MP4: File System
Team_56

Team members:

- 108020022 周昱宏
- 108071003 李彥璋

Contribution:

工作項目	負責人
Trace code	周昱宏、李彥璋
Implementation	李彥璋
Report	李彥璋

Part I.

(1) Explain how the NachOS FS manage and find free block space? Where is this information stored on the raw disk (which sector)?

filesys.cc

```
58  #define FreeMapSector 0
```

Sector containing the file header for the bitmap of free sectors, (sector 0).

```
81  FileSystem::FileSystem(bool format)
82  {
83      DEBUG(dbgFile, "Initializing the file system.");
84      if (format)
85      {
86          PersistentBitmap *freeMap = new PersistentBitmap(NumSectors);
87          Directory *directory = new Directory(NumDirEntries);
88          FileHeader *mapHdr = new FileHeader;
89          FileHeader *dirHdr = new FileHeader;
90
91          DEBUG(dbgFile, "Formatting the file system.");
92
93          // First, allocate space for FileHeaders for the directory and bitmap
94          // (make sure no one else grabs these!)
95          freeMap->Mark(FreeMapSector);
96          freeMap->Mark(DirectorySector);
```

在 FileSystem 建構子中，

首先建立 PersistentBitmap *freeMap，用以管理 block 的使用，

接著建立兩個 FileHeader，for this bitmap and directory respectively.

Mark() sets(=1) the "nth" bit in a bitmap，從上方 define 可知，FreeMapSector 是 0.

```
101         ASSERT(mapHdr->Allocate(freeMap, FreeMapFileSize));
```

Allocate space for the data blocks containing the contents of the bitmap files.

filehdr.cc

```
43  bool
44  FileHeader::Allocate(PersistentBitmap *freeMap, int fileSize)
45  {
46      numBytes = fileSize;
47      numSectors  = divRoundUp(fileSize, SectorSize);
48      if (freeMap->NumClear() < numSectors)
49      return FALSE;       // not enough space
50
51      for (int i = 0; i < numSectors; i++) {
52      dataSectors[i] = freeMap->FindAndSet();
53      // since we checked that there was enough free space,
54      // we expect this to succeed
55      ASSERT(dataSectors[i] >= 0);
56      }
57      return TRUE;
58  }
```

NumClear() returns how many bits are unallocated，和 numSectors 比較，檢查 freeMap 是否有足夠的空間可以用．如果夠用，就用 FindAndSet()尋找空間給剛剛的 FileHeader 使用．

bitmap.cc

FindAndSet()是 FS manage and find free block space 的關鍵步驟．

```
110  int
111  Bitmap::FindAndSet()
112  {
113      for (int i = 0; i < numBits; i++) {
114          if (!Test(i)) {
115              Mark(i);
116              return i;
117          }
118      }
119      return -1;
120  }
```

!Test() 表示該 bit 沒有被使用，Mark()則為標記使用該 bit.

回到 filesys.cc 建構子，

```
110          mapHdr->WriteBack(FreeMapSector);
```

Writes mapHdr header back to disk, before we can "Open".

之後 open file 的時候可以透過 filehdr 找到並開啟 file.

(2) What is the maximum disk size that can be handled by the current implementation? Explain why.

disk.cc

```
27   const int MagicSize = sizeof(int);
28   const int DiskSize = (MagicSize + (NumSectors * SectorSize));
```

disk.h

```
50 const int SectorSize = 128;   // number of bytes per disk sector
51 const int SectorsPerTrack  = 32;  // number of sectors per disk track
52 const int NumTracks = 32;   // number of tracks per disk
53 const int NumSectors = (SectorsPerTrack * NumTracks);
```

Disk size(maximum disk size)
= (4 + ((32*32) * (128))) = 131076 bytes.

(3) Explain how the NachOS FS manage the directory data structure? Where is this information stored on the raw disk (which sector)?

filesys.cc

```
59   #define DirectorySector      1
```

Sectors containing the file header for the directory of files, (sector 1).

基本上,

directory 的 header 創造的過程都跟問題(1)的 bitmap header 類似,

這裡只截一些重點.

```
FileHeader *dirHdr = new FileHeader;
freeMap->Mark(DirectorySector);
ASSERT(dirHdr->Allocate(freeMap, DirectoryFileSize));
dirHdr->WriteBack(DirectorySector);
```

接著說明 how the NachOS FS manage the directory data structure?

directory.cc

```
37  Directory::Directory(int size)
38  {
39      table = new DirectoryEntry[size];
40      tableSize = size;
41      for (int i = 0; i < tableSize; i++)
42      table[i].inUse = FALSE;
43  }
```

在 directory 的建構子中，

建立紀錄 directoryEntey 的 table，inUse(directoryEntry 有沒有被使用)

預設為 False.

directory.h

```
32 class DirectoryEntry {
33   public:
34     bool inUse;       // Is this directory entry in use?
35     int sector;       // Location on disk to find the
36           //   FileHeader for this file
37     char name[FileNameMaxLen + 1];  // Text name for file, with +1 for
38           // the trailing '\0'
39 };
```

inUse 儲存該 directory entry 有沒有被使用.

sector 儲存該 directory entry 紀錄的 file 的 fileHeader 在 disk 上的哪一個位置(sector).

name 儲存 file 的名稱.

(4) Explain what information is stored in an inode, and use a figure to illustrate the disk allocation scheme of current implementation.

inode 是這裡的 fileHeader.

filehdr.h

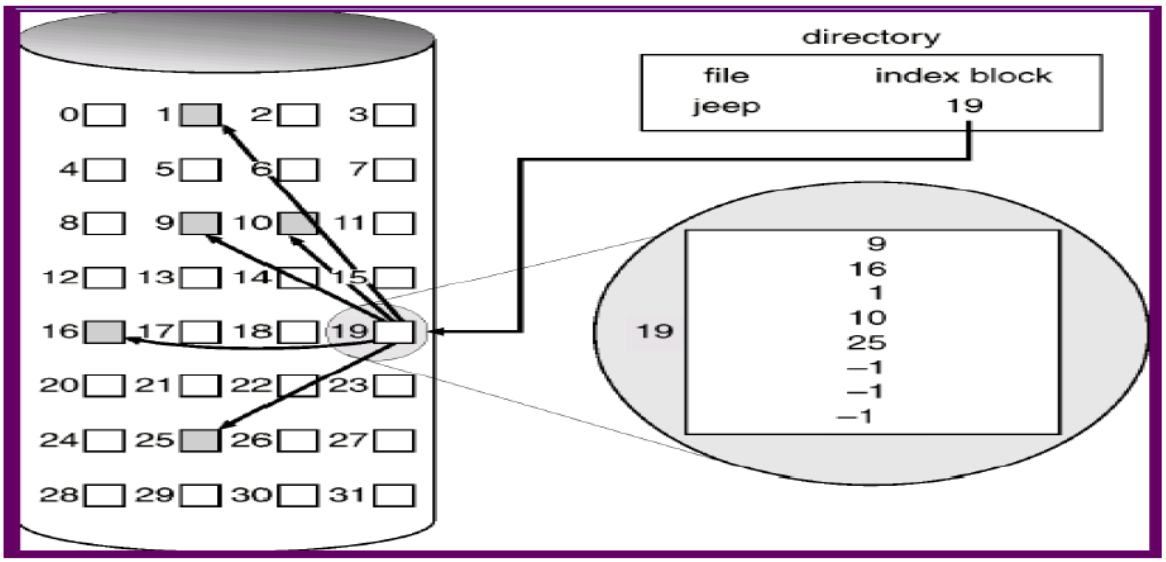
```
59  private:
60    int numBytes;      // Number of bytes in the file
61    int numSectors;      // Number of data sectors in the file
62    int dataSectors[NumDirect];   // Disk sector numbers for each data
63          // block in the file
```

numBytes: 一個檔案有多少個 byte.

numSectors: 一個檔案有多少個 data sector.

dataSectors[]: 檔案中每個 data block 的 disk sectoer number.

allocation scheme -> Indexed Allocation.



(5) Why is a file limited to 4KB in the current implementation?

filehdr.h

```
20 #define NumDirect   ((SectorSize - 2 * sizeof(int)) / sizeof(int))
21 #define MaxFileSize   (NumDirect * SectorSize)
```

SectorSize 原本是 128 bytes,
扣掉 metadata numBytes 以及 numSectors 各 4 bytes,
剩下 120 bytes 除以儲存 int(4 bytes),
等於可以存的 data sector 有 30 格.

而 MaxFileSize = 30(上面算出來的) * 128(一個 sector 128 bytes)
= 3840 bytes 大約等於 4KB.

Part II.

(1) Combine your MP1 file system call interface with NachOS FS.

跟 MP1 差不多，只貼 code

ksyscall.h

```
30 int SysCreate(char *name, int size) {
31   return kernel->fileSystem->Create(name, size);
32 }
33
34 OpenFileId SysOpen(char* name) {
35   return kernel->fileSystem->OpenAFile(name);
36 }
37
38 int SysWrite(char *buffer, int size, OpenFileId id) {
39   return kernel->fileSystem->Write(buffer, size, id);
40 }
41
42 int SysRead(char *buffer, int size, OpenFileId id) {
43   return kernel->fileSystem->Read(buffer, size, id);
44 }
45
46 int SysClose(OpenFileId id) {
47   return kernel->fileSystem->CloseAFile();
48 }
```

exception.cc

SC_Create

```
80          case SC_Create:
81              val = kernel->machine->ReadRegister(4);
82              {
83                  char *filename = &(kernel->machine->mainMemory[val]);
84                  int size = kernel->machine->ReadRegister(5);
85                  status = SysCreate(filename, size);
86                  kernel->machine->WriteRegister(2, (int) status);
87              }
88              kernel->machine->WriteRegister(PrevPCReg, kernel->machine->ReadRegister(PCReg));
89              kernel->machine->WriteRegister(PCReg, kernel->machine->ReadRegister(PCReg) + 4);
90              kernel->machine->WriteRegister(NextPCReg, kernel->machine->ReadRegister(PCReg) + 4);
91              return;
92              ASSERTNOTREACHED();
93              break;
```

SC_Open、SC_Read、SC_Write、SC_Close 皆類似.

(2) Implement five system calls.

filesys.cc

```
411  int FileSystem::Write(char *buffer, int size, OpenFileId id) {
412      return CurOpenFile->Write(buffer, size);
413  }
414
415  int FileSystem::Read(char *buffer, int size, OpenFileId id) {
416      return CurOpenFile->Read(buffer,size);
417  }
418
419  OpenFileId FileSystem::OpenAFile(char *name) {
420      CurOpenFile = Open(name);
421      return 1;
422  }
423
424  int FileSystem::CloseAFile() {
425      CurOpenFile = NULL;
426      return 1;
427  }
```

這些是剛剛 ksyscall.h 裡面呼叫的，

其中 Write()、Read()會以 OpenFile* CurOpenFile 再去呼叫 openfile.cc 當中的 Write()、Read()，執行讀寫檔案的動作.

CloseAFile()會將 OpenFile* CurOpenFile 設為 NULL.

```
121   OpenFile* CurOpenFile;
```

OpenFile* CurOpenFile 定義在 filesys.h 當中，用以紀錄當前開啟檔案(一次只會有一個檔案被開啟).

OpenAFile()會呼叫 filesys.cc 當中的另一個函數 Open()，

而因為 Open()、Create()會牽涉到 directory 路經問題，於 part III 再說明.

(3) Enhance the FS to let it support up to 32KB file size.

因為 bonusI 要 64MB，所以我們直接提高到可以容 64MB.

filehdr.h

```
25 #define MaxFileSize1  (NumDirect * SectorSize)
26 #define MaxFileSize2  (NumDirect * NumDirect * SectorSize)
27 #define MaxFileSize3  (NumDirect * NumDirect * NumDirect * SectorSize)
28 #define MaxFileSize4  (NumDirect * NumDirect * NumDirect * NumDirect * SectorSize)
```

MaxFileSize1: 30 * 128 = 3840 bytes (4KB).
MaxFileSize2: 30 * 30 * 128 = 115200 bytes (115KB).
MaxFileSize3: 30 * 30 * 30 * 128 = 3456000 bytes (3.5MB).
MaxFileSize4: 30 * 30 * 30 * 30 * 128 = 103680000 bytes (100MB).

disk.h

要修改 disk.h 當中的 NumTracks，改變 nachOS disk 的大小.

```
52 const int NumTracks = 17000;
```

(計算公式 by Part I. (2))
64 MB / 128 B = 500000
500000 / 32 = 15625(NumTracks)

Allocate

filehdr.h

```
73   void MultiLevelAlloc(PersistentBitmap *freeMap, int fileSize, int maxFileSize);
```

新增函數 MultiLevelAlloc().

filehdr.cc

```
 96  bool FileHeader::Allocate(PersistentBitmap *freeMap, int fileSize)
 97  {
 98      numBytes = fileSize;
 99      numSectors = divRoundUp(fileSize, SectorSize);
100
101      // NumClear() Return the number of clear bits in the bitmap.
102      // In other words, how many bits are unallocated?
103      if (freeMap->NumClear() < numSectors)
104          return FALSE;
105
106      if(fileSize > MaxFileSize4) {
107          MultiLevelAlloc(freeMap, fileSize, MaxFileSize4);
108      }
109      else if(fileSize > MaxFileSize3) {
110          MultiLevelAlloc(freeMap, fileSize, MaxFileSize3);
111      }
112      else if(fileSize > MaxFileSize2) {
113          MultiLevelAlloc(freeMap, fileSize, MaxFileSize2);
114      }
115      else if(fileSize > MaxFileSize1) {
116          MultiLevelAlloc(freeMap, fileSize, MaxFileSize1);
117      }
```

```
118     else {
119         for (int i=0 ; i<numSectors ; i++) {
120             dataSectors[i] = freeMap->FindAndSet();
121             // since we checked that there was enough free space,
122             // we expect this to succeed
123             ASSERT(dataSectors[i] >= 0);
124 
125             char* init = new char[SectorSize]();
126             // WriteSector()
127             // Write the contents of a buffer into a disk sector.
128             kernel->synchDisk->WriteSector(dataSectors[i], init);
129             delete init;
130         }
131     }
132     return TRUE;
133 }
```

Allocate()首先根據(fileSize 除以 SectorSize)，算出這個 file 需要多少 sector 來存.

接著根據 fileSize 決定需要分配幾層 index，大於 MaxFile1、2、3、4 的話，呼叫 MultiLevelAlloc().

```
70  void FileHeader::MultiLevelAlloc(PersistentBitmap *freeMap, int fileSize, int maxFileSize) {
71      int i = 0;
72      while(fileSize > 0) {
73          // FindAndSet() return the number of the first bit which is clear.
74          // As a side effect, set the bit (mark it as in use).
75          // In other words, find and allocate a bit.
76          dataSectors[i] = freeMap->FindAndSet();
77 
78          FileHeader *subHdr = new FileHeader();
79 
80          if(fileSize > maxFileSize) {
81              subHdr->Allocate(freeMap, maxFileSize);
82          }
83          else {
84              subHdr->Allocate(freeMap, fileSize);
85          }
86 
87          fileSize = fileSize - maxFileSize;
88 
89          // WriteBack()
90          // Write the modified contents of the file header back to disk.
91          subHdr->WriteBack(dataSectors[i]);
92          i++;
93      }
94  }
```

MultiLevelAlloc()當 fileSize 還大於 0，就會 while 下去，用一個 dataSectors[i]儲存指向下一層 table 的 sector num.

接著，if-else 判斷剩餘的檔案大小是否大於 maxFileSize，以 subHdr 遞迴呼叫下層 Allocate，當到達 Allocate()最下面的 else 時，先清空要儲存的 disk 的位置(以免有其他雜訊)，並在 MultiLevelAlloc()寫入.

最後，更新 fileSize 的值、i++逕行下一次的 while.

Deallocate

filehdr.cc

```
145  void FileHeader::Deallocate(PersistentBitmap *freeMap)
146  {
147      if(numBytes > MaxFileSize1) {
148          for(int i=0 ; i<numSectors ; i++) {
149              FileHeader *subhdr = new FileHeader();
150              subhdr->FetchFrom(dataSectors[i]);
151              subhdr->Deallocate(freeMap);
152          }
153      }
154      else {
155          for(int i=0 ; i<numSectors ; i++) {
156              // Clear()
157              // Clear the "nth" bit in a bitmap.
158              freeMap->Clear((int) dataSectors[i]);
159          }
160      }
161  }
```

和 Allocate()一樣，都是從上層一直往下層叫 Deallocate()，當到達最下層時，Clear()會清除剛剛 FetchFrom()的 content(of filehdr from disk).

ByteToSector

filehdr.h

```
74   void ByteToSectorTool(int offset, int maxFileSize);
```

新增函數 ByteToSectorTool().

filehdr.cc

```
213  void FileHeader::ByteToSectorTool(int offset, int maxFileSize) {
214      int entry = divRoundDown(offset, maxFileSize);
215      FileHeader *subhdr = new FileHeader;
216      subhdr->FetchFrom(dataSectors[entry]);
217      subhdr->ByteToSector(offset - maxFileSize*entry);
218  }
219
220  int FileHeader::ByteToSector(int offset)
221  {
222      if(numBytes > MaxFileSize4) ByteToSectorTool(offset, MaxFileSize4);
223      else if(numBytes > MaxFileSize3) ByteToSectorTool(offset, MaxFileSize3);
224      else if(numBytes > MaxFileSize2) ByteToSectorTool(offset, MaxFileSize2);
225      else if(numBytes > MaxFileSize1) ByteToSectorTool(offset, MaxFileSize1);
226      else return (dataSectors[offset / SectorSize]);
227  }
```

ByteToSector() return which disk sector is storing a particular byte within the file,

是
virtual address (the offset in the file)
轉換至
physical address (the sector where the data at the offset is stored)
的函數.

也是根據 numBytes 遞迴向下層叫, 要注意的地方是
[214] divRoundDown(), 因為 entry 是從 0 開始數、
[217] 遞迴呼叫時記得扣掉 maxFilseSize * entry.

Print
filehdr.h

```
75    void PrintTool();
```

新增 PrintTool()函數.

filehdr.cc
原本的 Print()只要新增一個 if, 並把其他部分變成 else(也是從上層往下層叫的概念).

```
if(numBytes > MaxFileSize1) {
    PrintTool();
}
```

```
246  void FileHeader::PrintTool() {
247      for(int i=0 ; i<numSectors ; i++) {
248          OpenFile *opfile = new OpenFile(dataSectors[i]);
249          FileHeader *subhdr = opfile->getHdr();
250          subhdr->Print();
251      }
252  }
```

Part III.

(1) Implement the subdirectory structure.

Create
filesys.cc

```
189 bool FileSystem::Create(char *name, int initialSize)
190 {
191     Directory *directory;
192     PersistentBitmap *freeMap;
193     FileHeader *hdr;
194     int sector;
195     bool success;
196
197     directory = new Directory(NumDirEntries);
198     directory->FetchFrom(directoryFile);
199
200     OpenFile *open_tool = directoryFile;
201     OpenFile *belong_dir = open_tool;
202     char *token = strtok(name, "/");
203     char *pre_token = token;
204
205     while(token != NULL) {
206         sector = directory->Find(token);
207         if(sector == -1) {
208             break;
209         }
211         belong_dir = open_tool;
212         open_tool = new OpenFile(sector);
213         directory->FetchFrom(open_tool);
214
215         pre_token = token;
216         token = strtok(NULL, "/");
217     }
218
219     if(token == NULL) {
220         token = pre_token;
221         directory->FetchFrom(belong_dir);
222     }
223
224     if(directory->Find(token) != -1) {
225         success = FALSE; // file is already in directory
226     }
227     else {
228         freeMap = new PersistentBitmap(freeMapFile, NumSectors);
229         sector = freeMap->FindAndSet(); // find a sector to hold the file header
230         if(sector == -1) {
231             success = FALSE; // no free block for file header
232         }
```

```
233         else if(!directory->Add(token, sector, FALSE)) {
234             success = FALSE; // no space in directory
235         }
236         else {
237             hdr = new FileHeader;
238             if (!hdr->Allocate(freeMap, initialSize)) {
239                 success = FALSE; // no space on disk for data
240             }
241             else {
242                 success = TRUE;
243                 // everthing worked, flush all changes back to disk
244                 hdr->WriteBack(sector);
245                 directory->WriteBack(open_tool);
246                 freeMap->WriteBack(freeMapFile);
247             }
248             delete hdr;
249         }
250         delete freeMap;
251     }
252     delete directory;
253     return success;
254 }
```

以 while 迴圈搭配 strtok()這個函數拆解 path，目標是要知道這個檔案要 create 在哪個 subdirectory 下、這個檔案的 name 是甚麼.

首先 Find() Look up file name in directory, and return the disk sector number where the file's header is stored, 如果回傳值是-1, 代表這就是我們要 create 的檔案, break while.
如果不是-1, 繼續 strtok()拆解直到 token == NULL.
(pre_token 會記錄前一個 token 的值, 而可以開的 directory 則會一直開).

離開 while 迴圈後, 如果 token == NULL, 代表剛剛的 path 每個 directory 或 file 都有找到, 但我們必須回到前一個 directory(可能該 file 早就存在了).

接著和原本 NachOS 的 Create()差不多, 檢查
1. 檔案是否已經存在.
2. 有沒有 free block 給 file hdr.
3. directory 裡面有沒有空間.
4. disk 有沒有空間.
如果上面的條件都通過, 就可以把檔案存進 disk 裡面.

注意[233] Add()新增了一個傳遞參數 FALSE, 代表這是 file 不是 directory.

CreateDir

filesys.cc

基本上和 Create()都一樣，不一樣的地方：

1. Add()裡面傳 TRUE，表示這是一個 directory.

2. Allocate 的時候，傳進去的 size 是 DirectoryFileSize (Create()的時候是 initialSize).

Open

filesys.cc

基本上拆解 path 的方法和上面的 Create()差不多.

有關 Create、Open

main.cc

```
100     char copied[256]; //有改
101     strcpy(copied, to); //有改
102
103     if (!kernel->fileSystem->Create(to, fileLength))
104     { // Create Nachos file
105         printf("Copy: couldn't create output file %s\n", to);
106         Close(fd);
107         return;
108     }
109
110     openFile = kernel->fileSystem->Open(copied); //有改
111     ASSERT(openFile != NULL);
```

在 main.cc 的 Copy()當中(也就是使用者打了"-cp"這個指令後會的)，

記得要先複製要創造檔案的 path(也就是 to)，如果沒複製，跑完 Create()後 Open()會開不起檔案，可能是 path 被拆解掉了.

List

main.cc

```
343     if (dirListFlag)
344     {
345         kernel->fileSystem->List(listDirectoryName, recursiveListFlag);
346     }
```

新增 List()裡面兩個傳遞參數.

filesys.h

```
100   void List(char *dirName, bool doRecursive);
```

同上.

filesys.cc

```
443 void FileSystem::List(char *name, bool doRecursive)
444 {
445     Directory *directory = new Directory(NumDirEntries);
446     directory->FetchFrom(directoryFile);
447 
448     int sector;
449     OpenFile *open_tool = directoryFile;
450 
451     //int root = (!strcmp(name, "/")) ? 1 : 0;
452 
453     char *token = strtok(name, "/");
454     char *pre_token = token;
455 
456     //if(!root) {
457         while(token != NULL) {
458             sector = directory->Find(token);
459             open_tool = new OpenFile(sector);
460             directory->FetchFrom(open_tool);
461             pre_token = token;
462             token = strtok(NULL, "/");
463         }
464         //token = pre_token;
465     //}
466     //else {
467         //token = name;
468     //}
469 
470     if(doRecursive) {
471         directory->RecursiveList(0);
472     } else {
473         directory->List();
474     }
475 
476     delete directory;
477 }
```

同樣是先拆解 path，先到我們要 list 的那個 directory 裡面，

再以 doRecursive 決定是否要遞迴 List(呼叫 directory 裡的 List()).

directory.h

```
77      bool checkIfDir(char *name);
78      void RecursiveList(int space);
```

新增兩個 function.

directory.cc

```
122  bool Directory::checkIfDir(char *name) {
123      int Index = FindIndex(name);
124      return Index >= 0 && table[Index].isDir;
125  }
```

判斷這個 file 是 file 還是 directory.

isDir 會在 Create()、CreateDir()的時候就被寫入.

```
180  void Directory::RecursiveList(int space) {
181      Directory *subDirectory = new Directory(NumDirEntries);
182      OpenFile *open_tool;
183
184      for (int i = 0; i < tableSize; i++) {
185          if (table[i].inUse) {
186              for(int j=0 ; j<space ; j++) {
187                  printf("  ");
188              }
189              char d_f = table[i].isDir ? 'D' : 'F';
190              printf("[%c]: %s\n", d_f, table[i].name);
191              if(table[i].isDir) {
192                  open_tool = new OpenFile(table[i].sector);
193                  subDirectory->FetchFrom(open_tool);
194                  subDirectory->RecursiveList(space+1);
195              }
196          }
197      }
198  }
```

傳進來的 space 代表要縮排多少，如果 inUse==TRUE，就可以縮排.

以 isDir 判斷是 directory 還是 file，如果是 directory，就遞迴呼叫 RecursiveList()，space 記得+1.

(2) Support up to 64 files/subdirectories per directory.

directory.cc

```
27   #define NumDirEntries    64
```

改成 64.

Bonus I.

已於 Part II.

(3) Enhance the FS to let it support up to 32KB file size, 說明.

Result.

./FS_partII_a.sh

```
[os21team56@localhost test]$ ./FS_partII_a.sh
rm -f -f *.o *.ii
rm -f -f *.coff
../../usr/local/nachos/bin/decstation-ultrix-gcc -G 0 -c -I.
/nachos/decstation-ultrix/bin/ -c FS_test1.c
decstation-ultrix-gcc: file path prefix `/usr/bin/local/nach
../../usr/local/nachos/bin/decstation-ultrix-gcc -G 0 -c -I.
/nachos/decstation-ultrix/bin/ -mips2 -c start.S
decstation-ultrix-gcc: file path prefix `/usr/bin/local/nach
../../usr/local/nachos/bin/decstation-ultrix-ld -T script -N
../../coff2noff/coff2noff.x86Linux FS_test1.coff FS_test1
numsections 4
Loading 4 sections:
        ".text", filepos 0xf0, mempos 0x0, size 0x2f0
        ".rdata", filepos 0x3e0, mempos 0x2f0, size 0x90
        ".data", filepos 0x470, mempos 0x380, size 0x0
        ".bss", filepos 0x0, mempos 0x380, size 0x0
../../usr/local/nachos/bin/decstation-ultrix-gcc -G 0 -c -I.
/nachos/decstation-ultrix/bin/ -c FS_test2.c
decstation-ultrix-gcc: file path prefix `/usr/bin/local/nach
../../usr/local/nachos/bin/decstation-ultrix-ld -T script -N
../../coff2noff/coff2noff.x86Linux FS_test2.coff FS_test2
numsections 4
Loading 4 sections:
        ".text", filepos 0xf0, mempos 0x0, size 0x300
        ".rdata", filepos 0x3f0, mempos 0x300, size 0xa0
        ".data", filepos 0x490, mempos 0x3a0, size 0x0
        ".bss", filepos 0x0, mempos 0x3a0, size 0x0
/FS_test1
abcdefghijklmnopqrstuvwxyz
/FS_test2
Passed! ^_^
[os21team56@localhost test]$
```

./FS_partII_b.sh

```
[os21team56@localhost test]$ ./FS_partII_b.sh
000000001 000000002 000000003 000000004 000000005 000000006 000000007 000000008 000000009 000000010
000000011 000000012 000000013 000000014 000000015 000000016 000000017 000000018 000000019 000000020
000000021 000000022 000000023 000000024 000000025 000000026 000000027 000000028 000000029 000000030
000000031 000000032 000000033 000000034 000000035 000000036 000000037 000000038 000000039 000000040
000000041 000000042 000000043 000000044 000000045 000000046 000000047 000000048 000000049 000000050
000000051 000000052 000000053 000000054 000000055 000000056 000000057 000000058 000000059 000000060
000000061 000000062 000000063 000000064 000000065 000000066 000000067 000000068 000000069 000000070
000000071 000000072 000000073 000000074 000000075 000000076 000000077 000000078 000000079 000000080
000000081 000000082 000000083 000000084 000000085 000000086 000000087 000000088 000000089 000000090
000000091 000000092 000000093 000000094 000000095 000000096 000000097 000000098 000000099 000000100
000000101 000000102 000000103 000000104 000000105 000000106 000000107 000000108 000000109 000000110
000000111 000000112 000000113 000000114 000000115 000000116 000000117 000000118 000000119 000000120
000000121 000000122 000000123 000000124 000000125 000000126 000000127 000000128 000000129 000000130
000000131 000000132 000000133 000000134 000000135 000000136 000000137 000000138 000000139 000000140
000000141 000000142 000000143 000000144 000000145 000000146 000000147 000000148 000000149 000000150
000000151 000000152 000000153 000000154 000000155 000000156 000000157 000000158 000000159 000000160
000000161 000000162 000000163 000000164 000000165 000000166 000000167 000000168 000000169 000000170
000000171 000000172 000000173 000000174 000000175 000000176 000000177 000000178 000000179 000000180
000000181 000000182 000000183 000000184 000000185 000000186 000000187 000000188 000000189 000000190
000000191 000000192 000000193 000000194 000000195 000000196 000000197 000000198 000000199 000000200
000000201 000000202 000000203 000000204 000000205 000000206 000000207 000000208 000000209 000000210
000000211 000000212 000000213 000000214 000000215 000000216 000000217 000000218 000000219 000000220
000000221 000000222 000000223 000000224 000000225 000000226 000000227 000000228 000000229 000000230
000000231 000000232 000000233 000000234 000000235 000000236 000000237 000000238 000000239 000000240
000000241 000000242 000000243 000000244 000000245 000000246 000000247 000000248 000000249 000000250
000000251 000000252 000000253 000000254 000000255 000000256 000000257 000000258 000000259 000000260
000000261 000000262 000000263 000000264 000000265 000000266 000000267 000000268 000000269 000000270
000000271 000000272 000000273 000000274 000000275 000000276 000000277 000000278 000000279 000000280
000000281 000000282 000000283 000000284 000000285 000000286 000000287 000000288 000000289 000000290
000000291 000000292 000000293 000000294 000000295 000000296 000000297 000000298 000000299 000000300
000000301 000000302 000000303 000000304 000000305 000000306 000000307 000000308 000000309 000000310
000000311 000000312 000000313 000000314 000000315 000000316 000000317 000000318 000000319 000000320
000000321 000000322 000000323 000000324 000000325 000000326 000000327 000000328 000000329 000000330
000000331 000000332 000000333 000000334 000000335 000000336 000000337 000000338 000000339 000000340
000000341 000000342 000000343 000000344 000000345 000000346 000000347 000000348 000000349 000000350
000000351 000000352 000000353 000000354 000000355 000000356 000000357 000000358 000000359 000000360
000000361 000000362 000000363 000000364 000000365 000000366 000000367 000000368 000000369 000000370
000000371 000000372 000000373 000000374 000000375 000000376 000000377 000000378 000000379 000000380
000000381 000000382 000000383 000000384 000000385 000000386 000000387 000000388 000000389 000000390
000000391 000000392 000000393 000000394 000000395 000000396 000000397 000000398 000000399 000000400
```

...

```
000000891 000000892 000000893 000000894 000000895 000000896 000000897 000000898 000000899 000000900
000000901 000000902 000000903 000000904 000000905 000000906 000000907 000000908 000000909 000000910
000000911 000000912 000000913 000000914 000000915 000000916 000000917 000000918 000000919 000000920
000000921 000000922 000000923 000000924 000000925 000000926 000000927 000000928 000000929 000000930
000000931 000000932 000000933 000000934 000000935 000000936 000000937 000000938 000000939 000000940
000000941 000000942 000000943 000000944 000000945 000000946 000000947 000000948 000000949 000000950
000000951 000000952 000000953 000000954 000000955 000000956 000000957 000000958 000000959 000000960
000000961 000000962 000000963 000000964 000000965 000000966 000000967 000000968 000000969 000000970
000000971 000000972 000000973 000000974 000000975 000000976 000000977 000000978 000000979 000000980
000000981 000000982 000000983 000000984 000000985 000000986 000000987 000000988 000000989 000000990
000000991 000000992 000000993 000000994 000000995 000000996 000000997 000000998 000000999 000001000
========================================
000000001 000000002 000000003 000000004 000000005 000000006 000000007 000000008 000000009 000000010
000000011 000000012 000000013 000000014 000000015 000000016 000000017 000000018 000000019 000000020
000000021 000000022 000000023 000000024 000000025 000000026 000000027 000000028 000000029 000000030
000000031 000000032 000000033 000000034 000000035 000000036 000000037 000000038 000000039 000000040
000000041 000000042 000000043 000000044 000000045 000000046 000000047 000000048 000000049 000000050
000000051 000000052 000000053 000000054 000000055 000000056 000000057 000000058 000000059 000000060
000000061 000000062 000000063 000000064 000000065 000000066 000000067 000000068 000000069 000000070
000000071 000000072 000000073 000000074 000000075 000000076 000000077 000000078 000000079 000000080
000000081 000000082 000000083 000000084 000000085 000000086 000000087 000000088 000000089 000000090
000000091 000000092 000000093 000000094 000000095 000000096 000000097 000000098 000000099 000000100
========================================
[F]: 100
[F]: 1000
[os21team56@localhost test]$
```

./FS_partIII.sh

```
[os21team56@localhost test]$ ./FS_partIII.sh
[D]: t0
[D]: t1
[D]: t2
========================================
[F]: f1
[D]: aa
[D]: bb
[D]: cc
========================================
[D]: t0
  [F]: f1
  [D]: aa
  [D]: bb
    [F]: f1
    [F]: f2
    [F]: f3
    [F]: f4
  [D]: cc
[D]: t1
[D]: t2
========================================
000000001 000000002 000000003 000000004 000000005 000000006 000000007 000000008 000000009 000000010
000000011 000000012 000000013 000000014 000000015 000000016 000000017 000000018 000000019 000000020
000000021 000000022 000000023 000000024 000000025 000000026 000000027 000000028 000000029 000000030
000000031 000000032 000000033 000000034 000000035 000000036 000000037 000000038 000000039 000000040
000000041 000000042 000000043 000000044 000000045 000000046 000000047 000000048 000000049 000000050
000000051 000000052 000000053 000000054 000000055 000000056 000000057 000000058 000000059 000000060
000000061 000000062 000000063 000000064 000000065 000000066 000000067 000000068 000000069 000000070
000000071 000000072 000000073 000000074 000000075 000000076 000000077 000000078 000000079 000000080
000000081 000000082 000000083 000000084 000000085 000000086 000000087 000000088 000000089 000000090
000000091 000000092 000000093 000000094 000000095 000000096 000000097 000000098 000000099 000000100
========================================
000000001 000000002 000000003 000000004 000000005 000000006 000000007 000000008 000000009 000000010
000000011 000000012 000000013 000000014 000000015 000000016 000000017 000000018 000000019 000000020
000000021 000000022 000000023 000000024 000000025 000000026 000000027 000000028 000000029 000000030
000000031 000000032 000000033 000000034 000000035 000000036 000000037 000000038 000000039 000000040
000000041 000000042 000000043 000000044 000000045 000000046 000000047 000000048 000000049 000000050
000000051 000000052 000000053 000000054 000000055 000000056 000000057 000000058 000000059 000000060
000000061 000000062 000000063 000000064 000000065 000000066 000000067 000000068 000000069 000000070
000000071 000000072 000000073 000000074 000000075 000000076 000000077 000000078 000000079 000000080
000000081 000000082 000000083 000000084 000000085 000000086 000000087 000000088 000000089 000000090
000000091 000000092 000000093 000000094 000000095 000000096 000000097 000000098 000000099 000000100
[os21team56@localhost test]$
```

Bounus I

```
[os21team56@localhost test]$ ../build.linux/nachos -f
[os21team56@localhost test]$ ../build.linux/nachos -cp num_1000000.txt /1000000
[os21team56@localhost test]$ ../build.linux/nachos -p /1000000
000000001 000000002 000000003 000000004 000000005 000000006 000000007 000000008 000000009 000000010
000000011 000000012 000000013 000000014 000000015 000000016 000000017 000000018 000000019 000000020
000000021 000000022 000000023 000000024 000000025 000000026 000000027 000000028 000000029 000000030
000000031 000000032 000000033 000000034 000000035 000000036 000000037 000000038 000000039 000000040
000000041 000000042 000000043 000000044 000000045 000000046 000000047 000000048 000000049 000000050
000000051 000000052 000000053 000000054 000000055 000000056 000000057 000000058 000000059 000000060
000000061 000000062 000000063 000000064 000000065 000000066 000000067 000000068 000000069 000000070
000000071 000000072 000000073 000000074 000000075 000000076 000000077 000000078 000000079 000000080
000000081 000000082 000000083 000000084 000000085 000000086 000000087 000000088 000000089 000000090
000000091 000000092 000000093 000000094 000000095 000000096 000000097 000000098 000000099 000000100
```

...

...

...

...

```
000999591 000999592 000999593 000999594 000999595 000999596 000999597 000999598 000999599 000999600
000999601 000999602 000999603 000999604 000999605 000999606 000999607 000999608 000999609 000999610
000999611 000999612 000999613 000999614 000999615 000999616 000999617 000999618 000999619 000999620
000999621 000999622 000999623 000999624 000999625 000999626 000999627 000999628 000999629 000999630
000999631 000999632 000999633 000999634 000999635 000999636 000999637 000999638 000999639 000999640
000999641 000999642 000999643 000999644 000999645 000999646 000999647 000999648 000999649 000999650
000999651 000999652 000999653 000999654 000999655 000999656 000999657 000999658 000999659 000999660
000999661 000999662 000999663 000999664 000999665 000999666 000999667 000999668 000999669 000999670
000999671 000999672 000999673 000999674 000999675 000999676 000999677 000999678 000999679 000999680
000999681 000999682 000999683 000999684 000999685 000999686 000999687 000999688 000999689 000999690
000999691 000999692 000999693 000999694 000999695 000999696 000999697 000999698 000999699 000999700
000999701 000999702 000999703 000999704 000999705 000999706 000999707 000999708 000999709 000999710
000999711 000999712 000999713 000999714 000999715 000999716 000999717 000999718 000999719 000999720
000999721 000999722 000999723 000999724 000999725 000999726 000999727 000999728 000999729 000999730
000999731 000999732 000999733 000999734 000999735 000999736 000999737 000999738 000999739 000999740
000999741 000999742 000999743 000999744 000999745 000999746 000999747 000999748 000999749 000999750
000999751 000999752 000999753 000999754 000999755 000999756 000999757 000999758 000999759 000999760
000999761 000999762 000999763 000999764 000999765 000999766 000999767 000999768 000999769 000999770
000999771 000999772 000999773 000999774 000999775 000999776 000999777 000999778 000999779 000999780
000999781 000999782 000999783 000999784 000999785 000999786 000999787 000999788 000999789 000999790
000999791 000999792 000999793 000999794 000999795 000999796 000999797 000999798 000999799 000999800
000999801 000999802 000999803 000999804 000999805 000999806 000999807 000999808 000999809 000999810
000999811 000999812 000999813 000999814 000999815 000999816 000999817 000999818 000999819 000999820
000999821 000999822 000999823 000999824 000999825 000999826 000999827 000999828 000999829 000999830
000999831 000999832 000999833 000999834 000999835 000999836 000999837 000999838 000999839 000999840
000999841 000999842 000999843 000999844 000999845 000999846 000999847 000999848 000999849 000999850
000999851 000999852 000999853 000999854 000999855 000999856 000999857 000999858 000999859 000999860
000999861 000999862 000999863 000999864 000999865 000999866 000999867 000999868 000999869 000999870
000999871 000999872 000999873 000999874 000999875 000999876 000999877 000999878 000999879 000999880
000999881 000999882 000999883 000999884 000999885 000999886 000999887 000999888 000999889 000999890
000999891 000999892 000999893 000999894 000999895 000999896 000999897 000999898 000999899 000999900
000999901 000999902 000999903 000999904 000999905 000999906 000999907 000999908 000999909 000999910
000999911 000999912 000999913 000999914 000999915 000999916 000999917 000999918 000999919 000999920
000999921 000999922 000999923 000999924 000999925 000999926 000999927 000999928 000999929 000999930
000999931 000999932 000999933 000999934 000999935 000999936 000999937 000999938 000999939 000999940
000999941 000999942 000999943 000999944 000999945 000999946 000999947 000999948 000999949 000999950
000999951 000999952 000999953 000999954 000999955 000999956 000999957 000999958 000999959 000999960
000999961 000999962 000999963 000999964 000999965 000999966 000999967 000999968 000999969 000999970
000999971 000999972 000999973 000999974 000999975 000999976 000999977 000999978 000999979 000999980
000999981 000999982 000999983 000999984 000999985 000999986 000999987 000999988 000999989 000999990
000999991 000999992 000999993 000999994 000999995 000999996 000999997 000999998 000999999 001000000
[os21team56@localhost test]$
```