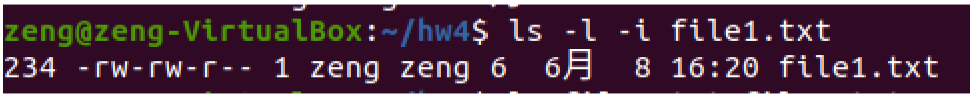
11.13*: This exercise examines the relationship between files and inodes on a UNIX or Linux system. You can complete this exercise on the Linux virtual machine that is provided with this text.

(1) In the source code available with this text, open file1.txt and examine its contents. Next, obtain the inode number of this file with the command:

ls –li file1.txt

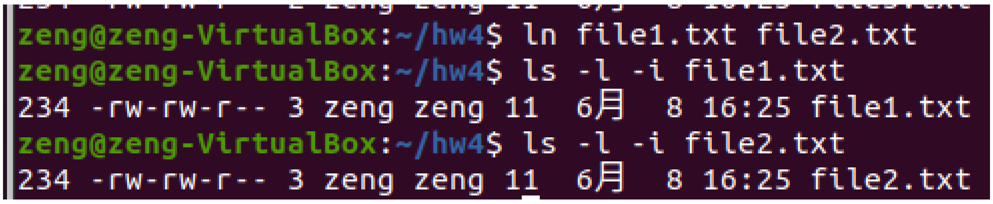


```
zeng@zeng-VirtualBox:~/hw4$ ls -l -i file1.txt
234 -rw-rw-r-- 1 zeng zeng 6  6月  8 16:20 file1.txt
```

Create a hard link between file1.txt and file2.txt:

ln file1.txt file2.txt

What are the inode values of file1.txt and file2.txt?

A: 

```
zeng@zeng-VirtualBox:~/hw4$ ln file1.txt file2.txt
zeng@zeng-VirtualBox:~/hw4$ ls -l -i file1.txt
234 -rw-rw-r-- 3 zeng zeng 11  6月  8 16:25 file1.txt
zeng@zeng-VirtualBox:~/hw4$ ls -l -i file2.txt
234 -rw-rw-r-- 3 zeng zeng 11  6月  8 16:25 file2.txt
```

inode values 都為 234。

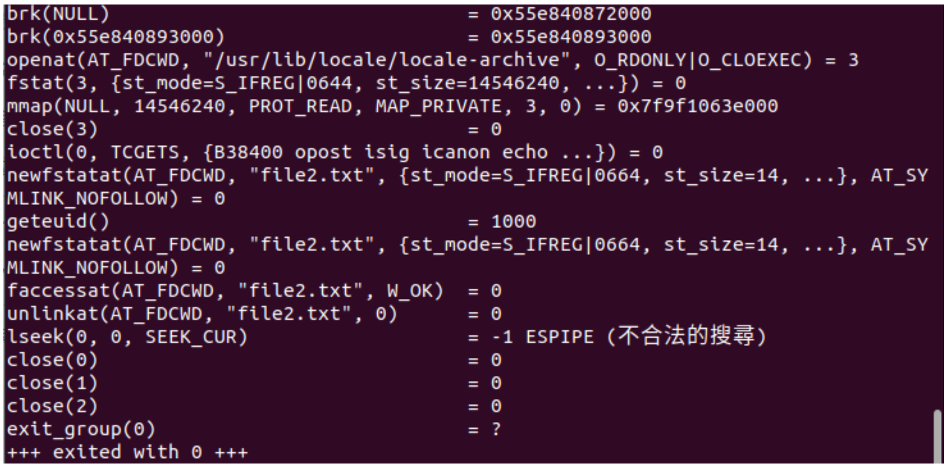
Are they the same or different? Do the two files have the same or different contents?

都一樣，inode values 與 contents 都一樣。

(2) Next, edit file2.txt and change its contents. Examine the contents of file1.txt. Are the contents of file1.txt and file2.txt the same or different? Next, remove file1.txt. Does file2.txt still exist as well? Check what system call is used for removing file2.txt by the following command: *strace rm file2.txt*.

A:改變 file2 後 ，file1 與 file2 內容依然一樣。

刪除 file1,file2 依然存在。

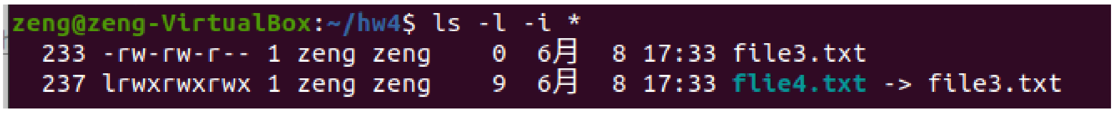


```
brk(NULL)                               = 0x55e840872000
brk(0x55e840893000)                     = 0x55e840893000
openat(AT_FDCWD, "/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=14546240, ...}) = 0
mmap(NULL, 14546240, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f9f1063e000
close(3)                                = 0
ioctl(0, TCGETS, {B38400 opost isig icanon echo ...}) = 0
newfstatat(AT_FDCWD, "file2.txt", {st_mode=S_IFREG|0664, st_size=14, ...}, AT_SY
MLINK_NOFOLLOW) = 0
geteuid()                               = 1000
newfstatat(AT_FDCWD, "file2.txt", {st_mode=S_IFREG|0664, st_size=14, ...}, AT_SY
MLINK_NOFOLLOW) = 0
faccessat(AT_FDCWD, "file2.txt", W_OK)  = 0
unlinkat(AT_FDCWD, "file2.txt", 0)      = 0
lseek(0, 0, SEEK_CUR)                   = -1 ESPIPE (不合法的搜尋)
close(0)                                = 0
close(1)                                = 0
close(2)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

使用 Unlinkat()刪除 file2.txt

(3) Create a soft link to file3.txt by the following command: *ln –s file3.txt file4.txt* Are the inode numbers of file3.txt and file4.txt the same, or is each unique? Next, edit the contents of file4.txt. Have the contents of file3.txt been altered as well? Last, delete file3.txt. Explain what happens when you attempt to edit file4.txt.

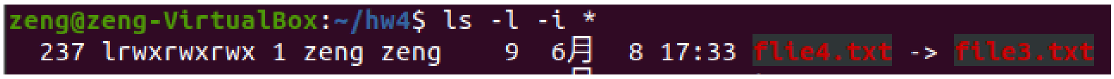
A: *file3.txt file4.txt* 的 inode 不一樣



```
zeng@zeng-VirtualBox:~/hw4$ ls -l -i *
  233 -rw-rw-r-- 1 zeng zeng    0  6月  8 17:33 file3.txt
  237 lrwxrwxrwx 1 zeng zeng    9  6月  8 17:33 flie4.txt -> file3.txt
```

因為是軟連結的關係，修改 file4.txt 的同時，file3.txt.也會跟著改

但是軟連結的原檔案若被刪除，那麼相關的軟連接的檔案就會變成死連結，無法修改甚至開啟。



```
zeng@zeng-VirtualBox:~/hw4$ ls -l -i *
  237 lrwxrwxrwx 1 zeng zeng    9  6月  8 17:33 flie4.txt -> file3.txt
```

12.16*: Write a program that implements the following disk-scheduling algorithms:

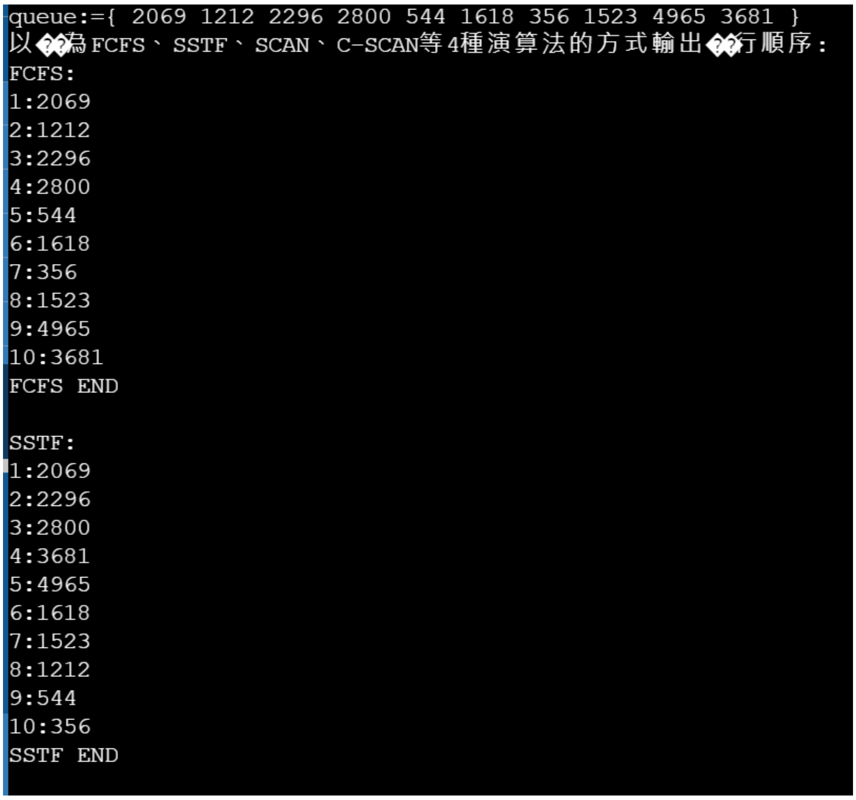
(a) FCFS

(b) SSTF

(c) SCAN

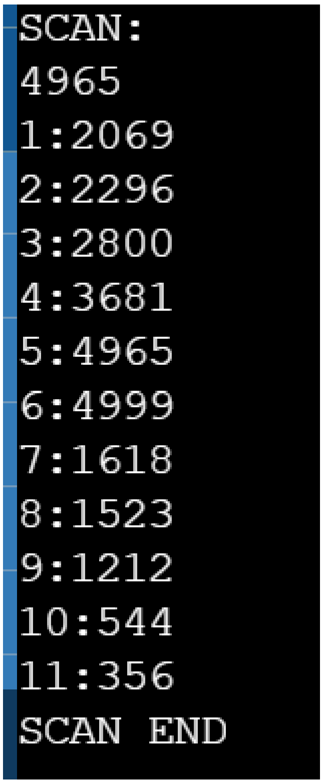
(d) C-SCAN

給定一個 queue array 以及 CYLINDERS 設定為 0-4999，然後輸出 4 種演算法的執行順序，以下為截圖畫面(使用 onlineGDB 編譯):

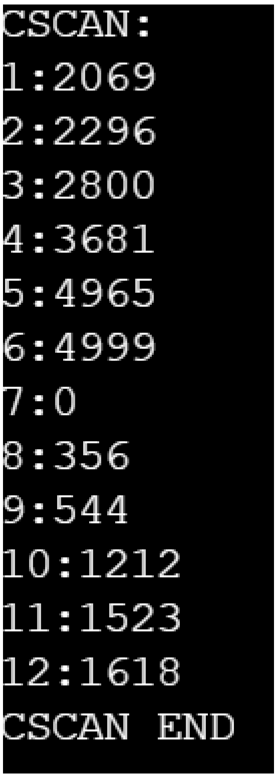


```
queue:={ 2069 1212 2296 2800 544 1618 356 1523 4965 3681 }
以��為FCFS、SSTF、SCAN、C-SCAN等4種演算法的方式輸出��行順序:
FCFS:
1:2069
2:1212
3:2296
4:2800
5:544
6:1618
7:356
8:1523
9:4965
10:3681
FCFS END

SSTF:
1:2069
2:2296
3:2800
4:3681
5:4965
6:1618
7:1523
8:1212
9:544
10:356
SSTF END
```



```
SCAN:
4965
1:2069
2:2296
3:2800
4:3681
5:4965
6:4999
7:1618
8:1523
9:1212
10:544
11:356
SCAN END
```



```
CSCAN:
1:2069
2:2296
3:2800
4:3681
5:4965
6:4999
7:0
8:356
9:544
10:1212
11:1523
12:1618
CSCAN END
```