# Cover Page

## Team61

## Member List

> 108070039 羅稚涵
> 108070038 簡志宇

## Member Contribution

> Trace code 簡志宇、羅稚涵
> Part I report 簡志宇、羅稚涵
> Part II report 簡志宇、羅稚涵
> Implementation 簡志宇

# Trace Code

- 1.
  - ANS：

    (1) 照以下說明的流程

    (2) sector 0
  - kernel.cc

    最一開始是從main.cc呼叫Kernel::Initialize()，可以看到 Kernel 在 initialize 時就創了一個 File System(以下簡稱FS)，且因為我們現在在實作 NachOS 本身的FS，所以 formatFlag 為 True，表示我們現在剛創建一個 FS，所以disk上甚麼資訊都還沒有，因此必須 initialize：

    1. an empty directory
    2. a bitmap of free sectors (with almost but not all of the sectors marked as free)

       這個 bitmap 記錄了所有 sector 的使用狀態。

    ```
    void
    Kernel::Initialize()
    {
        ...
    #ifdef FILESYS_STUB
        fileSystem = new FileSystem();
    #else
        fileSystem = new FileSystem(formatFlag);
    #endif // FILESYS_STUB
        ...
    }
    ```
  - FileSystem 的 constructor 有分兩條路：

    1. disk 上還未有任何資訊，因此要 initialize 前面說的兩個 data structure。步驟如下：
       a. 在 freeMap 上把此兩 object 所存在的 disk 位置 mark 起來。在這個步驟還只是在 memory 中改變 freeMap 的 data，尚未寫回去 disk 中。
       b. allocate 這兩個 object 一些 data blocks 以儲存他們的資料。FileHeader 即為 file control block 的 pointer。
       c. 將修改過的 file header 資料寫回 disk。因為 open file 會讀取 disk 中的資料，所以 open 之前必須先寫回去。
       d. 把 freeMap (和 directory) 的資料寫回去 disk。
    2. 已經 initialize 過了，那就只要把 directory 與存 free sectors 的 bitmap 的 file 打開並回傳 (OpenFile object) 即可。

    ```
    FileSystem::FileSystem(bool format)
    {
    ```

```
    DEBUG(dbgFile, "Initializing the file system.");
    if (format)
    {
        PersistentBitmap *freeMap = new PersistentBitmap(NumSectors);
        Directory *directory = new Directory(NumDirEntries);
        FileHeader *mapHdr = new FileHeader;
        FileHeader *dirHdr = new FileHeader;

        DEBUG(dbgFile, "Formatting the file system.");

        freeMap->Mark(FreeMapSector);
        freeMap->Mark(DirectorySector);

        ASSERT(mapHdr->Allocate(freeMap, FreeMapFileSize));
        ASSERT(dirHdr->Allocate(freeMap, DirectoryFileSize));

        DEBUG(dbgFile, "Writing headers back to disk.");
        mapHdr->WriteBack(FreeMapSector);
        dirHdr->WriteBack(DirectorySector);

        freeMapFile = new OpenFile(FreeMapSector);
        directoryFile = new OpenFile(DirectorySector);

        DEBUG(dbgFile, "Writing bitmap and directory back to disk.");
        freeMap->WriteBack(freeMapFile); // flush changes to disk
        directory->WriteBack(directoryFile);

        if (debug->IsEnabled('f'))
        {
            freeMap->Print();
            directory->Print();
        }
        delete freeMap;
        delete directory;
        delete mapHdr;
        delete dirHdr;
    }
    else
    {
        freeMapFile = new OpenFile(FreeMapSector);
        directoryFile = new OpenFile(DirectorySector);
    }
}
```

- 在 filesys.cc 中有兩行 code 定義 FreeMapSector 和 DirectorySector，前者就代表了記錄所有 sector 使用狀況的 bitmap (freeMap) 儲存在 sector 0。

```
#define FreeMapSector 0
#define DirectorySector 1
```

- 而要取得這個 freeMap (儲存 sector 狀態的 bitmap) 的資訊，我們會 new 一個 PersistentBitmap 來暫存 freeMap，透過此來 access freeMap。所以 NachOS 就是使用這種方式來 manage and find free block space。

```
freeMap = new PersistentBitmap(freeMapFile, NumSectors);
```

- 2.

- ANS：

  (1) sector size = 128 bytes

  (2) 128*(32*32) = 128 KB

- disk.h

```
//disk.h
const int SectorSize = 128;   // number of bytes per disk sector
const int SectorsPerTrack  = 32;  // number of sectors per disk track
const int NumTracks = 32;   // number of tracks per disk
const int NumSectors = (SectorsPerTrack * NumTracks);
          // total # of sectors per disk
```

- 3

- Ans:

  (1) 照以下說明的流程

  (2) sector 1

- filesys.cc
  一開始一樣由main.cc開始呼叫Kernel::Initialize()，再進入到FileSystem()。前面有做詳細說明，就不再贅述。

```
FileSystem::FileSystem(bool format)
{
    DEBUG(dbgFile, "Initializing the file system.");
    if (format)
    {
        PersistentBitmap *freeMap = new PersistentBitmap(NumSectors);
        Directory *directory = new Directory(NumDirEntries);
        FileHeader *mapHdr = new FileHeader;
        FileHeader *dirHdr = new FileHeader;

        DEBUG(dbgFile, "Formatting the file system.");

        freeMap->Mark(FreeMapSector);
        freeMap->Mark(DirectorySector);

        ASSERT(mapHdr->Allocate(freeMap, FreeMapFileSize));
        ASSERT(dirHdr->Allocate(freeMap, DirectoryFileSize));

        DEBUG(dbgFile, "Writing headers back to disk.");
        mapHdr->WriteBack(FreeMapSector);
        dirHdr->WriteBack(DirectorySector);

        freeMapFile = new OpenFile(FreeMapSector);
        directoryFile = new OpenFile(DirectorySector);

        DEBUG(dbgFile, "Writing bitmap and directory back to disk.");
        freeMap->WriteBack(freeMapFile); // flush changes to disk
        directory->WriteBack(directoryFile);

        if (debug->IsEnabled('f'))
        {
            freeMap->Print();
            directory->Print();
```

```
        }
        delete freeMap;
        delete directory;
        delete mapHdr;
        delete dirHdr;
    }
    else
    {
        freeMapFile = new OpenFile(FreeMapSector);
        directoryFile = new OpenFile(DirectorySector);
    }
}
```

- Directory.cc
  這裡會在需要的size下，將其用到的table的entry都設為未使用。

```
Directory::Directory(int size)
{
    table = new DirectoryEntry[size];

    memset(table, 0, sizeof(DirectoryEntry) * size); // dummy operation to keep valgrind happy

    tableSize = size;
    for (int i = 0; i < tableSize; i++)
        table[i].inUse = FALSE;
}
```

- OpenFile.cc

  這裡會將file header的內容存入directory sector內。

```
OpenFile::OpenFile(int sector)
{
    hdr = new FileHeader;
    hdr->FetchFrom(sector);
    seekPosition = 0;
}
```

- 在 filesys.cc 中有定義 DirectorySector，代表了記錄所有 sector 使用狀況的 bitmap (freeMap) 儲存在 sector 0。

```
#define FreeMapSector 0
#define DirectorySector 1
```

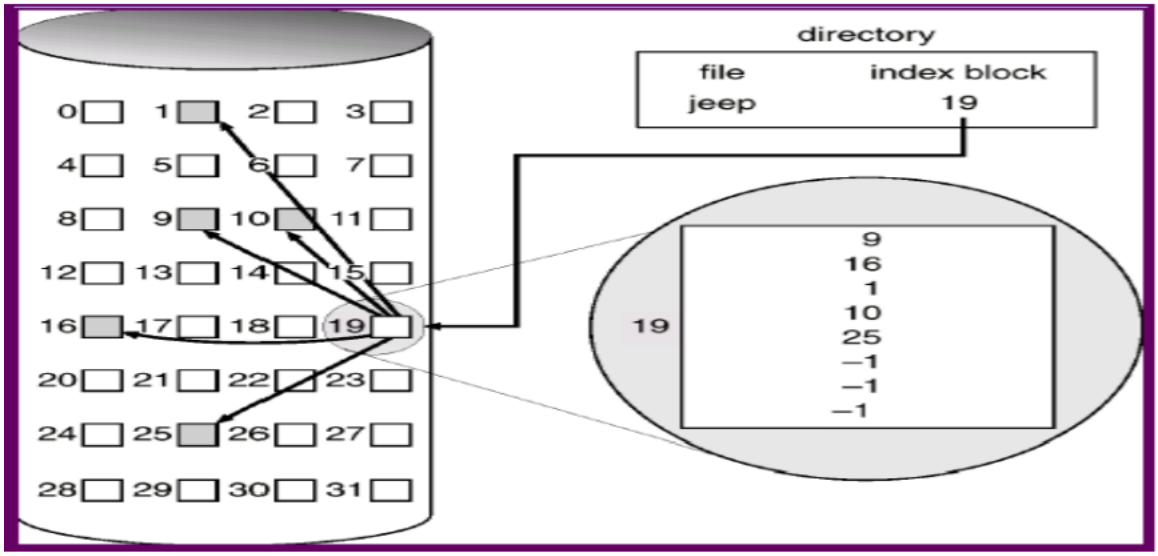
- 4.
  - ANS:

    1. number of bytes in the file也就是這個file內有幾bytes，number of data sectors in the file也就是這個file需要幾個data sectors，還有disk sector numbers for each data block in the file也就是data block存在哪 些disk sector內。

```
class FileHeader {
  private:
```

```
    int numBytes;
    int numSectors;
    int dataSectors[NumDirect];
}
```

2.




- FileSystem::Create()
  剛開始先創建Directory，並從disk讀取directory的內容。如果Find()不等於-1的話，就代表檔案已經存在。相反的，會確認是否有足夠的block給file header，與足夠的space給directory，如果都符合，就會讓file header進行Allocate()，Allocate成功的話就會讓file header、directory、bitmap皆寫回disk。

```
bool FileSystem::Create(char *name, int initialSize)
{
    Directory *directory;
    PersistentBitmap *freeMap;
    FileHeader *hdr;
    int sector;
    bool success;

    DEBUG(dbgFile, "Creating file " << name << " size " << initialSize);

    directory = new Directory(NumDirEntries);
    directory->FetchFrom(directoryFile);

    if (directory->Find(name) != -1)
        success = FALSE; // file is already in directory
    else
    {
        freeMap = new PersistentBitmap(freeMapFile, NumSectors);
        sector = freeMap->FindAndSet(); // find a sector to hold the file header
        if (sector == -1)
            success = FALSE; // no free block for file header
        else if (!directory->Add(name, sector))
            success = FALSE; // no space in directory
        else
        {
            hdr = new FileHeader;
```

```
            if (!hdr->Allocate(freeMap, initialSize))
                success = FALSE; // no space on disk for data
            else
            {
                success = TRUE;
                hdr->WriteBack(sector);
                directory->WriteBack(directoryFile);
                freeMap->WriteBack(freeMapFile);
            }
            delete hdr;
        }
        delete freeMap;
    }
    delete directory;
    return success;
}
```

- FileHeader::Allocate()
  一開始會記錄number of bytes這也等同於file的大小，number of sector則是由(file size/sector size)得到。如果可以使用的容量小於sector的容量，會直接回傳False，也就是沒有空間。接著，會用for迴圈找尋這個sector，並用FindAndSet來找尋哪些是可以使用的。

```
bool FileHeader::Allocate(PersistentBitmap *freeMap, int fileSize)
{
  numBytes = fileSize;
  numSectors = divRoundUp(fileSize, SectorSize);
  if (freeMap->NumClear() < numSectors)
    return FALSE; // not enough space

  for (int i = 0; i < numSectors; i++)
  {
    dataSectors[i] = freeMap->FindAndSet();
    ASSERT(dataSectors[i] >= 0);
  }
  return TRUE;
}
```

- 5.
  從filehdr.h知道SectorSize=128，可得出NumDirect=30，所以可知MaxFileSize的值大約等於4KB。

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

# Implementation

- 1.
  - 將2.中有需要修改的function，在syscall.h的地方就function的type與parameter進行變換。

```
/* syscalls.h */
...
typedef int OpenFileId;
...
int Create(char *name, int size); // FILE_SYS
...
OpenFileId Open(char *name);
...
int Write(char *buffer, int size, OpenFileId id);
...
int Read(char *buffer, int size, OpenFileId id);
...
int Close(OpenFileId id);
...
```

  - 在exception.cc的ExceptionHandler內也要進行修改。

```
case SC_Create:
    val = kernel->machine->ReadRegister(4);
    {
        int size = kernel->machine->ReadRegister(5);
        char *filename = &(kernel->machine->mainMemory[val]);
        status = SysCreate(filename, size);
        kernel->machine->WriteRegister(2, (int)status);
      }
      kernel->machine->WriteRegister(PrevPCReg, kernel->machine->ReadRegister(PCReg));
      kernel->machine->WriteRegister(PCReg, kernel->machine->ReadRegister(PCReg) + 4);
      kernel->machine->WriteRegister(NextPCReg, kernel->machine->ReadRegister(PCReg) + 4);
      return;
      ASSERTNOTREACHED();
      break;
#endif
case SC_Open:
      val = kernel->machine->ReadRegister(4);
      {
        char *filename = &(kernel->machine->mainMemory[val]);
        //cout << filename << endl;
        status = SysOpen(filename);
        kernel->machine->WriteRegister(2, (int)status);
      }
      kernel->machine->WriteRegister(PrevPCReg, kernel->machine->ReadRegister(PCReg));
      kernel->machine->WriteRegister(PCReg, kernel->machine->ReadRegister(PCReg) + 4);
      kernel->machine->WriteRegister(NextPCReg, kernel->machine->ReadRegister(PCReg) + 4);
      return;
      ASSERTNOTREACHED();
      break
case SC_Read:
      val = kernel->machine->ReadRegister(4);
      {
```

```
        int size = kernel->machine->ReadRegister(5);
        OpenFileId id = kernel->machine->ReadRegister(5);
        char *buf = &(kernel->machine->mainMemory[val]);
        //cout << filename << endl;
        status = SysRead(buf, size, id);
        kernel->machine->WriteRegister(2, (int)status);
      }
      kernel->machine->WriteRegister(PrevPCReg, kernel->machine->ReadRegister(PCReg));
      kernel->machine->WriteRegister(PCReg, kernel->machine->ReadRegister(PCReg) + 4);
      kernel->machine->WriteRegister(NextPCReg, kernel->machine->ReadRegister(PCReg) + 4);
      return;
      ASSERTNOTREACHED();
      break
case SC_Write:
      val = kernel->machine->ReadRegister(4);
      {
        int size = kernel->machine->ReadRegister(5);
        OpenFileId id = kernel->machine->ReadRegister(5);
        char *buf = &(kernel->machine->mainMemory[val]);
        status = SysWrite(buf, size, id);
        kernel->machine->WriteRegister(2, (int)status);
      }
      kernel->machine->WriteRegister(PrevPCReg, kernel->machine->ReadRegister(PCReg));
      kernel->machine->WriteRegister(PCReg, kernel->machine->ReadRegister(PCReg) + 4);
      kernel->machine->WriteRegister(NextPCReg, kernel->machine->ReadRegister(PCReg) + 4);
      return;
      ASSERTNOTREACHED();
      break;
case SC_Close:
      val = kernel->machine->ReadRegister(4);
      {
        OpenFileId id = val;
        status = SysClose(id);
        kernel->machine->WriteRegister(2, (int)status);
      }
      kernel->machine->WriteRegister(PrevPCReg, kernel->machine->ReadRegister(PCReg));
      kernel->machine->WriteRegister(PCReg, kernel->machine->ReadRegister(PCReg) + 4);
      kernel->machine->WriteRegister(NextPCReg, kernel->machine->ReadRegister(PCReg) + 4);
      return;
      ASSERTNOTREACHED();
      break;
```

- 在filesys.h裡，要新增下列function以讓filesys.cc使用

```
int CreateAFile(char *name, int initialSize);
OpenFileId OpenAFile(char *name);
int Read(char *buf, int size, OpenFileId id); // Read "size" characters from file
int Write(char *buf, int size, OpenFileId id); // Write "size" characters to file
int Close(OpenFileId id);
```

- 2.
  - int Create(char *name, int size)
    如果成功create一個file，會回傳1，反之為0。

```
int FileSystem::CreateAFile(char *name, int initialSize)
{
    if(Create(name, initialSize)) return 1;
```

```
    else return 0;
}
```

- OpenFileId Open(char *name)

```
OpenFileId FileSystem::OpenAFile(char *name)
{
    nowOpen = Open(name);
    return 1;
}
```

- int Read(char *buf, int size, OpenFileId id)
  將已open的file進行read的動作，將其設為res，並回傳res。

```
int FileSystem::Read(char *buf, int size, OpenFileId id)
{
     int res = nowOpen->Read(buf, size);
     return res;
}
```

- int Write(char *buf, int size, OpenFileId id)
  將已open的file進行read的動作，將其設為res，並回傳res。

```
int FileSystem::Write(char *buf, int size, OpenFileId id)
{
      int res = nowOpen->Write(buf, size);
      return res;
}
```

- int Close(OpenFileId id)
  如果id大於0且有已open的file，就會將open的file給delete，然後回傳1。

```
int FileSystem::Close(OpenFileId id)
{
    if(id > 0 && nowOpen) {
        delete nowOpen;
        nowOpen = NULL;
        return 1;
    }
    return 1;
}
```

# Difficulties & Feedback

## 簡志宇

在還有其他段考與project的情況下，時間有點不太夠。

## 羅稉涵

雖然作業很早就公布，但礙於各種段考與報告的襲來，這份作業是到離期限前幾天才開始運作。雖然File system的觀念在考試已經有先看過，但考完後也很有禮貌的還給了老師，所以還花了點時間複習，且這次沒有助教的Hint，需要自己從頭去trace code這也讓我剛開始耗費了不少時間。