山东大学 软件 学院

**操作系统课程设计** 实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202000300125 | 姓名：贾星宇 | | 班级：计软20.5 |
| 实验编号：Lab5 | | | |
| 实验题目：具有二级索引的文件系统 | | | |
| 实验学时：4 | | 实验日期：2022年11月30日 | |
| 实验目的：  1. Nachos系统原有的文件系统只支持单级索引，最大能存取30 \* 128 = 3840字节大小的文件。本实验将在理解原文件系统的组织结构基础上扩展原有的文件系统，在Lab4的基础上，设计并实现具有二级索引的文件系统。  2. 为Nachos增加命令行选项-DI。执行./nachos -DI时显示Nachos磁盘的以下信息：Nachos磁盘的总体大小，已使用空间大小，空闲空间大小，普通文件数目，全部普通文件的总字节数，全部普通文件占用的空间大小(不包括文件头占用的，但加上普通文件数据扇区的内碎片)，总内碎片字节数(仅计普通文件数据扇区造成的)。  3. 若要求为Nachos文件增加rwx权限(可读，可写，可执行)，请给出在Nachos中实现的具体方法(不要求实现可运行的代码。在实验报告中用文字描述即可，必要时可在文字中结合关键代码片段、数据结构、对象等说明)。 | | | |
| 硬件环境：  HUAWEI matebook14 2020笔记本  Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz CPU  8GB内存  512GB SSD | | | |
| 软件环境：  宿主机：Windows 10 21H2 64位  虚拟机软件：VMware Workstation Pro 16.1.2 build-17966106  Linux：Ubuntu 14.04.6 LTS Desktop i386 (Trusty Tahr)  gcc/g++：(Ubuntu 4.8.4-2ubuntu1~14.04.4) 4.8.4  MIPS交叉编译器：gcc-2.8.1-mips.tar.gz  Nachos：Nachos-3.4-UALR-2022 | | | |
| 实验步骤与内容：  根据各个实验的内容，以及实现的过程，可写的包括但不限于：  解决问题的思路；  实现步骤；  关键源代码及注释(尽量以文本格式，且必须与提交的源代码一致)；  程序说明，特别是自己认为的精彩之处；  调试过程及记录；  运行结果(可文本格式，必要时抓屏)；  等等…   1. Nachos系统原有的文件系统只支持单级索引，最大能存取30 \* 128 = 3840字节大小的文件。本实验将在理解原文件系统的组织结构基础上扩展原有的文件系统，在Lab4的基础上，设计并实现具有二级索引的文件系统。   相似的，我们先将lab4中的filehdr、openfile、fstest等可能需要进一步修改的文件复制到lab5中进行隔离，随后从-cp命令开始，寻找文件系统在无法进行二级索引时进行的相关操作。首先在main方法中跳转到了Copy方法，Copy方法利用Create方法创建了长度为fileLength的文件to。随后我们定位到FileSystem的Create方法，在此方法中利用了FileHeader的Allocate方法分配存储空间。在Create后，Copy方法又利用Write方法向空间中写文件，而对于OpenFile的Write方法，我们定位到了WriteAt方法中，并发现在此方法中首先判断是否需要扩展文件长度，扩展之后再通过FileHeader类中的ByteToSector方法找到需要写回的扇区位置，并利用WriteSector方法写回。  根据上述分析可以发现，我们重点修改的方法存在于FileHdr类中，**为Allocate、Deallocate、ByteToSector、Print以及在实验四中创建的方法setNumBytes**。  首先我们对宏变量进行重定义：    NumDirect表示以及索引能够指向的扇区数，NumDirect2表示二级索引所能指向的扇区数，MaxFileSize为一个文件能够存储的最大字节数。  Allocate方法：  此方法的主要功能为给文件头分配指定大小的扇区，主要用在利用cp命令创建文件时。初始情况下没有二级索引，因此只需要利用for循环依次为每一个扇区寻找相应位置即可。但现在我们需要判断要分配的fileSize大小是否需要分配第二级索引。  如下图所示，我们首先计算fileSize大小的区域需要多少扇区，并将扇区数存储到对象中。当空余位置满足对扇区的分配后，我们判断分配的扇区数是否超过了一级索引所能分配的最大扇区，也就是NumDirect – 1。如果没有超过，则按照先前的方法循环Find并分配，如果超过了则首先为一级索引分配NumDirect-1个扇区索引，随后创建二级索引并将剩余扇区分配到二级索引数组中，并将此数组利用写方法写回到一级索引最后一个位置存储的扇区地址中。详细代码如下：    （图：Allocate方法）  Deallocate方法：  此方法的操作同上，负责清空所分配的扇区，首先判断清空时候是否需要清空二级索引所分配的扇区。需要注意的是，先前Allocate时，如果没有二级索引，那么我们为指向二级索引的一级索引位置赋值-1，因此我们可以据此判断。如果没有二级索引，那么我们就只清空一级索引；如果有的话我们需要按照一级索引最后一个位置的值找到二级索引，并依次清空。    （图：Deallocate方法）  ByteToSector方法：此方法主要用在WriteAt和ReadAt方法中，为写和读提供扇区位置，因此修改完此方法后则可以让openFile在创建好二级索引位置后能够写入二级索引所对应位置或者读出。方法如下：    （图：ByteToSector方法）  在此方法中，如果需要找的数组位置没有超过一级索引所代表的位置（NumDirect-1），则说明找的是一级索引；否则需要先根据一级索引指向的位置找到二级索引的位置，随后将二级索引读出来并返回相应位置的值。  Print方法：  此方法利用一级索引尾元素是否为-1判断此文件是否存在二级索引，如果存在则打印二级索引，具体方法类似于Deallocate方法：   |  | | --- | | void FileHeader::Print()  {  printf("begin print content: last is :%d\n", dataSectors[NumDirect - 1]);  int i, j, k,temp;  char \*data = new char[SectorSize];  if (dataSectors[NumDirect - 1] == -1)  {  time\_t thetime = numSectors + 16 \* 3600;  struct tm \*ptr = localtime(&thetime);  char str[80];  strftime(str, 100, "%c", ptr);  printf("FileHeader contents. File size: %d. File modification time:%s\n", numBytes, str);  temp = numSectors;  numSectors = divRoundUp(numBytes, SectorSize);  // printf("numBytes:%d\n",numBytes);  for (i = 0; i < numSectors; i++)  printf("%d ", dataSectors[i]);  printf("\nFile contents:\n");  for (i = k = 0; i < numSectors; i++)  {  synchDisk->ReadSector(dataSectors[i], data);  for (j = 0; (j < SectorSize) && (k < numBytes); j++, k++)  {  if ('\040' <= data[j] && data[j] <= '\176') // isprint(data[j])  printf("%c", data[j]);  else  printf("\\%x", (unsigned char)data[j]);  }  printf("\n");  }  }  else  {    int dataSectors2[NumDirect2];  synchDisk->ReadSector(dataSectors[NumDirect - 1], (char \*)dataSectors2);  time\_t thetime = numSectors + 16 \* 3600;  struct tm \*ptr = localtime(&thetime);  char str[80];  strftime(str, 100, "%c", ptr);  printf("FileHeader contents. File size: %d. File modification time:%s\n", numBytes, str);  temp = numSectors;  numSectors = divRoundUp(numBytes, SectorSize);  for (i = 0; i < NumDirect - 1; i++)  {  printf("%d ", dataSectors[i]);  }  for (; i < numSectors; i++)  {  printf("%d ", dataSectors2[i - (NumDirect - 1)]);  }  printf("\nFile contents:\n");  for (i = k = 0; i < NumDirect - 1; i++)  {  synchDisk->ReadSector(dataSectors[i], data);  for (j = 0; (j < SectorSize) && (k < numBytes); j++, k++)  {  if ('\040' <= data[j] && data[j] <= '\176') // isprint(data[j])  printf("%c", data[j]);  else  printf("\\%x", (unsigned char)data[j]);  }  }  printf("\nstart print second level:\n");  for (i = 0; i < numSectors - (NumDirect - 1); i++)  {  synchDisk->ReadSector(dataSectors2[i], data);  for (j = 0; (j < SectorSize) && (k < numBytes); j++, k++)  {  if ('\040' <= data[j] && data[j] <= '\176') // isprint(data[j])  printf("%c", data[j]);  else  printf("\\%x", (unsigned char)data[j]);  }  }  printf("\nend second level\n");  }  delete[] data;  numSectors = temp;  } |   setNumBytes方法  此方法为Append方法中调用的核心方法，具体功能为当需要向文件中新增内容并且新增内容后文件长度超过了原有文件长度时调用的方法。在实验四中，此方法的功能为找到合适的扇区并分配到一级索引，然后写回分配好的扇区索引和相应的bitMap。当我们扩展了二级索引后，在分配索引时就需要分如下情况讨论：   1. 当前要新增的扇区是否要分配到二级索引上 2. 如果分配到二级索引上，在分配新扇区前这个二级索引是否已经存在   对于问题A，我们可以获取当前要分配的总扇区数是否超过了一级索引所能指向的所有扇区；对于问题B，存在于我们要分配二级索引扇区时，要判断二级索引是否存在来判断是否需要新建一个二级索引还是读出原来的，这个通过查看一级索引最后的位置是否为-1决定。详细代码如下：   |  | | --- | | bool FileHeader::setNumBytes(int numBytes)  {  int temp = numSectors;  numSectors = divRoundUp(this->numBytes, SectorSize);  int NewNumSectors = divRoundUp(numBytes, SectorSize);  if (NewNumSectors == numSectors) //sector do not change  {  this->numBytes = numBytes;  return true;  }  int difSector = NewNumSectors - numSectors;  OpenFile \*bitmapfile = new OpenFile(0);  BitMap \*freeMap = new BitMap(NumSectors);  freeMap->FetchFrom(bitmapfile);  if (NewNumSectors > (NumDirect - 1 + NumDirect2) || freeMap->NumClear() < difSector)  {  return false;  }  if (NewNumSectors < NumDirect)  {  // printf("begin append!\n");  for (int i = numSectors; i < NewNumSectors; i++)  {  dataSectors[i] = freeMap->Find();  }  }  else  { //the new file need two level index  // printf("begin extend and append!\n");  **if (numSectors < NumDirect)** //formal file do not have two level index  {  for (int i = numSectors; i < NumDirect; i++)  {  dataSectors[i] = freeMap->Find();  }  int dataSectors2[NumDirect2];  for (int i = 0; i < NewNumSectors - (NumDirect - 1); i++)  {  dataSectors2[i] = freeMap->Find();  }  synchDisk->WriteSector(dataSectors[NumDirect - 1], (char \*)dataSectors2);  }  else  { //formal file have two level index  int dataSectors2[NumDirect2];  synchDisk->ReadSector(dataSectors[NumDirect - 1], (char \*)dataSectors2);  for (int i = numSectors - (NumDirect - 1); i < NewNumSectors - (NumDirect - 1); i++)  {  dataSectors2[i] = freeMap->Find();  }  synchDisk->WriteSector(dataSectors[NumDirect - 1], (char \*)dataSectors2);  }  }  this->numBytes = numBytes;  freeMap->WriteBack(bitmapfile);  // printf("end extend and append!\n");  this->numSectors = temp;  return true;  } |   至此，便实现了二级索引的扩展。效果在最后展示。   1. 为Nachos增加命令行选项-DI。执行./nachos -DI时显示Nachos磁盘的以下信息：Nachos磁盘的总体大小，已使用空间大小，空闲空间大小，普通文件数目，全部普通文件的总字节数，全部普通文件占用的空间大小(不包括文件头占用的，但加上普通文件数据扇区的内碎片)，总内碎片字节数(仅计普通文件数据扇区造成的)。   首先我们回到main函数，在此处添加-DI指令和相对应要调用的方法：    随后我们前往fileSystem类中新建PrintDiskInfo方法。  浏览实验要求，我们分别需要打印的信息为：  **Nachos磁盘的总体大小**  **已使用空间大小**  **空闲空间大小**  **普通文件数目**  **全部普通文件的总字节数**  **全部普通文件占用的空间大小(不包括文件头占用的，但加上普通文件数据扇区的内碎片)**  **总内碎片字节数(仅计普通文件数据扇区造成的)**  磁盘总体大小  对于磁盘的大小我们可以前往Disk类中寻找，可以发现定义如下：    其中NumSectors为32轨道\*32扇区，SectorSize为每扇区的128字节，因此磁盘大小为32\*32扇区，32\*32\*128字节。  已使用空间大小  已使用的磁盘位置可以前往Bitmap类中查看，bitmap文件放置在了freeMapFile类中，以十六进制的形式存储，每一个字节代表了一个扇区是否被占用。通过freeMap的NumClear方法可以得到未使用扇区数，通过NumSectors可以得到总扇区数；做差即为已使用扇区数。  空闲空间大小  先前已经得到。  普通文件数目：  普通文件指的是除了头文件和目录文件之外的文件。查看Directory类，发现普通文件以DirectoryEntry的形式存储在目录中，如果inUse==1则说明被使用。因此可以使用这个方法新增统计使用文件的数目：    全部文件总字节数：  同样的，我们在判断table[i]处有文件后，利用FetchFrom方法得到该文件的文件头，并利用文件头中FileLength得到此文件的总字节数：    全部文件占用空间大小（磁盘数）  这个属性可以利用每个文件的文件大小除以SectorSize向上取整得到，计算方法:    总内碎片数：通过利用总的扇区数\*SectorSize减去文件总字节数得到。  **最终构造出打印函数：**    演示如下：   |  | | --- | | u1@ubuntu:~/oscp/nachos-3.4-ualr-2022/code/lab5$ ./nachos -f  No threads ready or runnable, and no pending interrupts.  Assuming the program completed.  Machine halting!  Ticks: total 82520, idle 82270, system 250, user 0  Disk I/O: reads 3, writes 5  Console I/O: reads 0, writes 0  Paging: faults 0  Network I/O: packets received 0, sent 0  Cleaning up... |  |  | | --- | | u1@ubuntu:~/oscp/nachos-3.4-ualr-2022/code/lab5$ ./nachos -DI  Disk size: 1024 sectors, 131072 bytes.  Used: 5 sectors, 640 bytes.  Free: 1019 sectors, 130432 bytes.  0 bytes in 0 files, occupy 0 bytes(0 sectors).  0 bytes of internal fragmentation in 0 sectors.  No threads ready or runnable, and no pending interrupts.  Assuming the program completed.  Machine halting!  Ticks: total 2660, idle 2500, system 160, user 0  Disk I/O: reads 5, writes 0  Console I/O: reads 0, writes 0  Paging: faults 0  Network I/O: packets received 0, sent 0  Cleaning up... |  |  | | --- | | u1@ubuntu:~/oscp/nachos-3.4-ualr-2022/code/lab5$ ./nachos -cp test/huge huge  No threads ready or runnable, and no pending interrupts.  Assuming the program completed.  Machine halting!  Ticks: total 4227020, idle 4211050, system 15970, user 0  Disk I/O: reads 268, writes 264  Console I/O: reads 0, writes 0  Paging: faults 0  Network I/O: packets received 0, sent 0  Cleaning up... |  |  | | --- | | u1@ubuntu:~/oscp/nachos-3.4-ualr-2022/code/lab5$ ./nachos -D  Bit map file header:  begin print content: last is :-1  FileHeader contents. File size: 128. File modification time:Thu Jan 1 08:00:01 1970  2  File contents:  \ff\ff\ff\1\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0  Directory file header:  begin print content: last is :-1  FileHeader contents. File size: 200. File modification time:Thu Jan 1 08:00:02 1970  3 4  File contents:  \1\0\0\0\5\0\0\0huge\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0  \0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0  Bitmap set:  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,  Directory contents:  Name: huge, Sector: 5  begin print content: last is :-1  FileHeader contents. File size: 2432. File modification time:Fri Aug 13 07:35:04 2021  6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24  File contents:  <huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge hug  e huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge hu  ge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<hug  e huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge hu  ge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge  huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge hu  ge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge h  uge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge  >\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge h  uge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge  huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<h  uge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge  huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge  huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge  huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge  huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge hu  ge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge  huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge hug  e huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a  No threads ready or runnable, and no pending interrupts.  Assuming the program completed.  Machine halting!  Ticks: total 15910, idle 15000, system 910, user 0  Disk I/O: reads 30, writes 0  Console I/O: reads 0, writes 0  Paging: faults 0  Network I/O: packets received 0, sent 0  Cleaning up... |  |  | | --- | | u1@ubuntu:~/oscp/nachos-3.4-ualr-2022/code/lab5$ ./nachos -ap test/huge huge  time:1669806942inodes have been written back  No threads ready or runnable, and no pending interrupts.  Assuming the program completed.  Machine halting!  Ticks: total 4995020, idle 4967950, system 27070, user 0  Disk I/O: reads 612, writes 290  Console I/O: reads 0, writes 0  Paging: faults 0  Network I/O: packets received 0, sent 0  Cleaning up... |  |  | | --- | | u1@ubuntu:~/oscp/nachos-3.4-ualr-2022/code/lab5$ ./nachos -D  Bit map file header:  begin print content: last is :-1  FileHeader contents. File size: 128. File modification time:Thu Jan 1 08:00:01 1970  2  File contents:  \ff\ff\ff\ff\ff\1f\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0  Directory file header:  begin print content: last is :-1  FileHeader contents. File size: 200. File modification time:Thu Jan 1 08:00:02 1970  3 4  File contents:  \1\0\0\0\5\0\0\0huge\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0  \0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0  Bitmap set:  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44,  Directory contents:  Name: huge, Sector: 5  begin print content: last is :33  FileHeader contents. File size: 4864. File modification time:Wed Nov 30 19:15:42 2022  6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 34 35 36 37 38 39 40 41 42 43 44  File contents:  <huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge  start print second level:  >\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a<huge huge huge huge huge huge huge>\a  end second level  No threads ready or runnable, and no pending interrupts.  Assuming the program completed.  Machine halting!  Ticks: total 54820, idle 53310, system 1510, user 0  Disk I/O: reads 50, writes 0  Console I/O: reads 0, writes 0  Paging: faults 0  Network I/O: packets received 0, sent 0  Cleaning up... |  |  | | --- | | u1@ubuntu:~/oscp/nachos-3.4-ualr-2022/code/lab5$ ./nachos -ap test/small huge  time:1669806993inodes have been written back  No threads ready or runnable, and no pending interrupts.  Assuming the program completed.  Machine halting!  Ticks: total 131020, idle 130200, system 820, user 0  Disk I/O: reads 20, writes 7  Console I/O: reads 0, writes 0  Paging: faults 0  Network I/O: packets received 0, sent 0  Cleaning up... |  |  | | --- | | u1@ubuntu:~/oscp/nachos-3.4-ualr-2022/code/lab5$ ./nachos -cp test/small small  No threads ready or runnable, and no pending interrupts.  Assuming the program completed.  Machine halting!  Ticks: total 151520, idle 150850, system 670, user 0  Disk I/O: reads 13, writes 9  Console I/O: reads 0, writes 0  Paging: faults 0  Network I/O: packets received 0, sent 0  Cleaning up... |  |  | | --- | | u1@ubuntu:~/oscp/nachos-3.4-ualr-2022/code/lab5$ ./nachos -cp test/medium mediumNo threads ready or runnable, and no pending interrupts.  Assuming the program completed.  Machine halting!  Ticks: total 360520, idle 359070, system 1450, user 0  Disk I/O: reads 26, writes 22  Console I/O: reads 0, writes 0  Paging: faults 0  Network I/O: packets received 0, sent 0  Cleaning up... |  |  | | --- | | u1@ubuntu:~/oscp/nachos-3.4-ualr-2022/code/lab5$ ./nachos -DI  Disk size: 1024 sectors, 131072 bytes.  Used: 51 sectors, 6528 bytes.  Free: 973 sectors, 124544 bytes.  5092 bytes in 3 files, occupy 5376 bytes(42 sectors).  284 bytes of internal fragmentation in 3 sectors.  No threads ready or runnable, and no pending interrupts.  Assuming the program completed.  Machine halting!  Ticks: total 40520, idle 40090, system 430, user 0  Disk I/O: reads 14, writes 0  Console I/O: reads 0, writes 0  Paging: faults 0  Network I/O: packets received 0, sent 0  Cleaning up... |  |  | | --- | | u1@ubuntu:~/oscp/nachos-3.4-ualr-2022/code/lab5$ ./nachos -r huge  No threads ready or runnable, and no pending interrupts.  Assuming the program completed.  Machine halting!  Ticks: total 66520, idle 66180, system 340, user 0  Disk I/O: reads 8, writes 3  Console I/O: reads 0, writes 0  Paging: faults 0  Network I/O: packets received 0, sent 0  Cleaning up...  u1@ubuntu:~/oscp/nachos-3.4-ualr-2022/code/lab5$ ./nachos -r small  No threads ready or runnable, and no pending interrupts.  Assuming the program completed.  Machine halting!  Ticks: total 66520, idle 66210, system 310, user 0  Disk I/O: reads 7, writes 3  Console I/O: reads 0, writes 0  Paging: faults 0  Network I/O: packets received 0, sent 0  Cleaning up...  u1@ubuntu:~/oscp/nachos-3.4-ualr-2022/code/lab5$ ./nachos -r medium  No threads ready or runnable, and no pending interrupts.  Assuming the program completed.  Machine halting!  Ticks: total 66520, idle 66210, system 310, user 0  Disk I/O: reads 7, writes 3  Console I/O: reads 0, writes 0  Paging: faults 0  Network I/O: packets received 0, sent 0  Cleaning up... |  |  | | --- | | u1@ubuntu:~/oscp/nachos-3.4-ualr-2022/code/lab5$ ./nachos -DI  Disk size: 1024 sectors, 131072 bytes.  Used: 5 sectors, 640 bytes.  Free: 1019 sectors, 130432 bytes.  0 bytes in 0 files, occupy 0 bytes(0 sectors).  0 bytes of internal fragmentation in 0 sectors.  No threads ready or runnable, and no pending interrupts.  Assuming the program completed.  Machine halting!  Ticks: total 2660, idle 2500, system 160, user 0  Disk I/O: reads 5, writes 0  Console I/O: reads 0, writes 0  Paging: faults 0  Network I/O: packets received 0, sent 0  Cleaning up... |  1. 若要求为Nachos文件增加rwx权限(可读，可写，可执行)，请给出在Nachos中**实现的具体方法**(不要求实现可运行的代码。在实验报告中用文字描述即可，必要时可在文字中结合关键代码片段、数据结构、对象等说明)。   首先我们前往main函数查看总共可以对文件系统做哪些操作，并将这些操作分为rwx三类：  r:-D,-DI,-l,-p  w:-cp,-ap,-hap,-nap  x:可执行，拥有此权限后方可以执行此文件，比如运行，此时需要获取目录相关内容。在nachos中咱不存在此类指令  如果需要为文件系统增加上述权限，则需要分别对文件头和目录文件作出修改。首先对文件的文件头增加rw权限，如果拥有r权限，则对于文件头的print方法访问有效；如果拥有w权限，则对于文件头的WriteBack方法有效。对于目录来说，当拥有了x权限，方可以对目录执行操作，如Print，Add等操作。以上限制操作通过if语句执行。  首先，由于nachos系统针对的是单用户，所以我们不需要考虑共享文件系统的问题。随后，为了标记每个文件的三个权限之一，我们需要占用文件头3位的空间来表示此用户对于此文件拥有哪些权限。比如，利用int类属性priority表示权限，其中取三位分别代表可读、可写和可执行。  比如，priority=7（111）表示可读可写可执行，priority=6（110）表示可读，可写不可执行。  对于文件的操作，基本上定义在了fstest内部，因此我们需要针对每一个方法添加限制：  Copy方法，由于是创建新文件，所以需要有写权限，在创建文件后首先一定需要有写权限才能创建，所以默认拥有即可。  Append方法是添加文件内容，因此需要具有读和写权限，随后当我们打开to文件后，需要首先读取文件头中的权限判断是否有读和写权限，如果有那么继续执行后续方法，如果没有则报错，这个过程可以通过在打开to文件后添加ASSERT（priority==6）实现。  同样的，NAppend和Happened也是先读取目标文件并判断权限是否为6（110）随后执行。  对于-D所调用的print方法，首先打印基本信息，随后打印每个普通文件的信息。无论如何，都是打印文件，因此在打印前需要判断是否具有可读权限（100，即4）。对于Directory的Print方法，第一行添加ASSERT（priority==4），如果满足后再执行后续方法。同理，在BitMap类和FileHeader类的Print方法前也需要加入ASSERT(priority==4)来判断。 | | | |
| 结论分析与体会：  通过此次实验，我对文件系统的二级索引以及文件系统的整体有了更加深入地认识，对于文件如何执行相关操作以及操作流程也有了更紧密的联系认识。  通过此次实验，我寻找错误、改正错误的能力也有了较大的提高  相信在这几个实验的铺垫下，后续的实验我将更加顺利地完成！ | | | |