首先创建和命名文件，通过循环执行 read_file 和 write_file 函数测试读写差
异。
      测试 blocksize 和 concurrency 对测试读写速度的影响，最后输出结果。*/
      int main()
      {     srand((unsigned)time(NULL));
            int i=0;


          char *
filepathram[19]={"/home/myram/ram1.txt","/home/myram/ram2.txt","/home/myram/ram3.txt","/ho
me/myram/ram4.txt","/home/myram/ram5.txt","/home/myram/ram6.txt","/home/myram/ram7.txt","/
home/myram/ram8.txt","/home/myram/ram9.txt","/home/myram/ram10.txt","/home/myram/ram11.t
xt","/home/myram/ram12.txt","/home/myram/ram13.txt","/home/myram/ram14.txt","/home/myram/
ram15.txt","/home/myram/ram16.txt","/home/myram/ram17.txt","/home/myram/ram18.txt","/home/
myram/ram19.txt"};


          char *
filepathdisk[19]={"/usr/disk1.txt","/usr/disk2.txt","/usr/disk3.txt","/usr/disk4.txt","/usr/disk5.txt","/u
sr/disk6.txt","/usr/disk7.txt","/usr/disk8.txt","/usr/disk9.txt","/usr/disk10.txt","/usr/disk11.txt","/usr/
disk12.txt","/usr/disk13.txt","/usr/disk14.txt","/usr/disk15.txt","/usr/disk16.txt","/usr/disk17.txt","/us
r/disk18.txt","/usr/disk19.txt"};
```

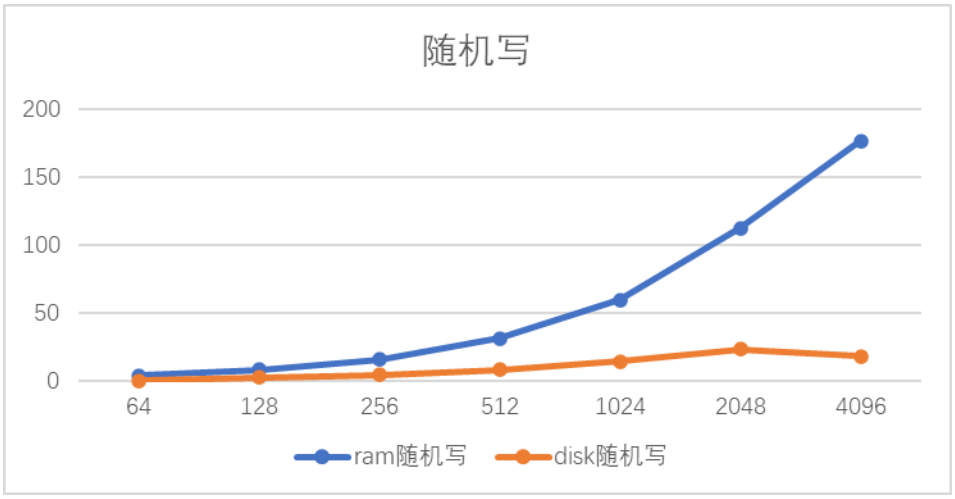
```
for(int j=0;j<maxline;){
   strncat(examtext,"hello",5);//字符串接在后面
   j=j+5;

}



for (int block = 64; block <= Blocksize;)
{
      for ( int concurrency= 1; concurrency <= Concurrency;)
      {
            gettimeofday(&starttime, NULL);
            for (i = 1; i <=concurrency; i++)
            {
                  if (fork() == 0)
                  {

                 //    write_file(block, true, filepathram[i-1]);
                        //ram 顺序写

                   //write_file(block, false, filepathram[i-1]);
                        //ram 随机写

                        read_file(block, true,filepathram[i-1]);
                        //ram 顺序读
                       // read_file(block, false, filepathram[i-1]);
                        //ram 随机读

                  //    write_file(block,true,filepathdisk[i-1]);
                        //磁盘顺序写
                    // write_file(block,false,filepathdisk[i-1]);
                        //磁盘随机写


                       // read_file(block, true,filepathdisk[i-1]);
                        //磁盘顺序读
                     //    read_file(block, false, filepathdisk[i-1]);
                        //磁盘随机读

                        exit(1);
                  }
            }
            while (wait(NULL) != -1)
            {
            }
            gettimeofday(&endtime, NULL);
```

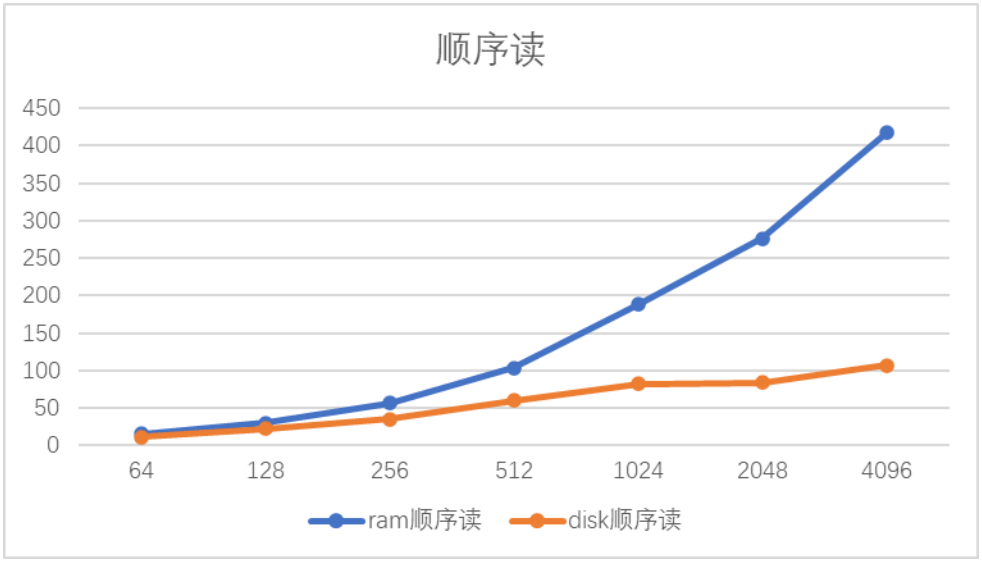
```
                /*等待子进程完成后，获取计算时间，计算读写操作所花时间，延时，吞吐量等*/
    //
                long alltime=get_time_left(starttime,endtime);//单位是毫秒   指代全部时间

                double alltime_s=alltime/1000.0;//单位为秒


                //double latency = (alltime) / (double) writetime / concurrency;

                  double latency = (alltime) / (double)readtime / concurrency;
                //计算延迟（单个 IO 操作的时间），单位为毫秒

                //double file_kB = (double) block * writetime *concurrency / 1024.0; /*  文件大小，单位为 KB */

                double file_kB = (double) block * readtime *concurrency / 1024.0; /*  文件大小，单位为 KB */
                double ops = file_kB / alltime_s / 1024.0; //计算的是吞吐量，文件大小/执行时间，单位为 mb/s

                printf("%d,%d,%f,%f\n", block, concurrency, latency, ops);


                concurrency += 4;
            }
            block *= 2;
        }



    return 0;
}
```

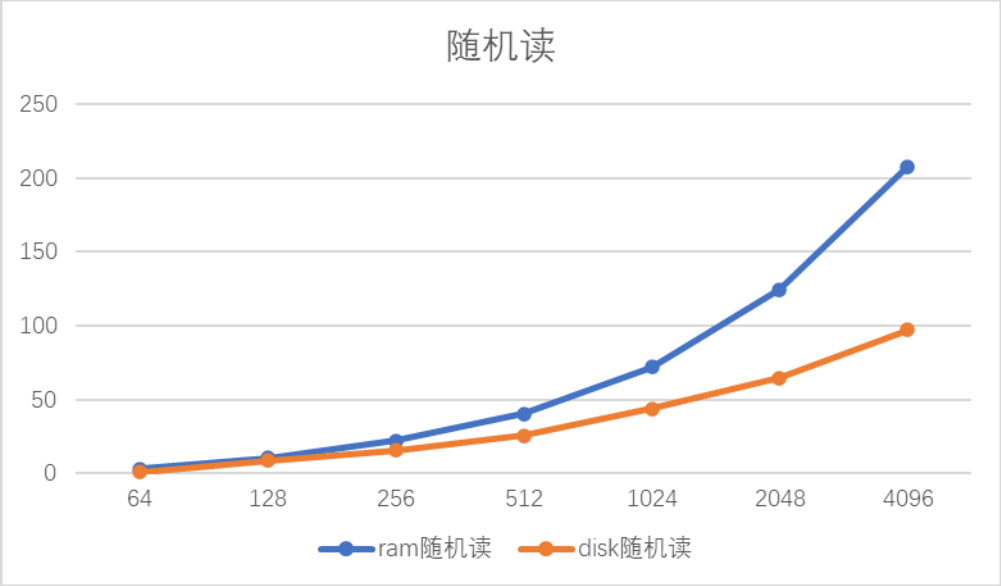
结果示意图：

吞吐率单位 MB/s


顺序写
350
300
250
200
150
100
50
0
64
128
256
512
1024
2048
4096
ram顺序写
disk顺序写


随机写
200
150
100
50
0
64
128
256
512
1024
2048
4096
ram随机写
disk随机写


顺序读
450
400
350
300
250
200
150
100
50
0
64
128
256
512
1024
2048
4096
ram顺序读
disk顺序读



纵向比较：Ram 总体性能要高于磁盘读写，且随着块数增大，ram 盘和 disk 盘的差距也越来越大。

横向比较：Ram 盘->顺序写高于随机写，顺序读高于随机读。

磁盘->顺序写高于顺序写，顺序读高于随机读。

五、总结

难度有一点，在编译过程中因为虚拟机读写太厉害，导致内存空间不足，重新装了虚拟机重新配置，耽误了很长时间。Lab 思路比较清楚，对读写 ram 和磁盘速度上有了更清晰的认识。