OS HW3 0711529 陳冠儒

Task1 - Soft Link and Hard Link (30%)

1. Hard link

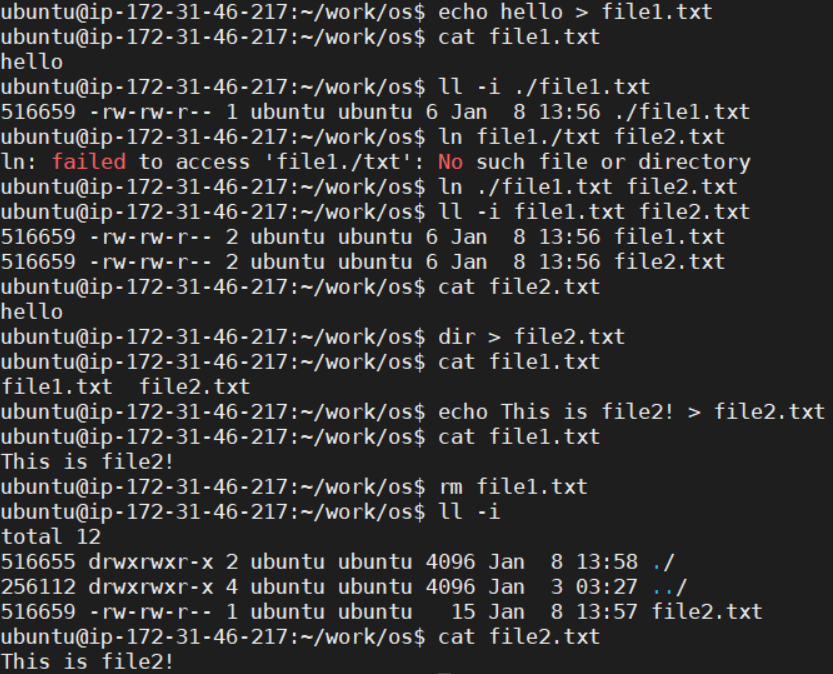
* 理論
* hard link原理
* 每個檔案都會占用一個inode，檔案內容由inode的紀錄來指向。
* 想要讀取該檔案，必須要經過目錄紀錄的檔名來指向到正確的inode號碼才能讀取。

故hard link就是在某個目錄下新增一筆檔名連結到某inode號碼的關聯紀錄。

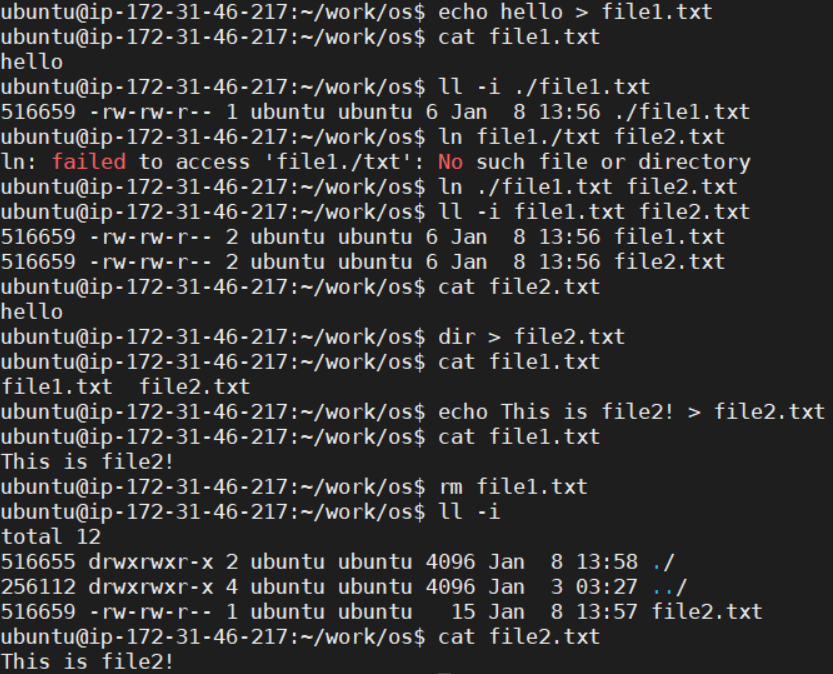
* hard link優點
* 安全：因為多個檔名指向相同內容，所以即使將任一檔名刪除，inode跟block都還是存在
* hard link限制
* 不能跨Filesystem
* 不能link目錄
* hard link空間使用

不會改變，因為只是在某目錄block中多寫入一個資料而已，故不會增加node也不會耗用block數量，除非該目錄block剛好被填滿，才需要新加一個block。

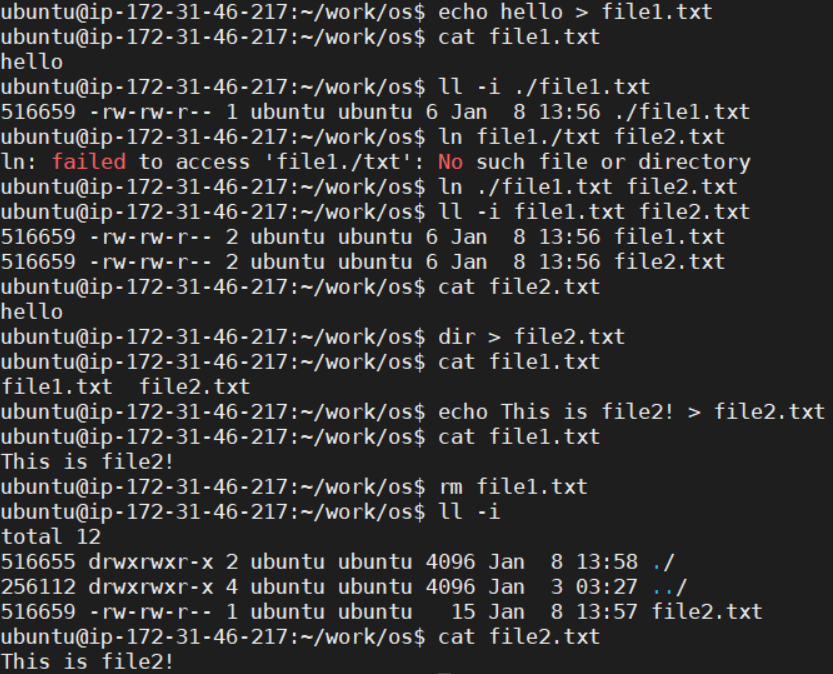
* 實做
* hard link from file1.txt to file2.tx



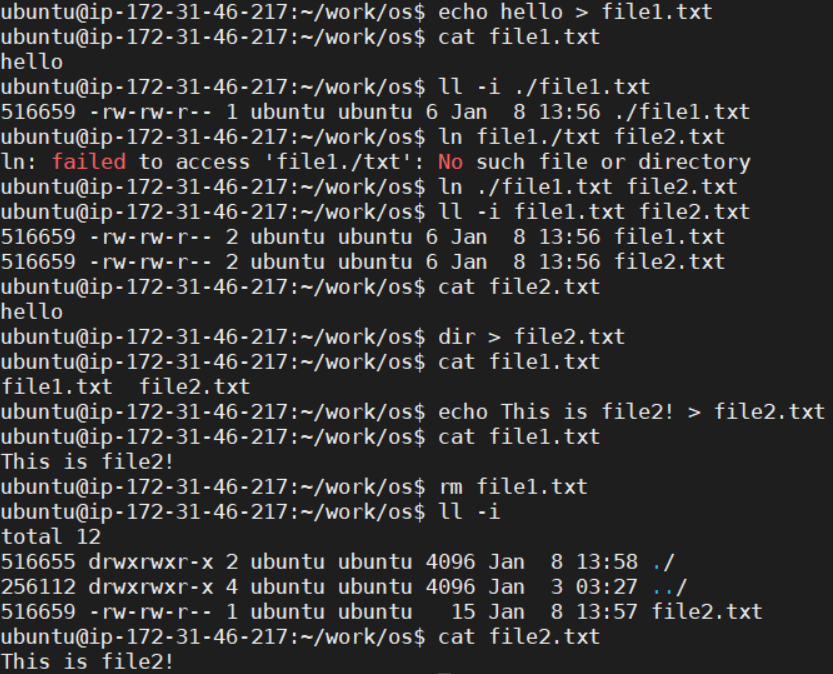
首先先建立一個file1.txt，並且內容為hello。觀察他的資訊可以看到他的inode為516659。



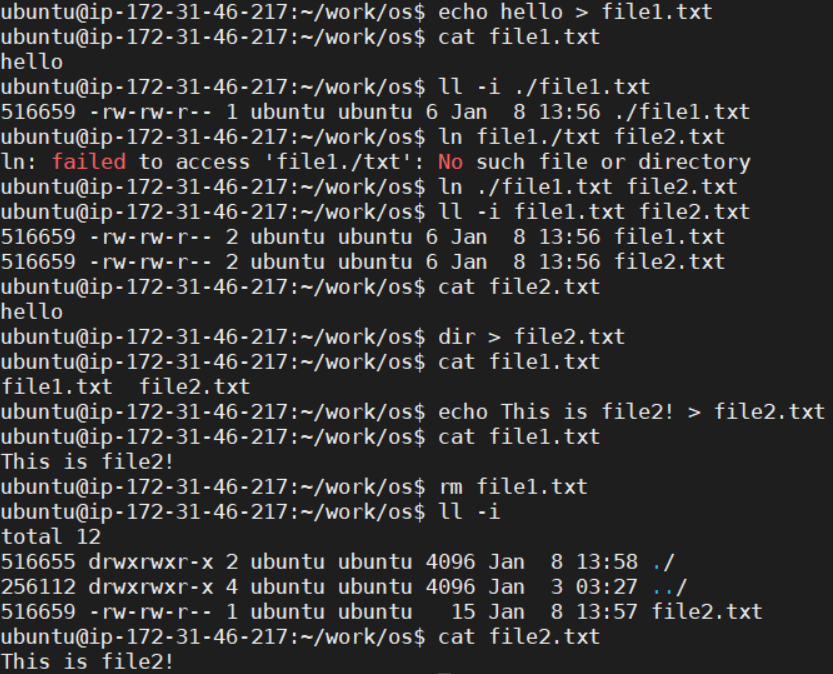
接著建立一個hard link from file1.txt to file2.txt



我們可以發現file2.txt的inode會變得跟file1.txt一樣都是516659，所以兩個的內容會是一樣的，並且可以看到第二欄的link number從原本的1變成了2。

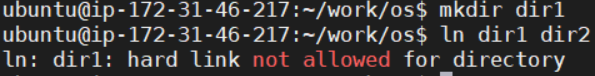


最後我們試著改變file2.txt的內容，可以發現file1.txt的內容亦會跟著改變，因為file1.txt跟file2.txt指向的是同一個inode，而inode會指向一塊block，因此file1.txt和file2.txt是使用相同inode與block。



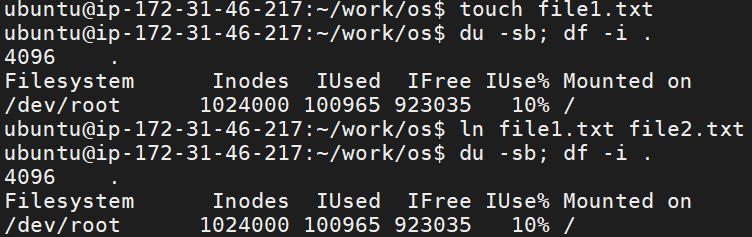
最後將file1.txt刪除後，可以看到file2.txt依然存在並且link number變回了1，可以證實hard link的「安全性」，即使將一個檔名刪除，他的inode和block都還會存在。

* hard link 目錄



可以看出無法hard link目錄，因為如果要hard link目錄，那底下的資料也都要hard link一次，會有很大的複雜度。

* hard link空間使用



使用hard link絕大部分都不會增加額外空間(額外indoe和block)。

1. Soft link

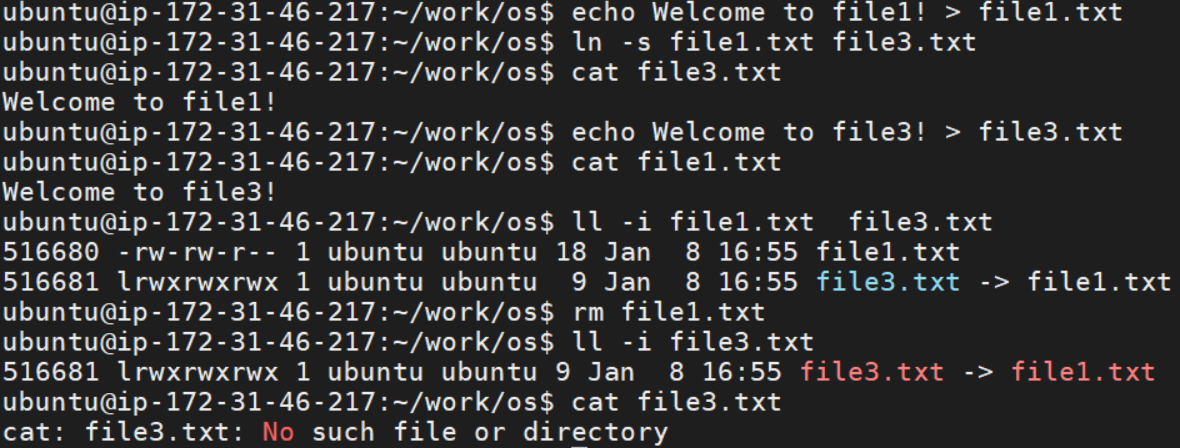
* 理論
* soft link原理

Linux中的soft link又稱為symbolic link，他就是再建立一個獨立的檔案，而這個檔案會讓資料的讀取指向他link的那個檔案的檔名。

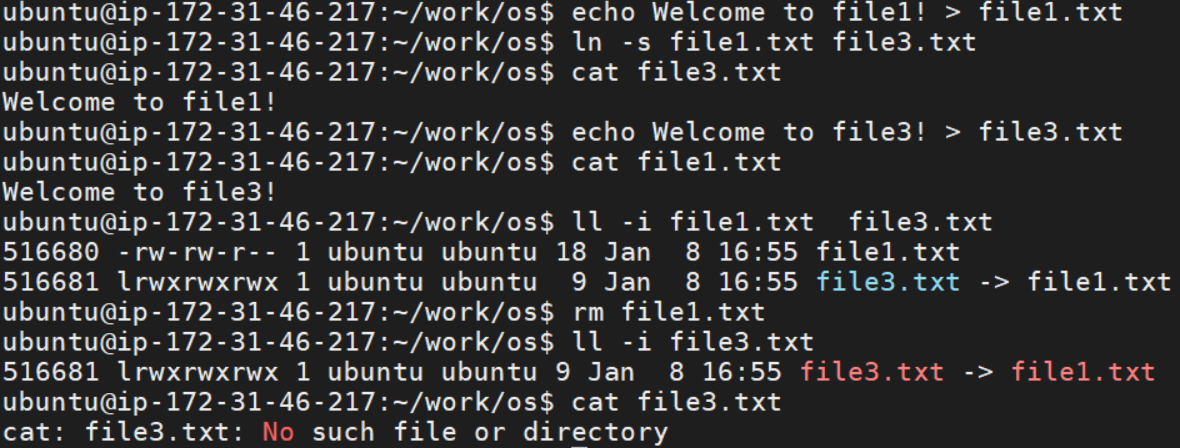
* soft link空間使用

Symbolic link與Windows的捷徑幾乎是相同的，故由Symbolic link所建立的檔案為一個獨立的新檔案，所以會占用掉inode與block。

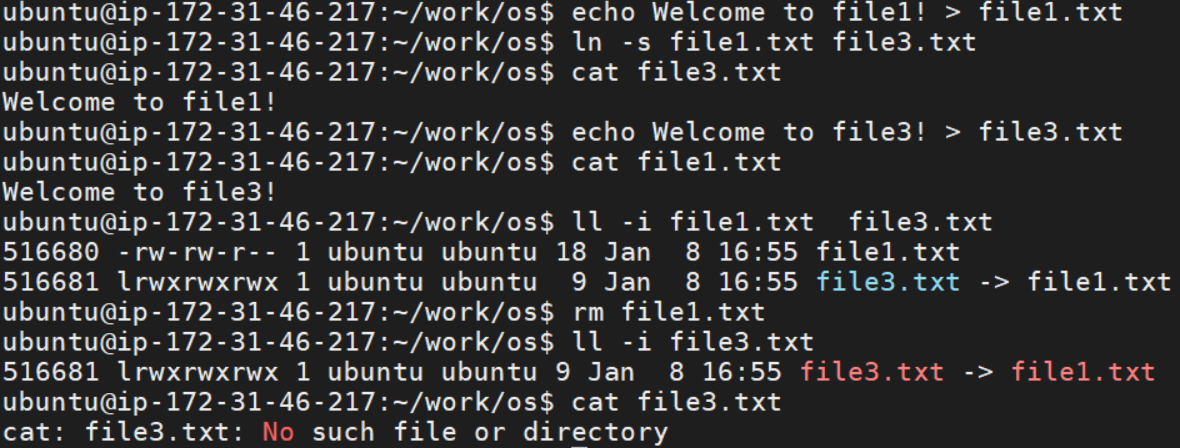
* 實做
* soft link from file1.txt to file3.txt



建立一個file1.txt，內容為「Welcome to file1!」，並建立一個file3.txt的soft link指向file1.txt，可以read/write file1.txt的內容。

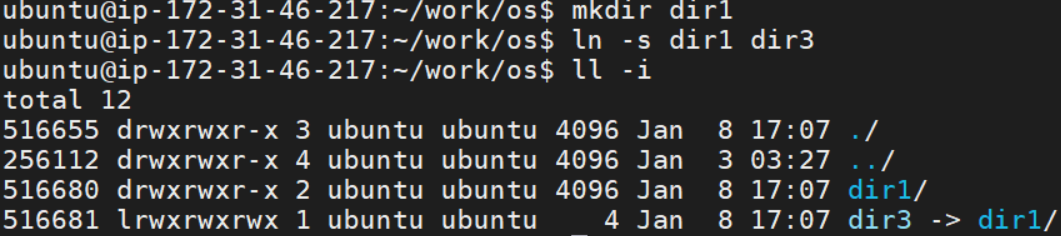


file1.txt和file3.txt的inode是不一樣的，並且用了soft link後，他們的link number都不會增加，還是保持著1。而可以注意到file3.txt的檔案大小為9bytes，是因為「file3.txt」這個檔名有9個英文字，一個英文字佔了一個bytes。



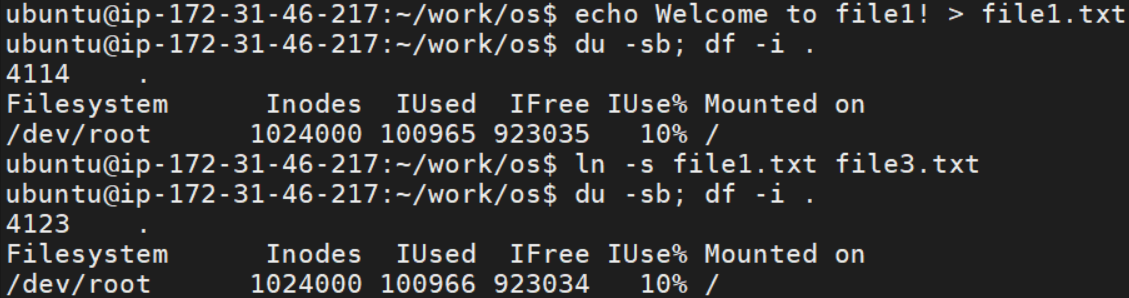
將file1.txt刪除後，file3.txt會失效，因為他指向的file1.txt的檔名不見了。

* soft link 目錄



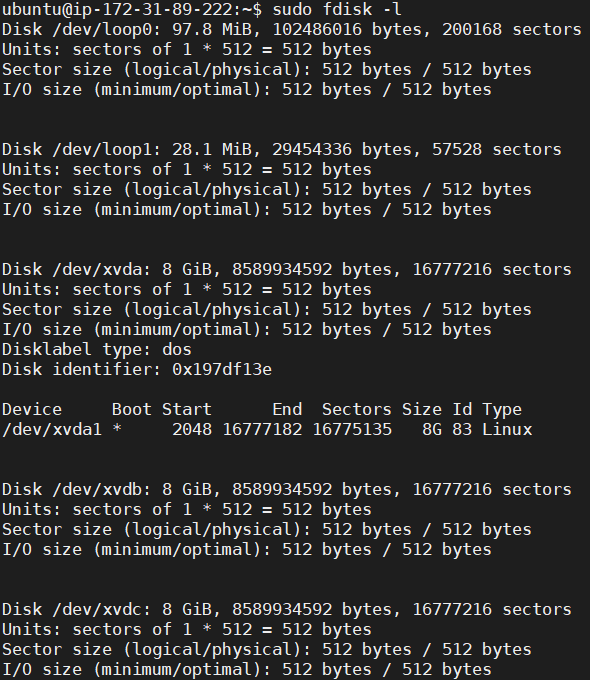
soft link可以link目錄。

* soft link 空間使用



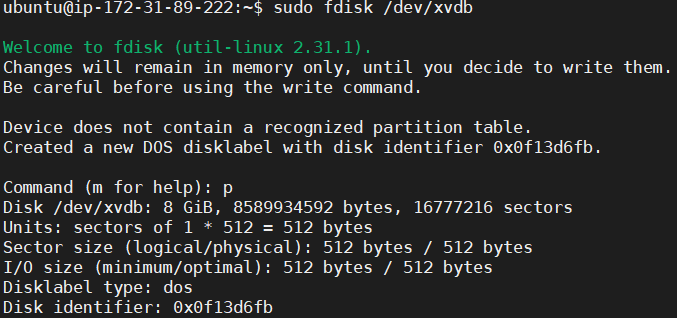
可以看到inode使用量多了1，且空間使用從4114 bytes變成了4123 bytes，增加了9 bytes，即為file3.txt檔名的大小。

Task2 - Creating and mounting file system (30%)

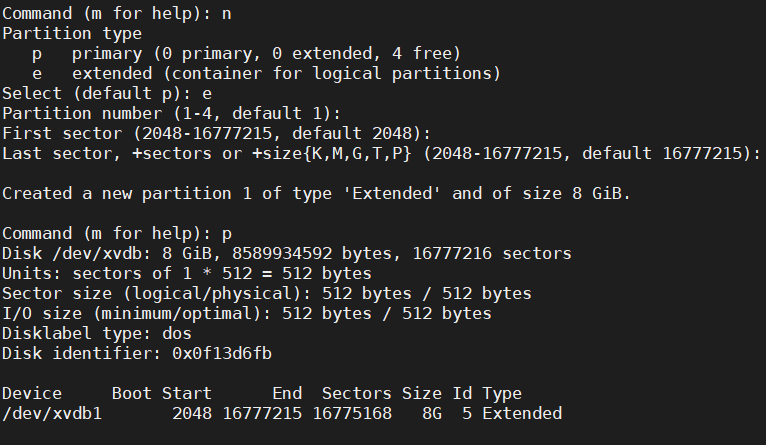


首先，我們用sudo fdisk -l觀察要選哪一個磁碟做分割，可以發現/dev/xvdb跟/dev/xvdc都還沒被分割過，故我們選用這兩個其中的/dev/xvdb來進行分割。

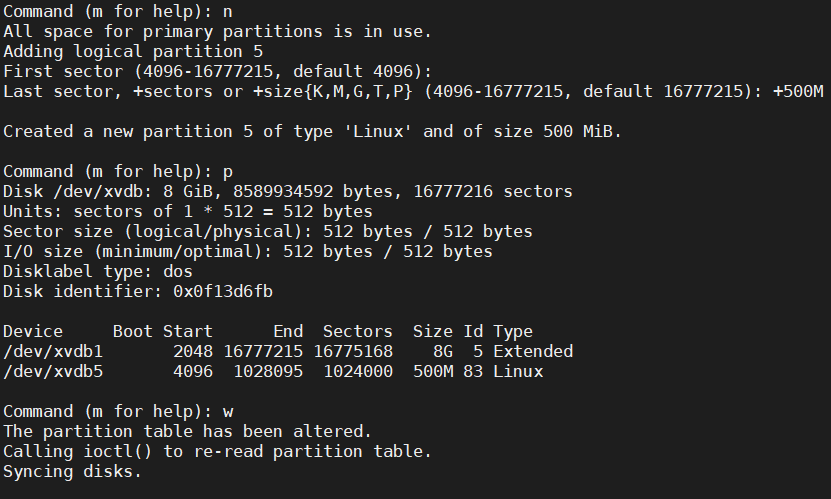
1. Use the fdisk command to add a new 500MB logical partition to your hard drive.



我們先用指令「p」看看這個磁碟的分割情況，上述已經提過他尚未被分割。

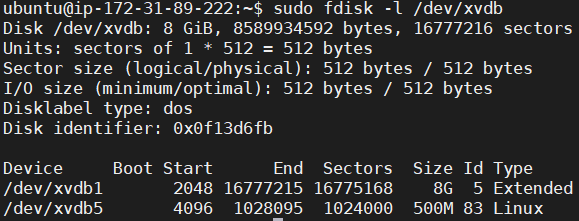


接著在建立logical partition前，我們先用指令「n」建立一個Extended partition，Extended partition可以被切成很多個logical partition。



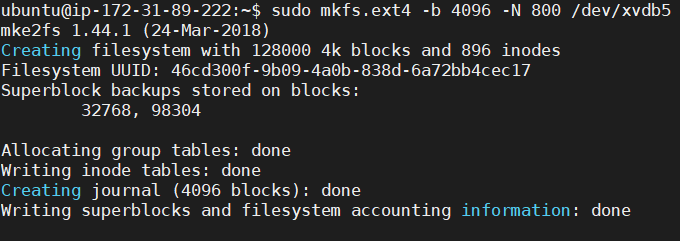
用指令「n」建立一個500M的logical partition，其他不用更改的地方，都用default值，指令「p」確認已建立後，輸入「w」讓他write to disk and exit。

1. Use the fdisk -l command to verify that the new partition has been created.

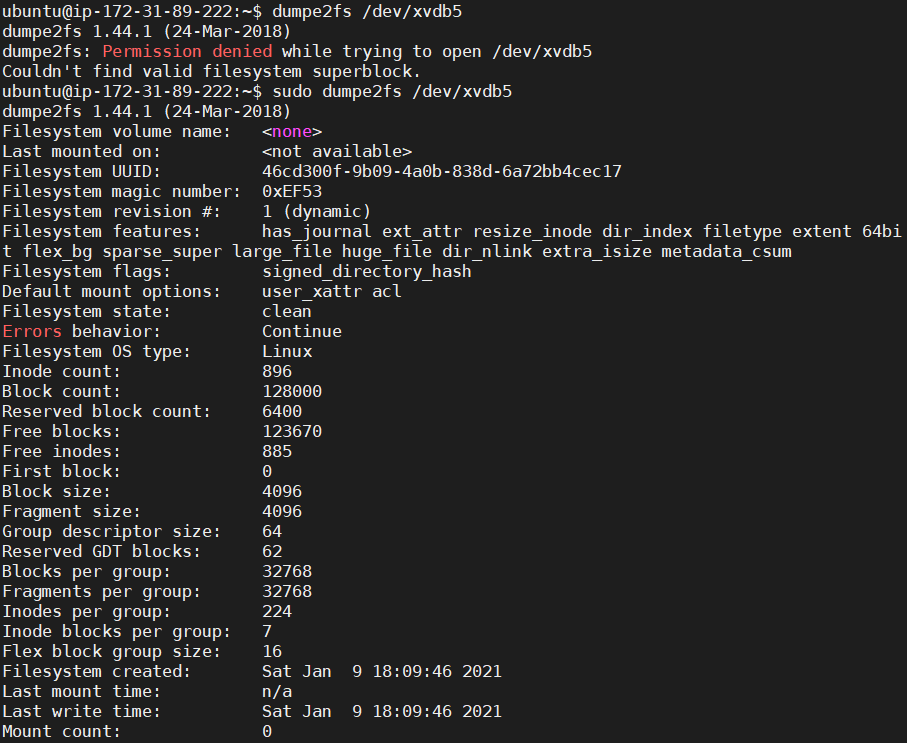


用sudo fdisk -l可以看到剛剛建立好的Extended partition /dev/xvdb1跟logical partition /dev/xvdb5。

1. Format this partition with an ext4 file system that contains 800 inodes and block size is 4096 bytes.

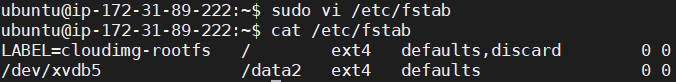


用mkfs.ext4去format partition，參數-b <block size>、-N <iNode number>，但因為分成了四個Block Group，而前11個inode為Special Inode被ext4保留以做些特殊的用途，剩下的要平均分配給內四個Block Group，因為800不能被平均分配，所以才不會是準確剛好800個inode。



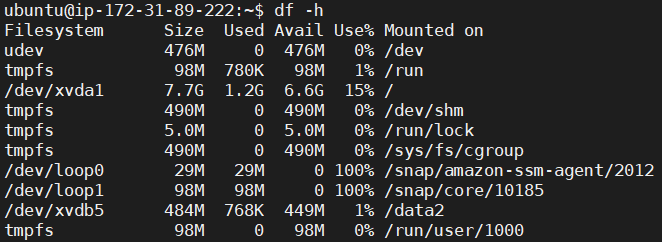
我們可以用dumpe2fs來確認該partition的superblock資訊。

1. Edit /etc/fstab and reboot to mount file system.



用vi修改/etc/fstab，讓他每次開機都能自動掛載，我讓他掛載到/data2這個地方。

1. Use the df command to confirm whether the mount is success.

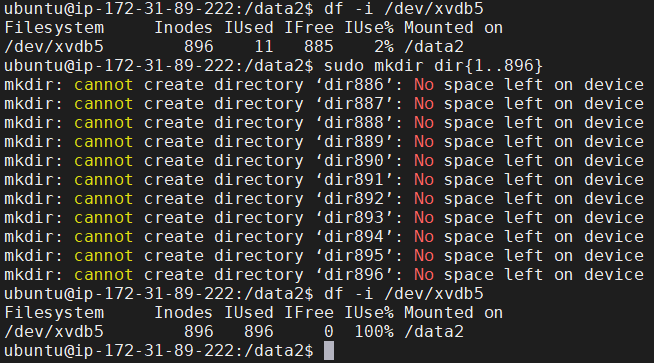


最後利用df去確認他是否有正確被掛載上去，可以看到是正確的！

Task3 – Inode and block (40%)

According to the file system in the Task2, implement and answer the following questions

1. Try to create directories in this file system as many as you can. How many directories can be created in this file system? Why? (Hint: inode)

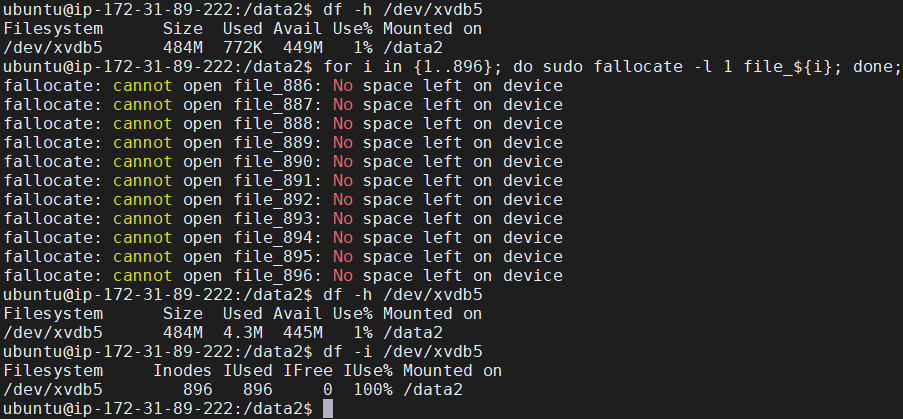


在建directories前，我先用df -i /dev/xvdb5看總共有多少inode數量和已經被用了多少。可以看到總共有896個inode，有11個special inodes被使用，只剩下了885個未被使用的。

接著用sudo mkdir dir{1...896}建立大量的directories，可以看到最後總共建了885個，跟前述未被使用過的inode數量是一樣的。

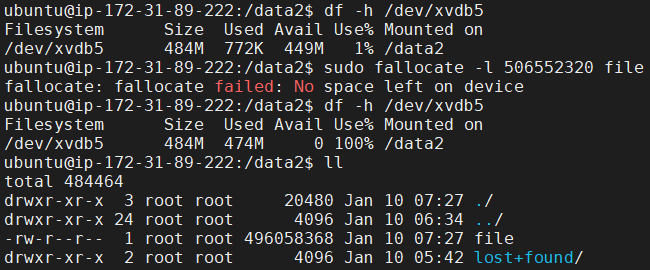
最後再用df來檢查看看是不是所有inode都被使用了，此時IFree為0，代表都被用掉了。

1. Try to create 1-byte files in this file system as many as you can. How many 1-byte files can be created in this file system? Can it completely use all space in this file system? (Hint: block size is 4096 bytes)



我們先用df -h /dev/xvdb5來查看總共有多少可以使用的空間，目前我們還有449M可以使用，再用for i in {1..896}; do sudo fallocate -l 1 file\_${i}; done; 去建立896個1-byte file，會發現最終只能建出885個，跟上面那題的結果是一樣的，因為一個檔案會占用一個inode，而當inode被用完後就無法再建立檔案了，所以即使每個block裡面還有相當大的空間，但依然無法建立，這就是所謂的internal fragmentation。

1. Try to create a file which size as large as you can. What is the maximum file size? Can it completely use all space in this file system?



最大可以建立一個大小為473M (496058368 bytes)的檔案，將所有可以使用的空間都使用完了，但是發現並不能使用掉所有file system的空間，總空間有484M，總使用量為474M，有10M是被特殊保留起來的。