Operating System
10610 CS342302
Prof. Pai H. Chou
TA:

Hsu PO-CHUN
Student ID: s105062803
Submitted: Dec. 02, 2017
Due Date: Dec. 03, 2017

Assignment 10

1 Problem 1.1

[20 points] 11.2 Contrast the performance of the three techniques for allocating disk blocks (contiguous, linked, and indexed) for both sequential and random file access.

1.1 Answer

- Contiguous Allocation: The files occupy contiguous blocks. The performance is the best in most cases, but users have to know size in advance.
  Sequential access: The accesses are as same way as its allocation by using start and length field in directory structures and it just need to traverse the contiguous disk blocks, hence, it works very well and providing faster sequential access.
  Random access: All it needs to do is add an offset entry to the first block location of the file. Hence, the efforts for calculating random disk location is minimized in this technique and it also works very well.

- Linked Allocation: Each file is a linked list of blocks. So, blocks may be scattered anywhere on the disk.
  Sequential access: Being different from contiguous allocation technique, it simply follows the links form one block to the next, so the performance is satisfactory.
  Random access: The performance is poor because it needs to perform several sequential searches through the links until arriving the intended seek points of the file.

- Indexed Allocation: Each file has its own index as the table of pointers to its data blocks.
  Sequential access: It takes minimum time span to locate a file block because it just needs to sequentially accessing each index.
  Random access: It also takes minimum time span to locate a file block because it costs few to determine the index related to the disk block containing the seek point to the file.

Problem 1.2

[10 points] 11.8 Consider a file system that uses inodes to represent files. Disk blocks are 8 KB in size, and a pointer to a disk block requires 4 bytes. This file system has 12 direct disk blocks, as well as single, double, and triple indirect disk blocks. What is the maximum size of a file that can be stored in this file system?

1.2 Answer

Given the disk blocks are 8KB.
There are 12 entries of direct blocks, a single indirect block, a double indirect block and a triple indirect disk block.

There are $\frac{8KB=8*1024bytes}{4bytes}$ = 2048 entries for each indirect disk blocks.
The direct size: 12 * 8 KB = 96KB.
The single indirect size: 2048 * 8KB = 16MB.
The double indirect size: 2048 * 2048 * 8KB= 32GB.
The triple indirect size: 2048 * 2048 *2048 * 8KB = 64TB.

The total are (96KB + 16MB + 32GB+ 64TB) and it is near to 64.03TB#

2 Problem 2.1

What are the inode values of file1.txt and file2.txt? Are they the same or different? Do the two files have the same or different contents?

2.1 Answer

The inode value of file1.txt is **659531** and The inode value of file2.txt is also **659531**. They have the same inode value and the same content as well.

Problem 2.2

Next, edit file2.txt and change its contents. After you have done so, examine the contents of file1.txt. Are the contents of file1.txt and file2.txt the same or different?

2.2 Answer

Modifying file2.txt change the contents of file1.txt. And the contents of file1.txt and file2.txt are the same because they are linked to the same inode.

Problem 2.3

Next, enter the following command which removes file1.txt:

rm file1.txt

Does file2.txt still exist as well?

2.3 Answer

Yes, file2.txt still exists after removing file1.txt because the reference count of the inode is 2.

Problem 2.4

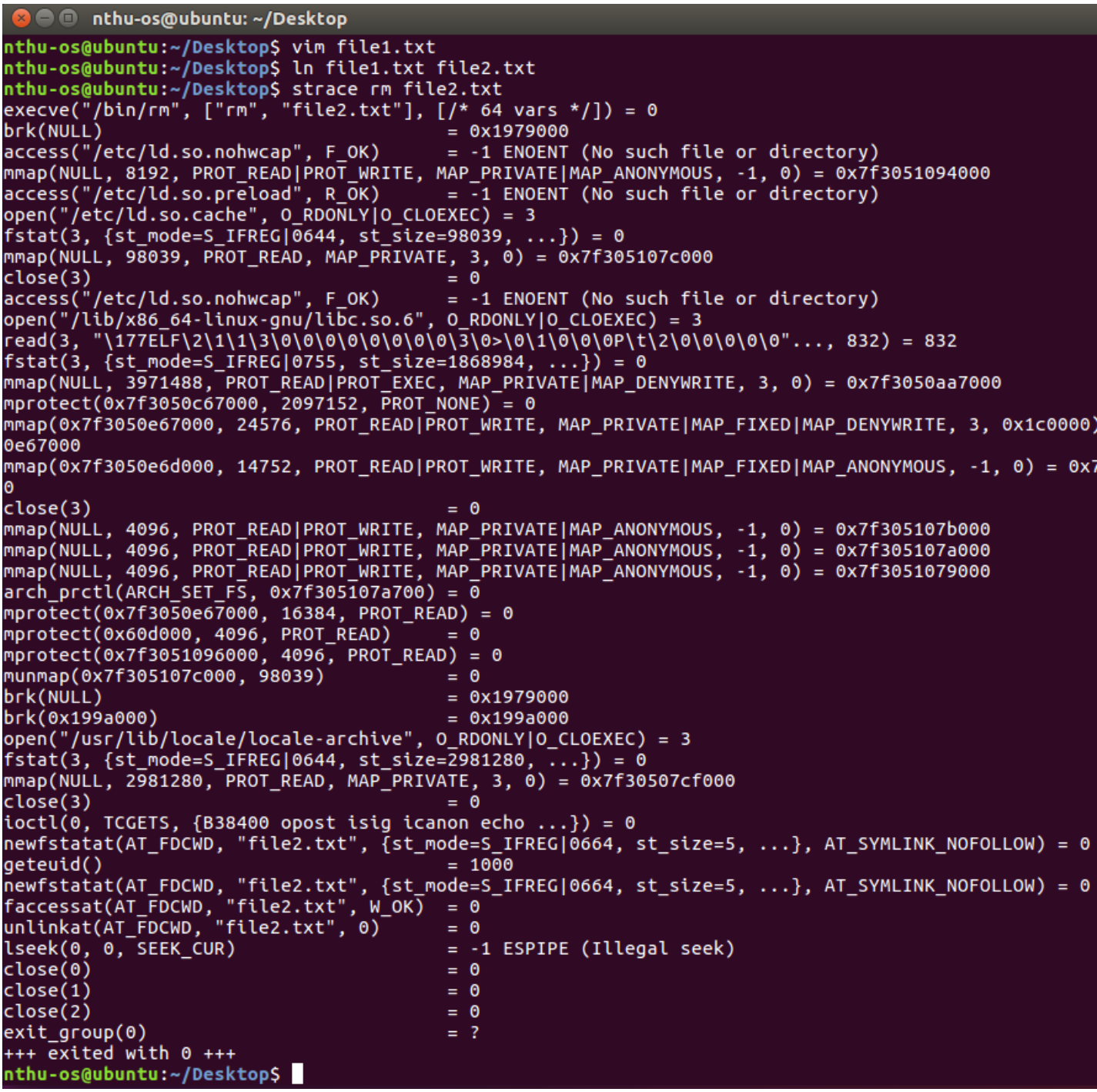
Now examine the man pages for both the rm and unlink commands. Afterwards, remove file2.txt by entering the command:

strace rm file2.txt

The strace command traces the execution of system calls as the command rm file2.txt is run. What system call is used for removing file2.txt?

2.4 Answer

The used system calls are as following:
execve brk access mmap open fstat close read mprotect arch_prctl munmap ioctl newfstatat geteuid faccessat unlinkat lseek exit_group



Problem 2.5

A soft link (or symbolic link) creates a new file that points to the name of the file it is linking to. In the source code available with this text, create a soft link to file3.txt by entering the following command:

ln -s file3.txt file4.txt

After you have done so, obtain the inode numbers of file3.txt and file4.txt using the command

ls -li file*.txt

Are the inodes 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. After you have done so, explain what happens when you attempt to edit file4.txt.

2.5 Answer

The inode value of file3.txt and file4.txt are different as it uses a soft link from the source of file3.txt to the target of file4.txt.
And Both file3.txt and file4.txt have the same contents even if only modifying the contents of one of them.
After removing file3.txt, attempting to edit file4.txt makes it to edit a new file.And save the contents after editing of file4.txt will restore file3.txt back.