

Lab5 文件系统实 习报告

姓名 姜慧强 学号 1801210840
日期 2019.06.02

目录

内容一：总体概述	3
内容二：任务完成情况	3
任务完成列表（Y/N）	3
具体 Exercise 的完成情况	3
内容三：遇到的困难以及解决方法	18
内容四：收获及感想	18
内容五：对课程的意见和建议	19
内容六：参考文献	19

内容一：总体概述

本次实验是操作系统高级课程的第五次实验。在 Lab4 阅读内存管理机制，完成 TLB、Paging 管理、Lazy Loading，主要完成了 Nachos 文件系统相关代码的实践。Nachos 中有两套文件系统，一套是基于 POSIX 的，在 Lab5 之前我们用的都是这套文件系统。另外一套文件系统是还未完全实现的，完善这套文件系统是本 Lab 的主要内容。总体感觉本次 Lab，任务量较大，实践难度较大，是历次实验中的难点。主要完成了扩展文件属性，扩展文件长度（多级索引），实现多级目录（Path 的解析），动态调整文件长度，文件的同步互斥访问，性能优化等子任务。经过本次 Lab 的实践，增强了对文件系统相关原理的认识，提高了对操作系统的整体理解。

内容二：任务完成情况

任务完成列表 (Y/N)

EXER1	EXER2	EXER3	EXER4	EXER5	EXER6	EXER7	C1	C2
Y	Y	Y	Y	Y	Y	Y	Y	N

具体 Exercise 的完成情况

一、文件系统的基本操作

Exercise1

文件系统定义于 `filesys/filesys.h` 文件中，可以看到通过全局变量 `FILESYS_STUB` 来控制 Nachos 具体使用哪种文件系统。当 `FILESYS_STUB` 为真时，使用基于 Unix 的文件系统。`FILESYS_STUB` 为 False 时，则进入本次 Lab 涉及到的文件系统。

在该文件系统中，定义了两个文件私有变量，`freeMapFile`, `directorFile`。分别表示该文件系统下来管理空闲磁盘块的 `FreeMap` 文件，和管理根目录文件的 `DirectorFile`。`FreeMap`, `DirectorFile` 分别保存在第 0 号，第 1 号 Sector 扇区。文件系统初始化的时候，会先去创建二者，在这两者之后才会初始化正常的文件。所以其他文件的扇区都是大于 1 的。

除了 `FreeMapSector`, `DirectorSector` 常量之外，在 `filesys/filesys.cc` 中还定义了

```
#define FreeMapFileSize (NumSectors / BitsInByte)
#define NumDirEntries 10
#define DirectoryFileSize (sizeof(DirectoryEntry) * NumDirEntries)
```

扇区文件大小，根目录文件目录数量，根目录文件文件大小。

在文件系统中还实现了一些函数。

```

FileSystem(bool format);    // Initialize the file system.
bool Create(char *name, int initialSize);
OpenFile* Open(char *name); // Open a file (UNIX open)
bool Remove(char *name);    // Delete a file (UNIX unlink)
void List();                // List all the files in the file system
void Print();               // List all the files and their contents

```

这些函数实现了初始化文件系统，在文件系统中创建名字为 **name** 的文件，打开名字为 **name** 的文件，删除名字为 **name** 的文件，打印出文件的列表，打印出该文件系统的所有信息。

在初始化文件系统 **filesys** 之前，**nachos** 的文件系统需要先初始化 **thread/system.h** 中的 **synchDisk** 变量。该变量是基于同步机制实现的同步磁盘。通过该磁盘来实现文件的互斥读写。在之后的实践中需要扩展该部分。需要注意的是，**SynchDisk** 的初始化需要在 **filesys** 之前。

```

#ifdef FILESYS
SynchDisk *synchDisk;
#endif

```

在 **Nashos** 文件系统中，文件通过文件头，扇区地址 **Sector** 等来定义。文件头的信息在 **bitmap.h**

```

#define HeaderIntNum 3
#define HeaderTimeNum 3
#define MaxFileTimeLen 26
#define MaxFileNameLen 5
#define HeaderStringLen MaxFileNameLen + HeaderTimeNum * MaxFileTimeLen

#define NumDirect ((SectorSize - (HeaderIntNum * sizeof(int) +
HeaderStringLen * sizeof(char))) / sizeof(int))
#define IndirectNum 1
#define SectorInt 32
#define MaxDirectNum NumDirect - IndirectNum
#define MaxFileSize (MaxDirectNum) * SectorSize + IndirectNum * SectorInt *
SectorSize

```

一个 **Sector** 的总长度是 128 个字节，所以 **Direct** 的信息是除了文件头之外的所有空间。因为在后面扩充了文件头的一些字段，所以在这里需要重新计算一些 **Direct** 可用的空间。

```

void FetchFrom(int sectorNumber); // Initialize file header from disk
void WriteBack(int sectorNumber); // Write modifications to file header
int ByteToSector(int offset); // Convert a byte offset into the file
int FileLength(); // Return the length of the file
void Print(); // Print the contents of the file.
void SetCreateTime(); // lab5 set create time
void SetLastVisterTime(); // lab5 set last visiter time
void SetLastModifyTime(); // lab5 set last modified time

```

```

void HeaderInit(char *type, int sector); // lab5 init header set
void setFileType(char* fileType) {
    printf("size of file type %d\n", sizeof type);
    strcmp(fileType, "") ? strncpy(type, fileType, sizeof type - 1) :
    strncpy(type, "None", sizeof type - 1);
    type[sizeof type - 1] = '\0';
    printf("file type %s\n", type);
}

// lab5 set file type
int SectorPos; // lab5 sector position
bool Extend(Bitmap *freeMap, int bytes); // lab 5 extend file length

```

其通过扇区标号从磁盘中获得文件头信息，WriteBack 是将文件头写回磁盘。其他实现的函数与文件系统的基本类似。

Directory 中定义的是目录文件中相关的内容，在 Nachos 中使用 DirectorEntry 来定义一个目录，其中包含，inuse, sector, name 三个初始字段。通过这些字段来实现目录文件的读写，查找空闲区块等操作。和 FileHeader 一样的是其也有 FetchFrom 和 WriteBack 两个函数，其含义也相同，都是把目录文件写入磁盘，和从磁盘中读取目录文件信息。

Openfile 文件中定义了对文件具体内容的读写操作。包括 Seek, Read, Write, ReadAt, WriteAt 等函数。在该文件中读写文件是通过字节数来定义的。

Bitmap 文件是用来定义位图模块，包括一些常量，位图大小，位图文件大小等等。主要实现的函数包括，寻找空闲区块，清除位图，标记指定位置等等。

Exercise2

需要扩展文件描述信息，文件描述信息在 Nachos 中是存储在 filehdr 中，在这里需要更改

FileHeader 类。

```

int numBytes; // Number of bytes in the file
int numSectors; // Number of data sectors in the file
int dataSectors[NumDirect]; // Disk sector numbers for each data
// block in the file
char createTime[MaxFileTimeLen]; // lab5 create time
char lastVisterTime[MaxFileTimeLen]; // lab5 last vister time
char lastModifiedTime[MaxFileTimeLen]; // lab5 last modified time
char type[MaxFileNameLen]; // lab5 file type

```

前面提到，文件头是存储在 Sector 中的，是和 Director 公用一个 Sector，所以当我们要扩充文件头的时候需要相应的更改 directory 的大小。

```

#define HeaderIntNum 3
#define HeaderTimeNum 3
#define MaxFileTimeLen 26

```

```

#define MaxFileNameLen 5
#define HeaderStringLen MaxFileNameLen + HeaderTimeNum * MaxFileTimeLen

#define NumDirect ((SectorSize - (HeaderIntNum * sizeof(int) + HeaderStringLen *
sizeof(char))) / sizeof(int))
#define IndirectNum 1
#define SectorInt 32
#define MaxDirectNum NumDirect - IndirectNum
#define MaxFileSize (MaxDirectNum) * SectorSize + IndirectNum * SectorInt *
SectorSize

```

在 `filehdr.cc` 中定义 `SetTime` 来实现时间设定。

```

void setTime(char *paramName, char *name){
    time_t timep;
    time (&timep);
    struct tm *timeinfo = localtime(&timep);
    strncpy(paramName, asctime(timeinfo), 25);
    paramName[24] = '\0';
    DEBUG('f', "\033[92m %s: %s \n \033[0m", name, paramName);
}

```

而通过 `HeaderInit` 来实现初始化 `FileHdr` 是需要同步的一些参数。

```

void FileHeader::HeaderInit(char *fileType, int sector){
    DEBUG('f', "\033[93m File Type: %s \033[0m\n", fileType);
    SectorPos = sector;
    setFileType(fileType);
    SetCreateTime();
    SetLastModifyTime();
    SetLastVisterTime();
}

```

对文件长度设置进行扩充，最简单的方式就是直接更改 `MaxFileLen` 常量。但这样的做法治标不治本。参考 `FreeMap` 和 `DirectoryFile` 的思路，通过设立一个 `NameFile` 来存储 `Name` 信息，从而一劳永逸的解决文件长度的问题。

```

#define NameSector 2
FileHeader *nameHdr = new FileHeader;
nameHdr->HeaderInit("Name", NameSector);
freeMap->Mark(NameSector);
ASSERT(nameHdr->Allocate(freeMap, FreeMapFileSize));
ameHdr->WriteBack(NameSector);

```

实验结果如下：

执行 `./nachos -f` 初始化文件系统之后，`./nachos -D` 查看文件列表，`./nachos -cp test/big big`


```

    for (int i = 0; i < MaxDirectNum; ++i)
        dataSectors[i] = freeMap->Find();
    dataSectors[MaxDirectNum] = freeMap->Find();
    int indirect_index[SectorInt] ;
    for (int i = 0; i < numSectors - MaxDirectNum; ++i)
        indirect_index[i] = freeMap->Find();
    synchDisk->WriteSector(dataSectors[MaxDirectNum], (char
*)indirect_index);
}

#define NumDirect    ((SectorSize - (HeaderIntNum * sizeof(int) + HeaderStringLen *
sizeof(char))) / sizeof(int))
#define IndirectNum 1
#define SectorInt 32
#define MaxDirectNum NumDirect - IndirectNum
#define MaxFileSize (MaxDirectNum) * SectorSize + IndirectNum * SectorInt * SectorSize

```

通过定义 **MaxDirectNum** 来实现对 **MaxDirect** 的控制。同样在收回文件头空间的时候也是需要分开来讨论的。

```

void
FileHeader::Deallocate(BitMap *freeMap)
{
    if (numSectors <= MaxDirectNum) {
        for (int i = 0; i < numSectors; ++i)
            freeMap->Clear((int) dataSectors[i]);
    } else {
        char *indirect_index = new char[SectorSize];
        synchDisk->ReadSector(dataSectors[MaxDirectNum], indirect_index);
        for (int i = 0; i < numSectors - MaxDirectNum; ++i)
            freeMap->Clear((int) indirect_index[i * 4]);
        for (int i = 0; i < NumDirect; ++i)
            freeMap->Clear((int) dataSectors[i]);
    }
}

```

在实现间接索引之前我们是不能存储大于 **896B** 的文件的，现在通过间接索引，我们可以存储 **4kB** 左右的文件。


```
1.root@1801210840: ~/nachos/nachos_dianti/lab5/nachos_dianti/nachos-3.4/code/filesys (docker)
SectorNumber: 33
Sector: 34
SectorNumber: 34
*)indirect_index);\a
Sector: 35
SectorNumber: 35
)\a
return TRUE;\a
Sector: 36
SectorNumber: 36
-----\a// FileHeader::Deallocate\a// De-allocate all the space allocated for data blocks for this file.\a// \a "freeMap" is the
Sector: 37
bit map of free disk sectors\a//-----\a\avoid \aFileHeader::Deallo
Sector: 38
SectorNumber: 38
cates(Bitmap *freeMap)\a{\a if (numSectors <= MaxDirectNum) {\a for (int i = 0; i < numSectors; ++i)\a
Sector: 39
SectorNumber: 39
freeMap->Clear((int) dataSectors[i]);\a \a } else {\a char
Sector: 40
SectorNumber: 40
*indirect_index = new char[SectorSize);\a synchDisk->ReadSector(dataSectors[MaxDirectNum], indirect_index)
Sector: 41
SectorNumber: 41
)\a for (int i = 0; i < numSectors - MaxDirectNum; ++i)\a
Sector: 42
SectorNumber: 42
freeMap->Clear((int) indirect_index[i * 4]);\a for (int i = 0; i\0\0\0\0

Directory contents:
Id: 0, Name: bigger, InUse: 1, Sector: 8
SectorNumber: 8
FileHeader contents. File size: 4444 File type None File blocks:
sectorPos: 8
CreateTime: Sat Jun 1 14:54:09 2019
LastVisteTime: Sat Jun 1 14:54:09 2019
LastModifyTime: Sat Jun 1 14:54:09 2019
Indirect Index: 16
9 10 11 12 13 14 15 SectorNumber: 16
17
18
19
20
21
22
23
24
25

x IPython: git/spider... #1 x ..bzz/灰度+锐化... #2 x ..etection-ctpn (z... #3 x ..eratingSystem (... #4 x ..bzz/灰度+锐化 (z... #5 x ../code/filesys (do... #6 x / (fzf) #7
```

```
1.root@1801210840: ~/nachos/nachos_dianti/lab5/nachos_dianti/nachos-3.4/code/filesys (docker)
-rw-r--r-- 1 root root 3.8K May 27 11:58 stats.o
-rw-r--r-- 1 root root 700 May 27 11:59 switch.o
-rw-r--r-- 1 root root 1.4K May 27 11:59 switch.s
-rw-r--r-- 1 root root 16K Jun 1 12:51 synch.o
-rw-r--r-- 1 501 dialout 3.6K May 28 06:59 synchdisk.cc
-rw-r--r-- 1 501 dialout 1.9K May 28 06:59 synchdisk.h
-rw-r--r-- 1 root root 9.0K Jun 1 12:53 synchdisk.o
-rw-r--r-- 1 root root 9.4K Jun 1 12:51 synchlist.o
-rw-r--r-- 1 root root 18K Jun 1 12:51 sysdep.o
-rw-r--r-- 1 root root 23K Jun 1 12:51 system.o
drwxr-xr-x 2 501 dialout 4.0K May 28 07:25 test
-rw-r--r-- 1 501 dialout 223 Jun 1 06:54 test.sh
-rw-r--r-- 1 root root 20K Jun 1 12:51 thread.o
-rw-r--r-- 1 root root 7.1K Jun 1 12:51 threadtest.o
-rw-r--r-- 1 root root 7.2K Jun 1 12:51 timer.o
-rw-r--r-- 1 root root 11K Jun 1 12:51 translate.o
-rw-r--r-- 1 root root 4.8K May 27 11:58 utility.o
→ 1801210840 filesys git:(develop) x ll test
total 136K
-rw-r--r-- 1 501 dialout 112K May 28 06:46 PI.100.000.TXT
-rw-r--r-- 1 501 dialout 608 May 28 06:59 big
-rw-r--r-- 1 501 dialout 152 May 28 06:59 medium
-rwxr-xr-x 1 501 dialout 753 May 28 06:46 my_test_script1.sh
-rw-r--r-- 1 501 dialout 38 May 28 06:59 small
-rwxr-xr-x 1 501 dialout 479 May 28 06:46 test_exercise_2-1.sh
-rwxr-xr-x 1 501 dialout 1014 May 28 06:46 test_exercise_3.sh
→ 1801210840 filesys git:(develop) x vim test/bigger
→ 1801210840 filesys git:(develop) x ll test
total 160K
-rw-r--r-- 1 501 dialout 112K May 28 06:46 PI.100.000.TXT
-rw-r--r-- 1 501 dialout 608 May 28 06:59 big
-rw-r--r-- 1 root root 22K Jun 1 14:21 bigger
-rw-r--r-- 1 501 dialout 152 May 28 06:59 medium
-rwxr-xr-x 1 501 dialout 753 May 28 06:46 my_test_script1.sh
-rw-r--r-- 1 501 dialout 38 May 28 06:59 small
-rwxr-xr-x 1 501 dialout 479 May 28 06:46 test_exercise_2-1.sh
-rwxr-xr-x 1 501 dialout 1014 May 28 06:46 test_exercise_3.sh
→ 1801210840 filesys git:(develop) x truncate -s 5555 test/bigger
→ 1801210840 filesys git:(develop) x ll test
total 144K
-rw-r--r-- 1 501 dialout 112K May 28 06:46 PI.100.000.TXT
-rw-r--r-- 1 501 dialout 608 May 28 06:59 big
-rw-r--r-- 1 root root 5.5K Jun 1 14:21 bigger
-rw-r--r-- 1 501 dialout 152 May 28 06:59 medium
-rwxr-xr-x 1 501 dialout 753 May 28 06:46 my_test_script1.sh
-rw-r--r-- 1 501 dialout 38 May 28 06:59 small
-rwxr-xr-x 1 501 dialout 479 May 28 06:46 test_exercise_2-1.sh
-rwxr-xr-x 1 501 dialout 1014 May 28 06:46 test_exercise_3.sh
→ 1801210840 filesys git:(develop) x
```

Exercise 4

实现多级目录，这个和前面的拓展文件长度不同，只需要记录 **Path**，然后依次划分找到相应的各级地址。

```
int type;           // lab 5 multi-dir
char path[20];      // lab 5 multi-dir
```

通过文件类型来判断是目录文件，还是普通文件，而 **Path** 则表示其绝对路径。在改完数据结构之后，需要对 **Directory** 中 **FindIndex** 等函数进行改造。

```
char *Directory::FindName(char *name){
    char fileName[FileNameMaxLen + 1];
    int pos = -1;
    for (int i = strlen(name) - 1; i >= 0; --i)
        if (name[i] == '/') {
            pos = i + 1;
            break;
        }
    if (pos == -1) pos = 0;
    int j = 0;
    for (int i = pos; i < strlen(name); ++i)
        fileName[j++] = name[i];
    fileName[j] = '\0';
    char *nameFile = fileName;
    return nameFile;
}
```

通过 **FindName** 来从 **Path** 解析到 **Name**。

```
int Directory::GetType(char *name){
    for (int i = 0; i < tableSize; i++) {
        DEBUG('f', "%s %d %d\n", table[i].name, table[i].inUse,
table[i].sector);
        if (table[i].inUse && !strncmp(table[i].name, name, FileNameMaxLen))
            return table[i].type;
    }
    return -1;           // name not in directory
}
```

通过 **GetType** 来获得文件的类型。为测试在 **main** 中创建 **-mkdir** 字段。

```
void CreateDir(char *name){
    printf("\033[93m Begin Create Dir \n\033[0m");
    fileSystem->Create(name, -1);
}
```

```
1.root@1801210840: ~/nachos/nachos_dianti/lab5/nachos_dianti/nachos-3.4/code/filesys (docker)
LastModifyTime:
File contents:
Id: 1, Name: test/small , InUse: 1, Sector: 11, Type: 1, Path: test/small
SectorNumber: 11
FileHeader contents. File size: 38. File type None File blocks:
sectorPos: 11
CreateTime: Sun Jun 2 05:24:18 2019
LastVisterTime: Sun Jun 2 05:24:18 2019
LastModifyTime: Sun Jun 2 05:24:18 2019
12
File contents:
SectorNumber: 12
This is the spring of our discontent.\a
Directory contents:
Id: 0, Name: test , InUse: 1, Sector: 10, Type: 0, Path: test
SectorNumber: 10
FileHeader contents. File size: -1. File type File blocks:
sectorPos: 10
CreateTime:
LastVisterTime:
LastModifyTime:
File contents:
Id: 1, Name: test/small , InUse: 1, Sector: 11, Type: 1, Path: test/small
SectorNumber: 11
FileHeader contents. File size: 38. File type None File blocks:
sectorPos: 11
CreateTime: Sun Jun 2 05:24:18 2019
LastVisterTime: Sun Jun 2 05:24:18 2019
LastModifyTime: Sun Jun 2 05:24:18 2019
12
File contents:
SectorNumber: 12
This is the spring of our discontent.\a
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!
Ticks: total 11230, idle 11000, system 230, user 0
Disk I/O: reads 22, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
Cleaning up...
→ 1801210840 filesys git:(develop) x
x IPython: git/sp... #1 x ..bzz/灰度+锐... #2 x ..etecion-ctp... #3 x ..eratingSyste... #4 x ..bzz/灰度+锐... #5 x ../code/filesys... #6 x / (zsh) #7 x ..op/git/spider... #8
```

Exercise 5

在文件 `filesys/filehdr.cc` 中实现 `Extend` 函数，通过比较增长前后扇区大小来判断是否需要扩充文件长度。

```
bool FileHeader::Extend(BitMap *freeMap, int bytes){
    numBytes = numBytes + bytes;
    int initialSector = numSectors;
    numSectors = divRoundUp(numBytes, SectorSize);
    if (initialSector == numSectors)
        return TRUE;
    if (freeMap->NumClear() < numSectors - initialSector) {
        return FALSE;
    }
    for (int i = initialSector; i < numSectors; ++i){
        int NowSector = freeMap->Find();
        dataSectors[i] = NowSector;
    }
    return
```

除此之外，修改 `openfile` 的 `WriteAt` 函数，但结束位置超过文件长度，则调用 `Extend` 的函数来看看需不需要扩充 `Sector`。

测试时，用-t 循环 20 次。因为一次是 10B，一个扇区是 128B，所以理论上第一次循环和 第 12 次循环是需要扩充 Sector 的。实验结果也吻合理论分析。

```

1. root@1801210840: ~/nachos/nachos_diant/nachos/nachos-3.4/code/filesys (docker)
SectorNumber: 11
Last Vister Time: Sun Jun  2 07:47:17 2019
MaxMaxFileSize: 4992
Last Vister Time: Sun Jun  2 07:47:17 2019
Last Modified Time: Sun Jun  2 07:47:17 2019
Writing bytes 120 to 10, Need: 200
Position: 120, FileLength: 120, NumBytes: 10
SectorNumber: 0
Init Open file Sector 0
Reading 128 bytes at 0, from file of length 128.
LastSector: 0
MaxMaxFileSize: 4992
SectorNumber: 3
Last Vister Time: Sun Jun  2 07:47:17 2019
Extend 1 Sectors
Extend Sector Id: 1, Sector: 12
Sector Id: 10
FileHeader contents. File size: 130. File type None File blocks:
sectorPos: 10
CreateTime: Sun Jun  2 07:47:17 2019
LastVisterTime: Sun Jun  2 07:47:17 2019
LastModifyTime: Sun Jun  2 07:47:17 2019
11 12
File contents:
SectorNumber: 11
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890\0\0\0\0\0\0\0\0
SectorNumber: 12
\0\0
Position: 0, FileLength: 128, NumBytes: 128
Writing 128 bytes at 0, from file of length 128.
MaxMaxFileSize: 4992
Last Vister Time: Sun Jun  2 07:47:17 2019
Last Modified Time: Sun Jun  2 07:47:17 2019
Exit openfile 0
Writing 10 bytes at 120, from file of length 120.
Reading 128 bytes at 0, from file of length 130.
LastSector: 0
MaxMaxFileSize: 4992
SectorNumber: 11
Last Vister Time: Sun Jun  2 07:47:17 2019
Reading 2 bytes at 128, from file of length 130.
LastSector: 1
MaxMaxFileSize: 4992
SectorNumber: 12
Last Vister Time: Sun Jun  2 07:47:17 2019
MaxMaxFileSize: 4992
MaxMaxFileSize: 4992
Last Vister Time: Sun Jun  2 07:47:17 2019
Last Modified Time: Sun Jun  2 07:47:17 2019

```

[illegible]

Exercise 6

synchDisk 是通过在原有的 Disk 基础上，增加 Lock 和 semaphore 来实现同步磁盘，保证数据一致性。之前我们提到 filesys 启动前，会先去启动这个 synchDisk。

```
void
SynchDisk::ReadSector(int sectorNumber, char* data)
{
    lock->Acquire();           // only one disk I/O at a time
    disk->ReadRequest(sectorNumber, data);
    semaphore->P();           // wait for interrupt
    lock->Release();
}
```

在他读取扇区的时候，是先去加一个锁，获得锁之后再去读取 Disk。同样的在 Console 基础上实现 synch 版本的 Console，思路也是相同的。

```
static Semaphore *readAvail = new Semaphore("Read avail", 0);
static Semaphore *writeDone = new Semaphore("Write Done", 0);
static void ReadAvail(int arg) { readAvail->V();}
static void WriteDone(int arg) { writeDone->V();}

void SynchConsole::PutChar(char ch){
    lock->Acquire();
    console->PutChar(ch);
    writeDone->P();
    lock->Release();
}
```

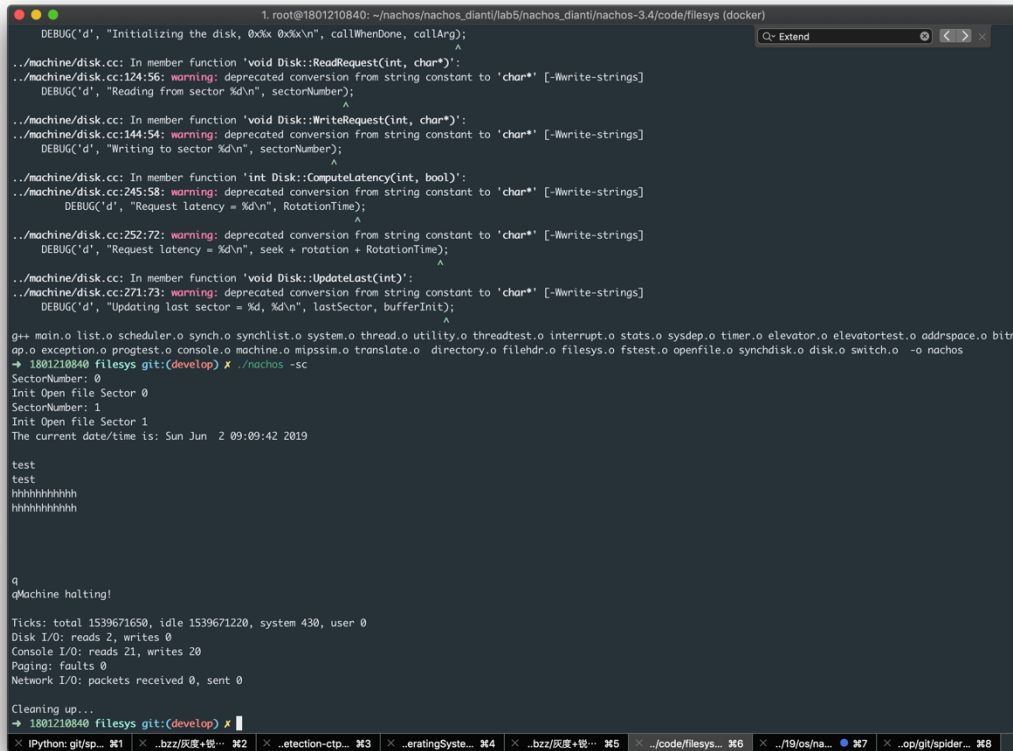
通过定义 readAvail, WriteDone 两个信号量 Write 和 read 来管理同步问题，再加上 Lock 来控制互斥问题。
在测试中，在 main 中定义-sc。

```
static SynchConsole *synchConsole;

void SynchConsoleTest (char *in, char *out)
{
    char ch;
    synchConsole = new SynchConsole(in, out);

    for (;;) {
        ch = synchConsole->GetChar();
        synchConsole->PutChar(ch);    // echo it!
        if (ch == 'q') return;    // if q, quit
    }
}
```

}



```
1.root@1801210840: ~/nachos/nachos_diant/lab5/nachos_diant/nachos-3.4/code/filesys (docker)
DEBUG('d', "Initializing the disk, 0x0x 0x0x\n", callWhenDone, callArg);
^
../machine/disk.cc: In member function 'void Disk::ReadRequest(int, char*)':
../machine/disk.cc:124:56: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
    DEBUG('d', "Reading from sector %d\n", sectorNumber);
    ^
../machine/disk.cc: In member function 'void Disk::WriteRequest(int, char*)':
../machine/disk.cc:144:54: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
    DEBUG('d', "Writing to sector %d\n", sectorNumber);
    ^
../machine/disk.cc: In member function 'int Disk::ComputeLatency(int, bool)':
../machine/disk.cc:245:58: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
    DEBUG('d', "Request latency = %d\n", RotationTime);
    ^
../machine/disk.cc:252:72: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
    DEBUG('d', "Request latency = %d\n", seek + rotation + RotationTime);
    ^
../machine/disk.cc: In member function 'void Disk::UpdateLast(int)':
../machine/disk.cc:221:73: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
    DEBUG('d', "Updating last sector = %d, %d\n", lastSector, bufferInit);
    ^
g++ main.o list.o scheduler.o synch.o synchlist.o system.o thread.o utility.o threadtest.o interrupt.o stats.o sysdep.o timer.o elevator.o elevator.o elevator.o addressspace.o bitm
ap.o exception.o progttest.o console.o machine.o mipssim.o translate.o directory.o filehdr.o filesys.o fstest.o openfile.o synchdisk.o disk.o switch.o -o nachos
→ 1801210840 filesys git:(develop) x ./nachos -sc
SectorNumber: 0
Init Open file Sector 0
SectorNumber: 1
Init Open file Sector 1
The current date/time is: Sun Jun 2 09:09:42 2019

test
test
hhhhhhhhhhhh
hhhhhhhhhhhh

q
qMachine halting!

Ticks: total 1539671650, idle 1539671220, system 430, user 0
Disk I/O: reads 2, writes 0
Console I/O: reads 21, writes 20
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
→ 1801210840 filesys git:(develop) x
```

Exercise 7

利用 `synchDisk` 来实现文件的同步异步访问。为保证能多人同步读取一个文件，借用读写锁的思路，设立 `ReadLock` 锁，和 `NumSector` 个信号量+记录 `read` 人数来完成读写锁模型的文件同步互斥访问。

```
// SynchDisk::PlusRead
// lab 5 Plus reader
//-----

void SynchDisk::PlusRead(int sector) {
    DEBUG('f', "Num Readers: %d\n", numReaders[sector]);
    readLock->Acquire();
    ++numReaders[sector];
    if (numReaders[sector] == 1){
        mutex[sector]->P();
    }
    printf("\033[91m Reader cnt: %d\n \033[0m",
numReaders[sector]);
```

```

        readLock->Acquire();
    }

//-----
// SynchDisk::MinusRead
//   lab 5 minus reader
//-----

void SynchDisk::MinusRead(int sector) {
    readLock->Acquire();
    --numReaders[sector];
    if (numReaders[sector] == 0){
        mutex[sector]->V();
    }
    printf("\033[91m Reader cnt: %d\n \033[0m",
numReaders[sector]);
    readLock->Acquire();
}

//-----
// SynchDisk::BeginWrite
//   lab 5 begin write
//-----

void SynchDisk::BeginWrite(int sector) {
    mutex[sector]->P();
}

//-----
// SynchDisk::EndWrite
//   lab 5 end write
//-----

void SynchDisk::EndWrite(int sector) {
    mutex[sector]->V();
}

```

```
1.root@1801210840: ~/nuchos/nuchos_dianti/lab5/nuchos_dianti/nuchos-3.4/code/filesys (docker)
Write Read main
Sequential write of 200 byte file, in 10 byte chunks
Directory contents:

Directory contents:
Id: 0, Name: TestFile, InUse: 1, Sector: 10, Type: 1, Path: TestFile
FileHeader contents. File size: 0. File type None File blocks:
sectorPos: 10
CreateTime: Sun Jun 2 12:07:38 2019
LastVisterTime: Sun Jun 2 12:07:38 2019
LastModifyTime: Sun Jun 2 12:07:38 2019

File contents:

Directory contents:

Directory contents:
Id: 0, Name: TestFile, InUse: 1, Sector: 10, Type: 1, Path: TestFile
FileHeader contents. File size: 0. File type None File blocks:
sectorPos: 10
CreateTime: Sun Jun 2 12:07:38 2019
LastVisterTime: Sun Jun 2 12:07:38 2019
LastModifyTime: Sun Jun 2 12:07:38 2019

File contents:

Directory contents:

Directory contents:
Id: 0, Name: TestFile, InUse: 1, Sector: 10, Type: 1, Path: TestFile
FileHeader contents. File size: 0. File type None File blocks:
sectorPos: 10
CreateTime: Sun Jun 2 12:07:38 2019
LastVisterTime: Sun Jun 2 12:07:38 2019
LastModifyTime: Sun Jun 2 12:07:38 2019

File contents:

Perf test: can't create TestFile
Begin Read main
Sequential read of 200 byte file, in 10 byte chunks
Directory contents:

Begin Read Reader1
Sequential read of 200 byte file, in 10 byte chunks
```

Challenge 1

性能优化, 考虑到 `freeMap` 分配的时候需要循环取调用 `freeMap->Find()` 接口, 当文件大小比较大的时候, 这个开销是比较大的。想通过 `BitMap` 连续分配地址来实现这个优化。在 `BitMap` 中定义 `Find2` 函数。

```
//-----
// BitMap::Find2
// lab 5 challenge 1
//-----

int BitMap::Find2(int cnt){
    for (int i = 0; i < numBits; ++i){
        int flag = 1;
        for (int j = 0; j < cnt; ++j){
            if (Test(i + j))
                flag = 0;
        }
        if (flag) {
```



```

        for (int j = 0; j < cnt; ++j)
            Mark(i + j);
        return i;
    }
}
return -1;
}

```

同时修改 filehdr 的 Allocate 函数，优先使用 Find2 来进行 BitMap 分配。

```

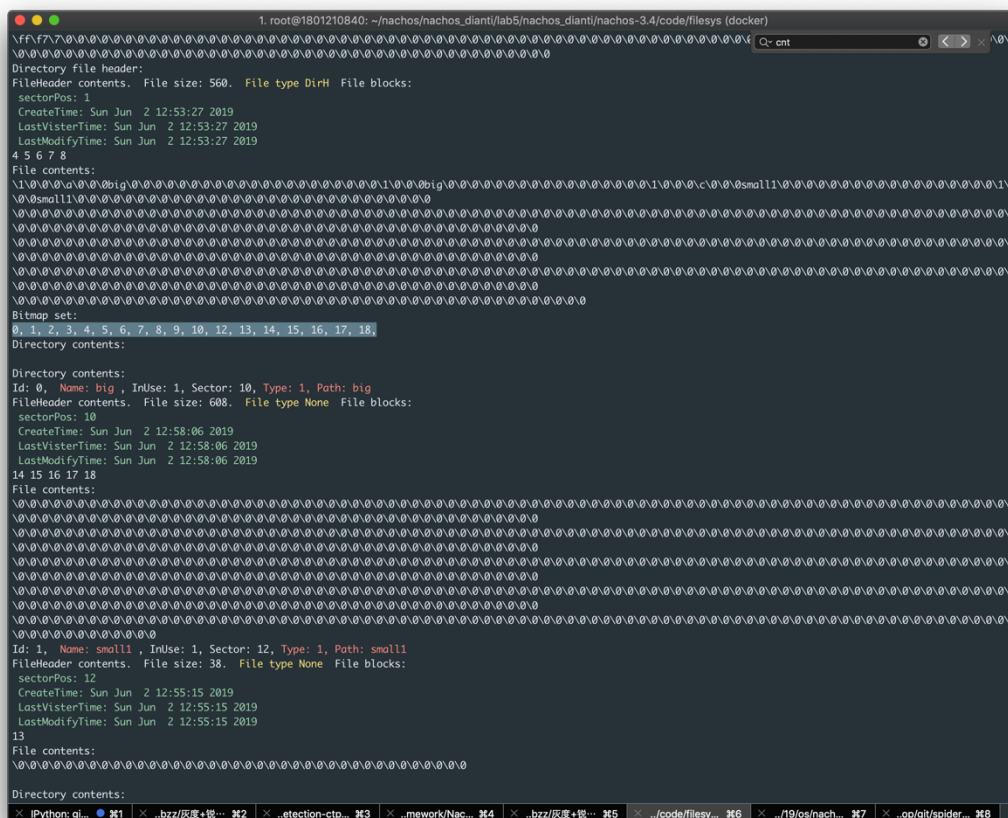
int freeSpace = freeMap->Find2(numSectors);
if (freeSpace != -1) {
    for (int i = 0; i < numSectors; ++i)
        dataSectors[i] = freeSpace + i;
} else {
    for (int i = 0; i < numSectors; i++)
        dataSectors[i] = freeMap->Find();
}

```

在实验过程中，先 ./nachos -cp test/small small && ./nachos -cp test/small small1

然后删除 small。./nachos -r small.这时候 bitMap 出现不连续空闲块。再 ./nachos -cp test/big big

在优化前，会分配不连续的 BitMap 给到 big 文件，而优化之后分配的 BitMap 地址则是连续的。提升了效率。



内容三：遇到的困难以及解决方法

在一开始调试的时候出现了很多问题，各种打断点之后还不能解决。主要是三个问题，一个是因为 IDE 对代码 `format` 进行自动优化，这个操作会对 `#include` 顺序进行排序，这就导致了在一些情况下会报错，实际上 Nachos 在很多地方都是有严格的 `#include` 顺序依赖的。第二个是使用 `time_t` 就会 `segment fault`。后来使用了回退代码一个个查，发现是在 Lab4 引进的，但是具体在哪并不清楚，也是 Lab5 是在 Lab3 代码的基础上进行操作的。

第三个问题，是一开始对代码理解不到位，实际上在进行文件系统操作之前，需要**-f**先初始化文件系统。

除此之外，还有很多小细节，在实验的过程中并没有很好地实现。在之后的实验中会更加注意的。

内容四：收获及感想

这次的作业的难度比前面四个 lab 有所增加，做起来还是比较吃力的，尤其是 **Exercise3** 的实现，实际上，为了偷懒在 **Exercise3** 的实现中，仅仅完成了一级间接索引，对文件大小的支持也仅仅到 **4KB**，并没有真正的扩展文件大小。这在之后的实践过程中，可以继续完善。

内容五：对课程的意见和建议

本次实验实际上在调试过程中花费了大量时间，如果有更为详尽的指导文档，可能会更好一些。

内容六：参考文献

[1] Stevens, W. R. (2002). UNIX 环境高级编程：英文版．机械工业出版社．