|  |  |
| --- | --- |
| 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：2022.12.7 |
| 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)。  注1：若Lab5全部完成，演示提交的代码为带有-DI命令行选项的。 | |
| 硬件环境：  DELL品牌DESKTOP-3CCOK51型号笔记本  Intel Core i5-10210U CPU  8GB内存  1T 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：  在Lab4的基础上，设计并实现具有二级索引的文件系统:    如上图所示，构建具有二级索引的文件头，原先的前NumDirect-1项还是直接索引，最后一项(数组下标为NumDirect-1)指向一个二级索引块，这个块存放新的索引条目，共NumDirect+2项(数组下标为0~NumDirect+1)。扩大后的文件最大长度为（29 + 32）\* 128 = 7808字节。  二级索引块是动态产生的。当文件大小不需要它时，一级索引块的最后一项可设置为-1，此时不存在二级索引块。当文件大小增长到一级索引无法支持时，再分配一个新的块存二级索引，并将二级索引块的扇区号存入一级索引块的第31项。  具体实现是在filehdr.cc中所有相关方法里新增关于二级索引的逻辑，使二级索引的逻辑对于nachos的其他部分是透明的，即可以像原来一样操作文件，只不过文件大小的上限增加了。  我们知道，nachos模拟的磁盘中默认一个扇区共128个字节，即sectorSize = 128, sizeof(int) = 4。用于存储文件头时，去掉总字节数和修改时间后，存储文件内容的扇区NumDirect = 30，即有存储文件消耗30个sector，30\*128 = 3840，即文件可占字节为3840。  想要继续扩展，将该数组的最后一个元素定义为下一级dataSector的索引，即dataSector2所在扇区的扇区号，即可从磁盘中读取下一级索引的内容。下一级索引将全部用于存放文件内容所在扇区的编号，所以大小可以是128/4=32。  我们直接利用nachos的同步磁盘完成二级索引的读写。  读取时，调用synchDisk->ReadSector()将二级索引所在扇区的内容装填到临时数组dataSector2[]中；  写入时，调用synchDisk->WriteSector()将临时数组ddataSector2[]中内容写回磁盘对应的扇区。  利用这个思路，我们就可以修改文件头中的各种方法了，首先是扩展Extend：   1. **bool** 2. FileHeader::Extend(**int** newNumBytes){ 3. **if**(newNumBytes<numBytes)**return** **false**;//wrong param 4. **if**(newNumBytes==numBytes)**return** **true**;//no need to change 5. **int** newNumSectors=ceil((**double**)newNumBytes/(**double**)SectorSize); 6. **if**(newNumSectors==numSectors()){//same num of sectors 7. numBytes=newNumBytes; 8. **return** **true**; 9. } 10. **int** deltaSectors=newNumSectors-numSectors(); 11. OpenFile \*openFile=**new** OpenFile(0); 12. BitMap \*bitMap=**new** BitMap(NumSectors); 13. bitMap->FetchFrom(openFile); 14. //disk is full or file is too big 15. **if**(newNumSectors>=NumDirect+NumDirect2||deltaSectors>bitMap->NumClear()){ 16. printf("disk is full/ file is too big\n"); 17. printf("old size:%dB--new size:%dB\n",numBytes,newNumBytes); 18. printf("new sectors:%d   delta:%d   direct:%d+%d   clear:%d\n",newNumSectors,deltaSectors,NumDirect,NumDirect2,bitMap->NumClear()); 19. bitMap->Print(); 20. **return** **false**; 21. } 22. //allocate 23. **for**(**int** i=numSectors();i<newNumSectors&&i<LastIndex;i++)dataSectors[i]=bitMap->Find(); 24. **if**(newNumSectors>=NumDirect){//修改后的文件大小需要扇区数量多于一级索引表的大小：需要扩展二级索引 25. **int** dataSectors2[NumDirect2],start=0; 26. **if**(dataSectors[LastIndex]!=-1){//已经扩展了二级索引 27. //从硬盘将A位置的内容读入B地址（存的是扇区号） 28. synchDisk->ReadSector(dataSectors[LastIndex],(**char**\*)dataSectors2); 29. start=numSectors()-NumDirect+1; 30. }**else** dataSectors[LastIndex]=bitMap->Find(); //未扩展二级索引 31. //allocate for level 2 32. **for**(**int** i=start;i<=newNumSectors-NumDirect;i++)dataSectors2[i]=bitMap->Find(); 33. synchDisk->WriteSector(dataSectors[LastIndex],(**char**\*)dataSectors2); 34. } 35. bitMap->WriteBack(openFile); 36. numBytes=newNumBytes; 37. **return** **true**; 38. }   在分配空间的方法Allocate中也新增关于二级索引的逻辑，以便于文件创建时就可以拥有这么大的空间：   1. **bool** 2. FileHeader::Allocate(BitMap \*freeMap, **int** fileSize) 3. { 4. numBytes = fileSize; 5. **if** (freeMap->NumClear() < numSectors())**return** FALSE;//not enough disk space 6. **if**(NumDirect+NumDirect2<=numSectors())**return** **false**;//not enough file indices 8. **for** (**int** i = 0; i < numSectors()&&i<LastIndex; i++) 9. dataSectors[i] = freeMap->Find(); 10. **if**(numSectors()<LastIndex)dataSectors[LastIndex]=-1;//no need level 2 11. **else**{//需要二级索引 12. dataSectors[LastIndex]=freeMap->Find(); 13. **int** dataSectors2[NumDirect2]; 14. **for**(**int** i=0;i<=numSectors()-NumDirect;i++) 15. dataSectors2[i]=freeMap->Find(); 16. synchDisk->WriteSector(dataSectors[LastIndex],(**char**\*)dataSectors2); 17. } 18. **return** TRUE; 19. }   在释放空间的方法Deallocate中新增关于二级索引的逻辑：若存在二级索引则将二级索引块读取到内存中，再将二级索引所在块号传入Clear方法，将位图中的该块标记为空闲状态，随后对二级索引中的各个占用扇区调用Clear方法进行释放。   1. **void** 2. FileHeader::Deallocate(BitMap \*freeMap) 3. { 4. **for** (**int** i = 0; i < numSectors()&&i<LastIndex; i++) { 5. ASSERT(freeMap->Test((**int**) dataSectors[i]));  // ought to be marked! 6. freeMap->Clear((**int**) dataSectors[i]); 7. } 8. **if**(dataSectors[LastIndex]!=-1){//需要释放二级索引的空间 9. **int** dataSectors2[NumDirect2]; 10. synchDisk->ReadSector(dataSectors[LastIndex],(**char**\*)dataSectors2); 11. freeMap->Clear((**int**)dataSectors[LastIndex]); 12. **for**(**int** i=0;i<=numSectors()-NumDirect;i++) 13. freeMap->Clear((**int**)dataSectors2[i]); 14. } 15. }   方法ByteToSector的作用是根据文件的字节序号，返回该字节所在扇区。同样需要新增关于二级索引的逻辑：判断该字节位置是否处于二级索引中，若不在则字节序号和扇区大小做除法运算取上整即可得到对应扇区的数组下标；若在二级索引中则取出（offset/SectorSize-LastIndex）位置的扇区号并返回。   1. **int** 2. FileHeader::ByteToSector(**int** offset) 3. { 4. **if**(offset/SectorSize<LastIndex)//不在二级索引中 5. **return**(dataSectors[offset / SectorSize]); 6. **else**{//在二级索引中 7. **int** dataSectors2[NumDirect2]; 8. synchDisk->ReadSector(dataSectors[LastIndex],(**char**\*)dataSectors2); 9. **return** dataSectors2[offset/SectorSize-LastIndex]; 10. } 11. }   此方法是OpenFile等类操作文件的重要部分，nachos将所有关于文件某字节的操作的入口设到了这里，只要这个方法正确，那么文件具体存储在哪个扇区对于openfile来说就是透明的。所以修改这一方法后，OpenFile对文件的读写等操作就可以顺利进行了，不需要关心二级索引对其的影响。  打印单个文件信息的Print方法中新增关于二级索引的逻辑：   1. **void** 2. FileHeader::Print() 3. { 4. **int** i, j, k; 5. **char** \*data = **new** **char**[SectorSize]; 7. **bool** level2=dataSectors[LastIndex]!=-1; 8. **int** dataSectors2[NumDirect2]; 9. **if**(level2) 10. synchDisk->ReadSector(dataSectors[LastIndex],(**char**\*)dataSectors2);  13. printf("FileHeader contents.\nFile size: %d.\nFile blocks:", numBytes); 14. **for** (i = 0; i < numSectors()&&i<LastIndex; i++)printf("%d ", dataSectors[i]); 15. **if**(level2){ 16. printf("  (level2)"); 17. **for**(i=0;i<=numSectors()-NumDirect;i++)printf("%d ",dataSectors2[i]); 18. } 20. **if**(modifiedTime){//only normal file can have modified time 21. **char** s[100]; 22. **time\_t** tmpTime=(**time\_t**)modifiedTime; 23. strftime(s, **sizeof**(s), "%Y-%m-%d %H:%M:%S", &\*localtime(&tmpTime)); 24. printf("\nLast modified time:%s", s); 25. } 27. printf("\nFile contents:\n"); 28. **for** (i = k = 0; i < numSectors(); i++) { 29. **if**(i<LastIndex)synchDisk->ReadSector(dataSectors[i], data); 30. **else** synchDisk->ReadSector(dataSectors2[i-LastIndex],data); 31. **for** (j = 0; (j < SectorSize) && (k < numBytes); j++, k++) { 32. **if** ('\040' <= data[j] && data[j] <= '\176')   // isprint(data[j]) 33. printf("%c", data[j]); 34. **else** 35. printf("\\%x", (unsigned **char**)data[j]); 36. } 37. printf("\n"); 38. } 39. **delete** [] data; 40. }   实验要求2中：为Nachos增加命令行选项-DI。因此在main.cc的FILESYS部分增加-DI选项的捕获，若输入为“-DI”则由fileSystem调用PrintInfo()方法进行磁盘信息的打印：   1. #ifdef FILESYS 2. **…** 3. } **else** **if** (!strcmp(\*argv, "-DI")){  // 新增DI指令 打印磁盘信息  print disk info 4. fileSystem->PrintInfo(); 5. }   在filesys.cc中新增用于打印磁盘信息的方法PrintInfo：首先获取磁盘的位图和目录等结构，再逐步计算并打印磁盘状态信息。   1. **void** 2. FileSystem::PrintInfo(){ 3. BitMap \*freeMap=**new** BitMap(NumSectors); 4. freeMap->FetchFrom(freeMapFile); 5. Directory \*directory = **new** Directory(NumDirEntries); 6. //NumDirEntries：目录的最大文件数 7. directory->FetchFrom(directoryFile); 9. //总体大小 10. **int** totalSize=NumSectors\*SectorSize; 11. printf("Total size: %d Sectors, %d Bytes\n",NumSectors,totalSize); 13. //已使用空间大小、空闲空间大小 14. **int** clearSectors=freeMap->NumClear(); 15. printf("Used size: %d Sectors, %d Bytes\n",NumSectors-clearSectors,totalSize-clearSectors\*SectorSize); 16. printf("Free size: %d Sectors, %d Bytes\n",clearSectors,clearSectors\*SectorSize); 18. //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、总内碎片字节数 19. **int** idealBytes=directory->BytesUsed(**false**); 20. **int** normalSectors=directory->SectorStat(**false**); 21. **int** allBytes=normalSectors\*SectorSize; 22. **int** fragmentedSectors=directory->SectorStat(**true**); 23. printf("Size used by %d normal files:\n\twithout internal fragments: %d Bytes\n",directory->NumUsing(),idealBytes); 24. printf("\tactually used: %d Bytes in %d Sectors\n",allBytes,normalSectors); 25. printf("\tfragmented: %d Bytes in %d Sectors\n",allBytes-idealBytes,fragmentedSectors); 26. }   其中对于一些较为深入的统计（例如碎片大小等），我们对其进行了封装。  统计目录中字节数的方法Directory::ByteUsed(bool includingFrag)：   1. int 2. Directory::BytesUsed(bool includingFrag){ 3. FileHeader \*hdr=new FileHeader; 4. int res=0; 5. for(int i=0;i<tableSize;i++) 6. if(table[i].inUse){ 7. hdr->FetchFrom(table[i].sector); 8. res+=hdr->NumBytes(includingFrag); 9. } 10. return res; 11. }   此方法可以累加各个文件的字节数，是否包含碎片也作为参数传入FileHeader::NumBytes()：   1. int 2. FileHeader::NumBytes(bool includingFrag){ 3. return includingFrag?numSectors()\*SectorSize:numBytes; 4. }   这其中的逻辑就较为简单了。  同样，对于扇区信息的统计，有Directory::SectorStat()：   1. int 2. Directory::SectorStat(bool onlyFragmented){ 3. FileHeader \*hdr=new FileHeader; 4. int res=0; 5. for(int i=0;i<tableSize;i++) 6. if(table[i].inUse){ 7. hdr->FetchFrom(table[i].sector); 8. res+=onlyFragmented?hdr->NumBytes(true)!=hdr->NumBytes(false):hdr->numSectors(); 9. } 10. return res; 11. }   也是分文件统计扇区数量。其中核心语句：  res+=onlyFragmented?hdr->NumBytes(true)!=hdr->NumBytes(false):hdr->numSectors();  分开来看，若只统计存在碎片的扇区，则每个文件对结果的贡献值最多为1，此处采用hdr->NumBytes(true)!=hdr->NumBytes(false)的布尔值表示，即若改文件包含与不包含碎片的字节数相同，则该文件没有包含碎片的扇区。若统计的是所有扇区，直接调用FileHeader::numSectors()即可。  对于实验要求的第三点：为Nachos文件增加rwx权限(可读，可写，可执行)  这一部分设想的实现四库是在每个文件的头部新增三个默认值为true的布尔类型变量r、w、x来对应读、写、执行三种权限，并在filehdr.cc中新增针对这三个属性值的get和set方法，并在openfile.cc中的ReadAt和writeAt方法中，进行具体续写操作前分别对r和w进行是否为true的检验，若为false则打印无权限的相关控制信息并返回。若后续增添了执行文件的方法则也需要相应地对x权限进行检验。  可以通过保持读写文件入口的唯一性，由此保证文件无法绕过权限检验就进行操作，同时这种方式也具有较好的封装性。  Readme中指令运行结果：  rm -f DISK  ./nachos -f  ./nachos -DI  ./nachos -cp test/huge huge  ./nachos -D  ls --full-time test/huge  ./nachos -ap test/huge huge  ./nachos -D  ./nachos -ap test/small huge  ./nachos -cp test/small small  ./nachos -cp test/medium medium  ./nachos -DI  ./nachos -r huge  ./nachos -r small  ./nachos -r medium  ./nachos -DI  u1@ubuntu:~/oscp2/nachos-3.4-ualr-2022/code/lab5$ rm -f DISK  u1@ubuntu:~/oscp2/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:~/oscp2/nachos-3.4-ualr-2022/code/lab5$ ./nachos -DI  Total size: 1024 Sectors, 131072 Bytes  Used size: 5 Sectors, 640 Bytes  Free size: 1019 Sectors, 130432 Bytes  Size used by 0 normal files:  without internal fragments: 0 Bytes  actually used: 0 Bytes in 0 Sectors  fragmented: 0 Bytes 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:~/oscp2/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:~/oscp2/nachos-3.4-ualr-2022/code/lab5$ ./nachos -D  Bit map file header:  FileHeader contents.  File size: 128.  File blocks: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:  FileHeader contents.  File size: 200.  File blocks: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  FileHeader contents.  File size: 2432.  File blocks:6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24  Last modified time:2021-08-12 15:35:04  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:~/oscp2/nachos-3.4-ualr-2022/code/lab5$ ls --full-time test/huge  -rwxrw-rw- 1 u1 u1 2432 2021-08-12 15:35:04.000000000 -0700 test/huge  u1@ubuntu:~/oscp2/nachos-3.4-ualr-2022/code/lab5$ ./nachos -ap test/huge huge  No threads ready or runnable, and no pending interrupts.  Assuming the program completed.  Machine halting!  Ticks: total 4963020, idle 4937660, system 25360, user 0  Disk I/O: reads 556, writes 289  Console I/O: reads 0, writes 0  Paging: faults 0  Network I/O: packets received 0, sent 0  Cleaning up...  u1@ubuntu:~/oscp2/nachos-3.4-ualr-2022/code/lab5$ ./nachos -D  Bit map file header:  FileHeader contents.  File size: 128.  File blocks: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:  FileHeader contents.  File size: 200.  File blocks: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  FileHeader contents.  File size: 4864.  File blocks: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 (level2)36 37 38 39 40 41 42 43 44  Last modified time:2022-12-10 17:29:36  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  <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 54350, idle 52840, 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:~/oscp2/nachos-3.4-ualr-2022/code/lab5$ ./nachos -ap test/small huge  No threads ready or runnable, and no pending interrupts.  Assuming the program completed.  Machine halting!  Ticks: total 147020, idle 146170, system 850, user 0  Disk I/O: reads 20, writes 8  Console I/O: reads 0, writes 0  Paging: faults 0  Network I/O: packets received 0, sent 0  Cleaning up...  u1@ubuntu:~/oscp2/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:~/oscp2/nachos-3.4-ualr-2022/code/lab5$ ./nachos -cp test/medium medium  No 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:~/oscp2/nachos-3.4-ualr-2022/code/lab5$ ./nachos -DI  Total size: 1024 Sectors, 131072 Bytes  Used size: 51 Sectors, 6528 Bytes  Free size: 973 Sectors, 124544 Bytes  Size used by 3 normal files:  without internal fragments: 5092 Bytes  actually used: 5376 Bytes in 42 Sectors  fragmented: 284 Bytes 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:~/oscp2/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:~/oscp2/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:~/oscp2/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:~/oscp2/nachos-3.4-ualr-2022/code/lab5$ ./nachos -DI  Total size: 1024 Sectors, 131072 Bytes  Used size: 5 Sectors, 640 Bytes  Free size: 1019 Sectors, 130432 Bytes  Size used by 0 normal files:  without internal fragments: 0 Bytes  actually used: 0 Bytes in 0 Sectors  fragmented: 0 Bytes 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:~/oscp2/nachos-3.4-ualr-2022/code/lab5$ | |
| 结论分析与体会：  实验五的要求1主要是在原有文件头逻辑的基础上在各个方法中进行有关二级索引的扩展，逻辑上并不复杂，但有些细节需要注意，例如一些边界问题等。而在调用这些函数的部分则大多不需要改动，体现了封装的优良特性：调用者只关注返回的结果，实现流程交给下级函数，当某个功能需要扩展时修改对应的方法即可。  实验要求2则是促使我们再次仔细研究了文件系统的代码，特别是目录相关的逻辑，我们才得以统计出相应的信息。  此外，这个过程促使我对操作系统中文件存储管理等操作的具体实现进行复习，而经过了实验中的多次尝试又加深了理解，想必多级索引的实现也是类似的。这也启示我在今后的学习中也要注重理论实践相互结合，达到对知识的掌握和运用。  实验的要求2中新增了一个命令，通过这部分代码实现我学习了命令行中指令具体调用的流程，命令行变得没有先前那样玄妙了。  最后我要感谢为我提供技术支持的组员，在我看不懂源码时组员很认真地为我讲述他的理解，并和我讨论思路、一起查资料共同学习。调试过程中linux时常出现段错误的返回，多亏了组员们的支持鼓励让我最终坚持尝试并完成了实验要求。 | |