109590004 呂育瑋 作業系統 HW#4

系統環境：VMware, Linux, Ubuntu 18.04

* Problem 11.13

1. file1.txt和file2.txt的inode是一樣的，並且有相同的內容。

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

1. 修改file2.txt也會讓file1.txt的內容更新，刪除file1.txt後file2.txt仍能正常使用，並使用strace查看移除file2.txt所呼叫到的system call。

一張含有 文字, 螢幕擷取畫面, 功能表, 字型 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

1. 利用弱連結的檔案結構，連結的兩個檔案inode並不會相同，修改file4.txt也會讓file3.txt的內容一併更新，刪除file3.txt會讓file4.txt的內容遺失，由於弱連結指向的檔案被刪除，因此它變為唯獨且內容遺失。

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

* Problem 12.16

編寫一個能根據給予的請求序列，在不同的磁碟排程演算法（FCFS、SSTF、SCAN、C-SCAN）中，磁碟頭需要移動的距離（又稱耗時），在實現後能發現，SSTF和SCAN在大多數情況下的效率表現較佳。

FCFS：依照請求序列的順序，header會依序移動至指定的cylinder，並累加移動距離，得到以FCFS演算法處理的header移動路徑長度。

SSTF：先標記請求序列的標頭索引header，將請求序列依照大小排序，並定義2個距離header最近的索引（forward、backward），取得兩個方向需要的移動距離，header以計算後最短移動距離的方向移動，累加header移動的距離，header如果往forward跑，forward - 1；header如果往backward跑，backward + 1，直到所有的cylinder請求都被執行過，得到以SSTF演算法處理的header移動路徑長度，是四種方法中在絕大部分情況下最有效率的排程演算法。

SCAN：先標記請求序列的標頭索引header，將請求序列依照大小排序，再將排序後的序列，以header作為序列分割的索引，將header包含的部分序列作為新序列，再將剩餘部份的序列翻轉後串接在新序列後面，得到SCAN演算法中header會依序執行的SCAN序列，舉例：

假設header在 2150 cylinder：

已排序的序列：356 544 1212 1523 1618 2069 2150 2296 2800 3681 4965

SCAN序列：2150 2296 2800 3681 4965 2069 1618 1523 1212 544 356  
header再依序以新序列陸續移動，並累加header的移動距離，得到以SCAN演算法處理的header移動路徑長度。

C-SCAN：與SCAN前面的做法一樣，不同的是在做序列分割時，不包含剩餘部分的序列直接串接在新序列後面，得到在C-SCAN演算法中header會依序執行的C-SCAN序列，舉例：

假設header在 2150 cylinder：

已排序的序列：356 544 1212 1523 1618 2069 2150 2296 2800 3681 4965

C-SCAN序列：2150 2296 2800 3681 4965 356 544 1212 1523 1618 2069

header再依序以新序列陸續移動，並累加header的移動距離，得到以

C-SCAN演算法處理的header移動路徑長度。

程式碼：Source code\Chap12.16\disk\_scheduling.c

