|  |
| --- |
| 实验目的：  Nachos系统原有的文件系统只支持单级索引，最大能存取NumDirect \* SectorSize = 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，设计并实现具有二级索引的文件系统。 |
| 硬件环境：  惠普品牌型号笔记本 Intel Core i5-8300 CPU 8GB内存 512GB SSD |
| 软件环境：  宿主机：Windows 10 21H1 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-LW |
| 实验步骤与内容： 5.1 实验内容 Nachos系统原有的文件系统只支持单级索引，最大能存取NumDirect \* SectorSize = 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，设计并实现具有二级索引的文件系统。    二级索引文件头i-node设计：  如上图所示，构建具有二级索引的i-node，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，并将其扇区号存入一级索引块的最后一项，形成上图所示的结构。 5.2 实现思路 一个sector里128个字节，sectorSize = 128, sizeof(int) = 4，则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。想要扩展文件系统，还是从存储文件字节的数组下手。建立一个dataSectors2数组，之前单级索引是用数组的最后1位作为索引号，dataSectors[0]-dataSectors[28]作为存取数据的块号；如果dataSectors[29]=-1，则表明没有没有二级索引块，如果dataSectors[29]为任意正值，则与二级索引块相应块号对应，即dataSectors2[]值。文件超过dataSectors[]可存放大小，则放到dataSector2[]中，dataSectors2[]返回的Sector号对应着dataSectors[-1]的值。 5.3 实验代码5.3.1 FileHeader中的Allocate（） bool FileHeader::Allocate(BitMap \*freeMap, int fileSize) {  numBytes = fileSize;  int lastindex = NumDirect - 1;  numSectors = divRoundUp(fileSize, SectorSize);  if (freeMap->NumClear() < numSectors)  return FALSE; // not enough space  else if (NumDirect + NumDirect2 <= numSectors)  return false;  if (numSectors < lastindex)  {  for (int i = 0; i < numSectors; i++)  dataSectors[i] = freeMap->Find();  dataSectors[lastindex] = -1;  }  else  {  for (int i = 0; i < lastindex; i++)  dataSectors[i] = freeMap->Find();  dataSectors[lastindex] = freeMap->Find();  int dataSector2[NumDirect2];  for (int i = 0; i < numSectors - NumDirect; i++)  dataSector2[i] = freeMap->Find();  synchDisk->WriteSector(dataSectors[lastindex], (char \*)dataSector2);  }  return TRUE; } 5.3.2 FileHeader中的deallocate（） void FileHeader::Deallocate(BitMap \*freeMap) {  int lastIndex = NumDirect - 1;  if (dataSectors[lastIndex] == -1)  {  for (int i = 0; i < numSectors; i++)  {  ASSERT(freeMap->Test((int)dataSectors[i])); // ought to be marked!  freeMap->Clear((int)dataSectors[i]);  }  }  else  {  int i = 0;  for (; i < lastIndex; i++)  {  ASSERT(freeMap->Test((int)dataSectors[i])); // ought to be marked!  freeMap->Clear((int)dataSectors[i]);  }  int dataSector2[NumDirect2];  synchDisk->ReadSector(dataSectors[lastIndex], (char \*)dataSector2);  freeMap->Clear((int)dataSectors[lastIndex]);  for (; i < numSectors; i++)  freeMap->Clear((int)dataSectors[i - lastIndex]);  } } 5.3.3 OpenFile中的Extend（） bool FileHeader::ExtendFileSize(int filesize) { // printf("start extend \n");  int newNumSectors = divRoundUp(filesize, SectorSize); //上取整   if (newNumSectors == numSectors)  {  numBytes = filesize;  return true; //扇区数量不变  }   int diffSector = newNumSectors - numSectors;   OpenFile \*bitmapfile = new OpenFile(0);  BitMap \*freeMap = new BitMap(NumSectors);  freeMap->FetchFrom(bitmapfile);   //printf("debug in fhdr extend where new Sector=%d \n",newNumSectors);  if (newNumSectors > (NumDirect + NumDirect2) || freeMap->NumClear() < diffSector)  {  return false; //磁盘空间不足  }  //allocate the new sectors and store them into file header  int i;   if (newNumSectors < NumDirect)  {   for (i = numSectors; i < newNumSectors; i++)  {  dataSectors[i] = freeMap->Find();  }  }  else  { // printf("start extend, append \n");  if (numSectors < NumDirect)  {  //原来无二级索引  for (i = numSectors; i < NumDirect; i++)  {  dataSectors[i] = freeMap->Find();  }  int dataSectors2[NumDirect2];  for (i = 0; i < newNumSectors - NumDirect + 1; i++)  dataSectors2[i] = freeMap->Find();  //将二级索引保存  synchDisk->WriteSector(dataSectors[NumDirect - 1], (char \*)dataSectors2);  }  else  { //原来有二级索引  int dataSectors2[NumDirect2];  synchDisk->ReadSector(dataSectors[NumDirect - 1], (char \*)dataSectors2);  for (i = numSectors - NumDirect + 1; i < newNumSectors - NumDirect + 1; i++)  dataSectors2[i] = freeMap->Find();  //将二级索引保存  synchDisk->WriteSector(dataSectors[NumDirect - 1], (char \*)dataSectors2);  }  }  numBytes = filesize;  numSectors = newNumSectors;   freeMap->WriteBack(bitmapfile);  // printf("finish extend, append \n"); // printf("finish extend \n");   return true; } |
| 结论分析与体会：  实验结果  ./nachos -f ./nachos -cp test/huge huge ./nachos -D      ./nachos -ap test/huge huge ./nachos -D      ./nachos -ap test/huge huge ./nachos -D      实验体会  本次实验修改在code/filesys中。修改部分仅为fileheader（filehdr），通过修改该部分将原本的一级索引改为二级索引。注意，这需要在实验四的基础上才能实现，请先去实现实验四，出于实现的考虑，将给ExtendSpace的参数中原本的（position+numBytes）修改为（position+numBytes-fileLength）才可以，切记切记。实验思路是利用一级索引最后一个扇区存储二级索引的索引号。dataSectors[i]里面存储的是扇区号（0-1023），通过这个扇区号可以找到一个128字节的扇区。使用WriteSector()方法将索引号写入dataSectors[29]所存储的那个扇区，用ReadSector()方法从dataSectors[29]所存储的那个扇区中读出索引号。两个方法的第一个参数都是扇区号，第二个参数是一个data\*对象，用来读写扇区。  我们可以发现nachos的文件系统是可以进行大规模扩展的，希望以后能继续对nachos的文件系统进行扩展。 |