CH14-1-A Very Simple File System

資工三 穆冠蓁 4107056007

1. 將file 轉為實體的位置(physical devices)

2. 提供一個檔案存取介面(File access’s Interface)

* Overall Organization

1. Disk 會被分為許多blocks

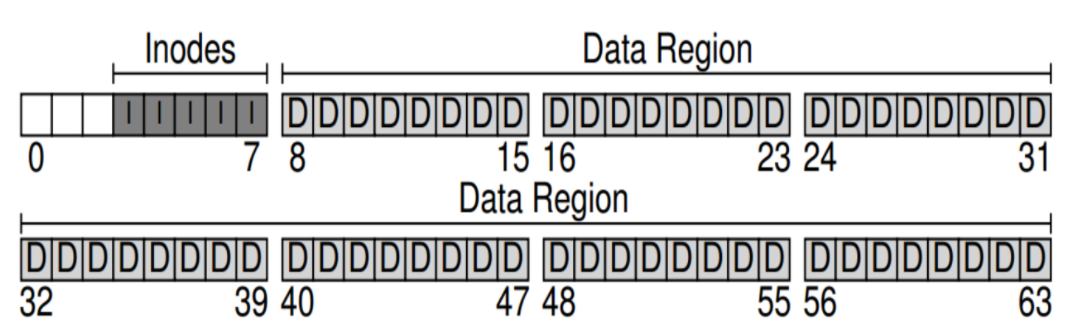
2. Data Region : 存取user data

3. Inode Table :

(1)metadata 用來記錄每個檔案的資訊(每個file都有自己的metadata類型)

(2)inode(index node)就是拿來存放metadata的結構

(3)因此FS需要保留一塊空間來存放inodes(metadata) -> inode table



4. Allocation Structures :

(1) 用來追蹤inodes 或 data blocks是否配置(allocated)

(2) Bitmap : data,inode bitmap

5. Superblock : 記錄整個FS 的info.

(1) 內容:

- bitmap 的大小及位置

- Inode and data blocks 的數量及位置

- inode 的起始位址

(2) A magic number to identify the file system type

* File Organization : The Inode

1. 每個inode 都跟一個數字有關，稱為i-number

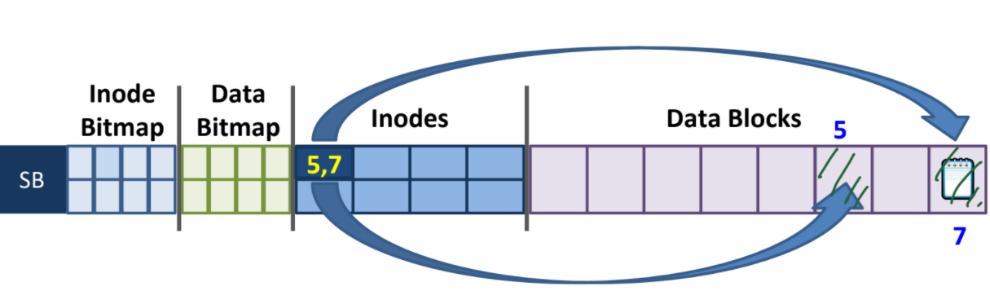
2. Inode 紀錄的內容:

(1) size : 檔案大小

(2) block : 位置(disk pointer) max:15

(3) dir\_acl : access control lists

3. 若是一個檔案占用大於一個block該如何記錄?利用inode內direct pointers



* Directory Organization

1. 在vsfs中，一個目錄(directory)包含一整列的(entry name,inode number)

2. A True Directory Structure

(1) 事實上:目錄是一種特殊形態的檔案

- 包含一個將file names 轉換為 file control blocks(inode) 的表格

- 每一列儲存一個檔名及unique identifier

- unique identifier(inode) : 指向檔案的metadata

(2) Directory Organization :

- 目錄是一種特殊的檔案型態(儲存在data block region，有 inode)

- 不用修改硬碟的結構，就能支援目錄

* Access Paths: Reading and Writing

1. Opening a File : open("/foo/bar", O RDONLY)

(1) 找到bar的inode

- 從root開始讀(read super block)

- 利用inode找到root的data blocks pointer(讀目錄，找foo)

- 找到foo的inode再去找bar

(2) 將bar的inode讀進記憶體

(3) 配置file descriptor

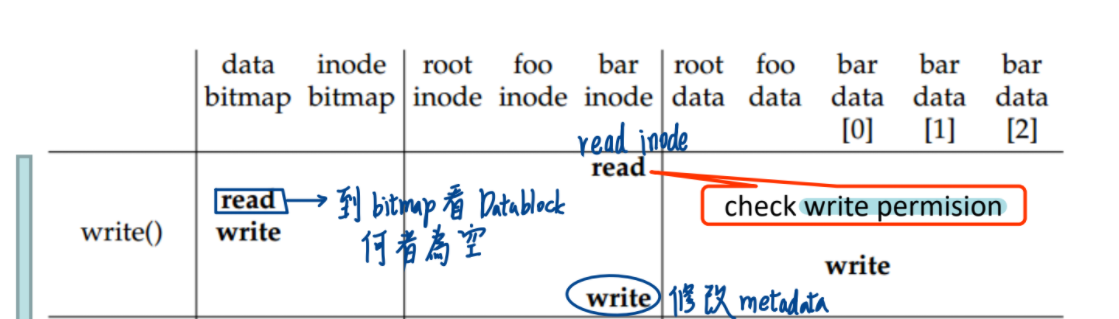
(4) open()產生的I/O量與filepath 長度成正比

(5) 目錄中如果有太多檔案，可能會存在兩個以上的block中

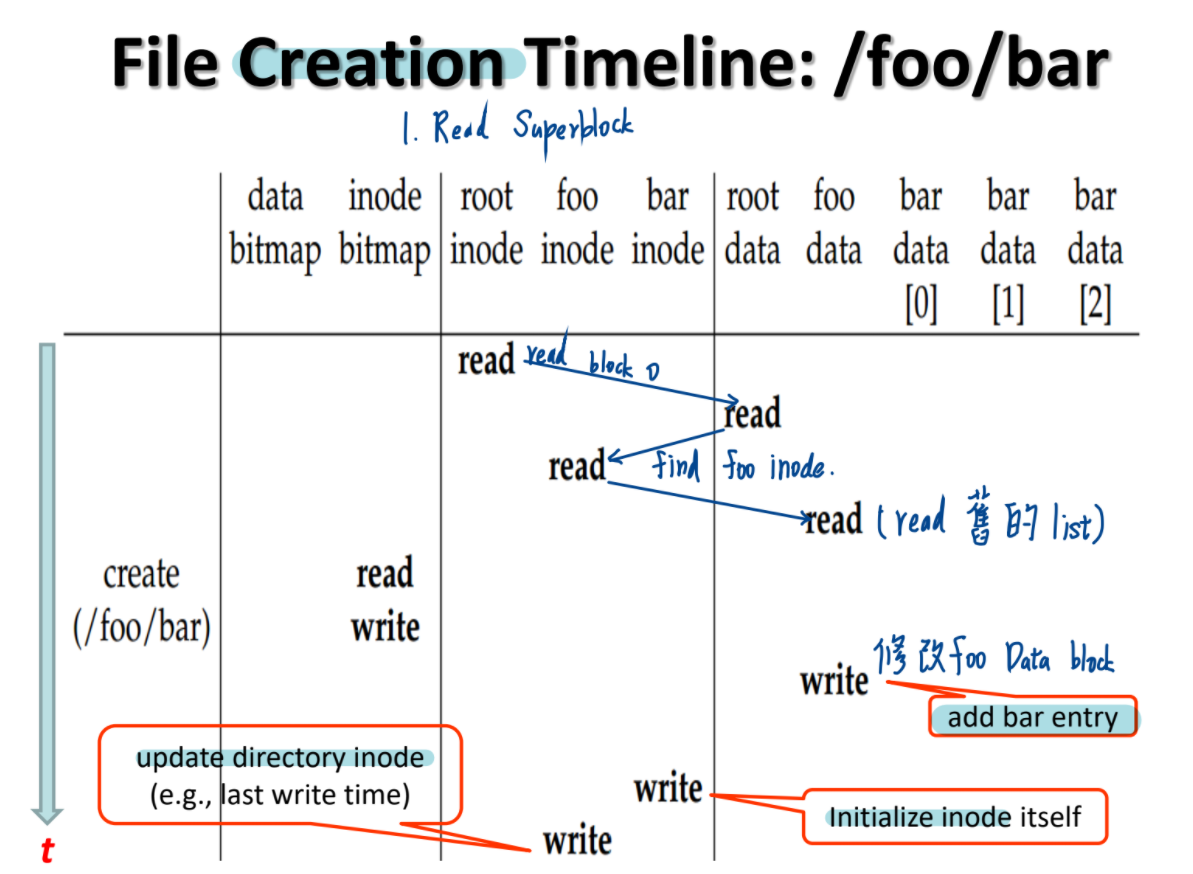
2. Reading : read()

(1) read 透過inode找到data block，也會更新最後存取時間、位移量

3. New Write:



4. Creating a File



* Caching and Buffering

1. 由於要重複存取HD很多次，大部分的檔案系統使用記憶體增加效能

(1) Read caching : 不用直接存取HD到Page Cache存取(包含data blocks 、 inode 、檔案、目錄)

(2) Write buffering : 先寫到mem再找時間write back

- 減少write response time

- 可以批次寫入

- 可以對i/o進行排程

- avoid some writes, e.g., overwrites(對page cahe寫入多次，但對硬碟只需寫入一次)

心得 : 這份ppt的上課內容，在介紹一個真正的檔案處理系統前，先介紹一個簡單的系統。