**Student name:** Federico Watkins

**CSIS 3810:** Operating Systems

**Chapter:** 12-13 System Implementation & I/O Systems

**Assignment:** 9

|  |  |  |
| --- | --- | --- |
| 1. 🡪 12.21 | 1. 🡪 13.12 | 1. 🡪 13.15 |

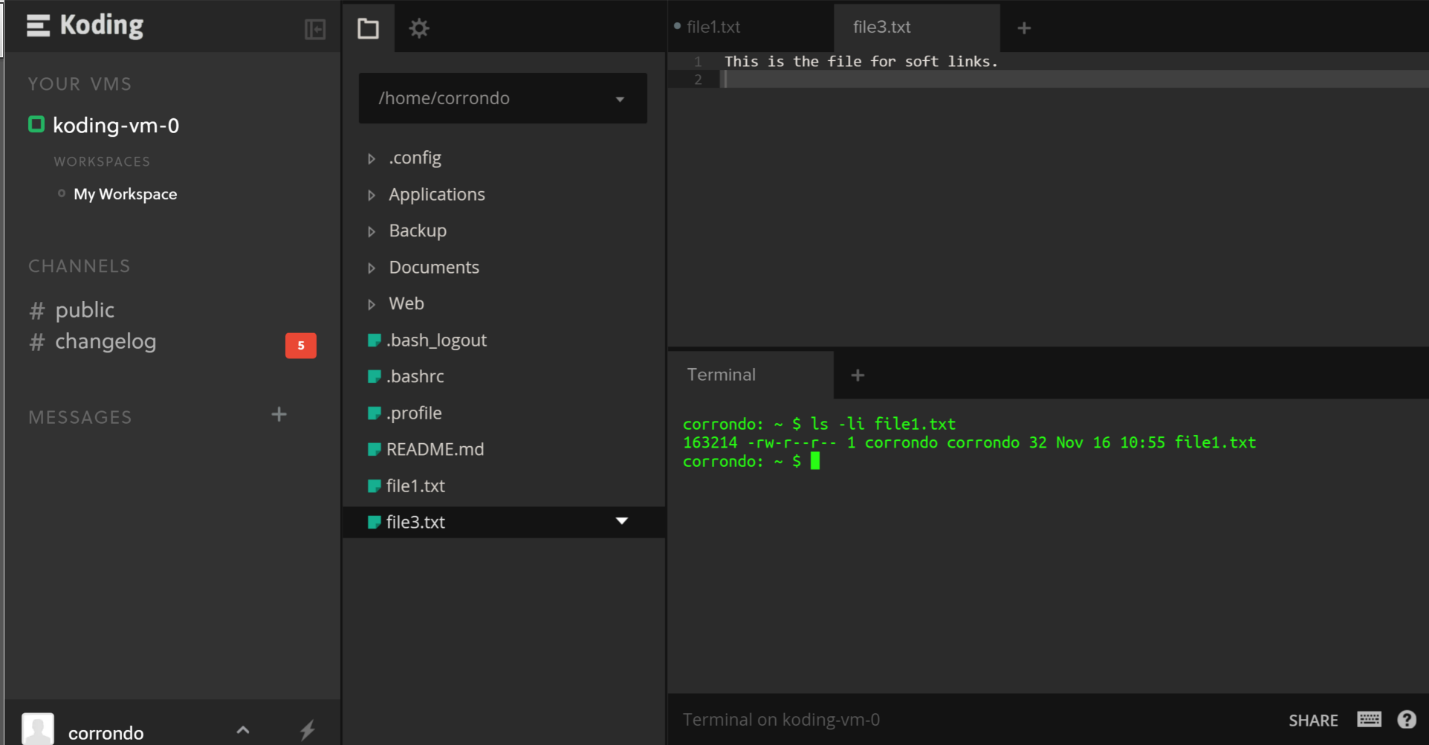
1. In the source code available with this text, open file1.txt and examine its contents. Next, obtain the inode number of this file with the command

ls -li file1.txt

This will produce output similar to the following:

**16980** -rw-r--r-- 2 os os 22 Sep 14 16:13 file1.txt

where the inode number is boldfaced. (The inode number of file1.txt is likely to be different on your system.)



* Inode = **163214**

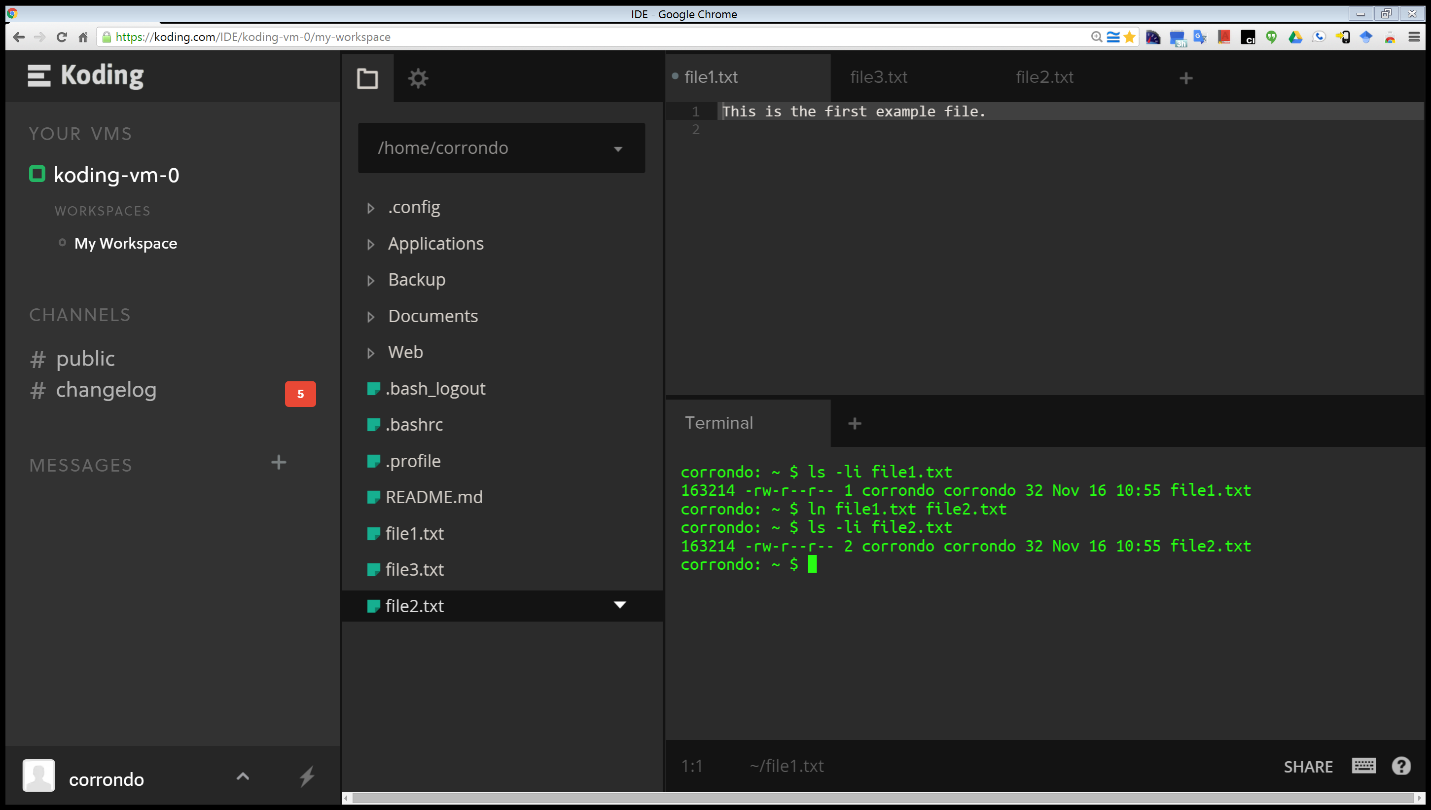
The UNIX ln command creates a link between a source and target file.

This command works as follows:

ln [-s] <source file> <target file>

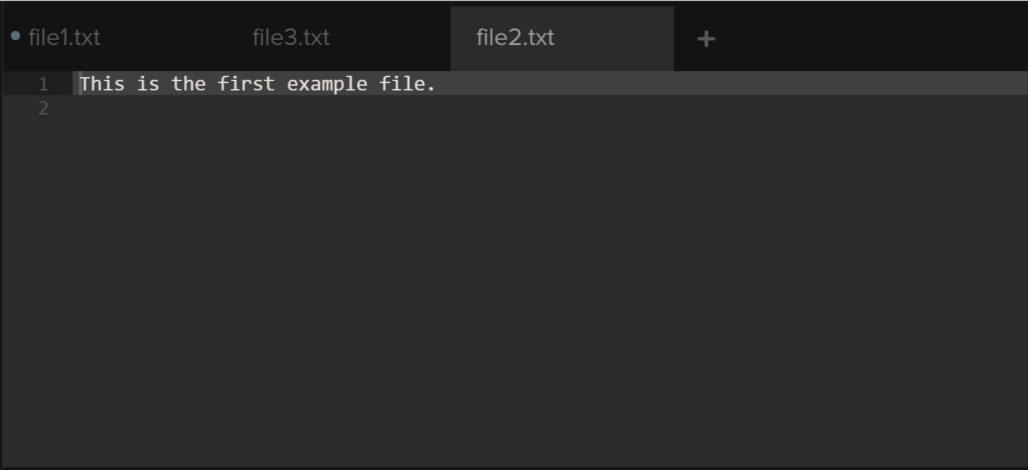
UNIX provides two types of links: (1) **hard links** and (2) **soft links**. A hard link creates a separate target file that has the same inode as the source file. Enter the following command to create a hard link between file1.txt and file2.txt:

ln file1.txt file2.txt



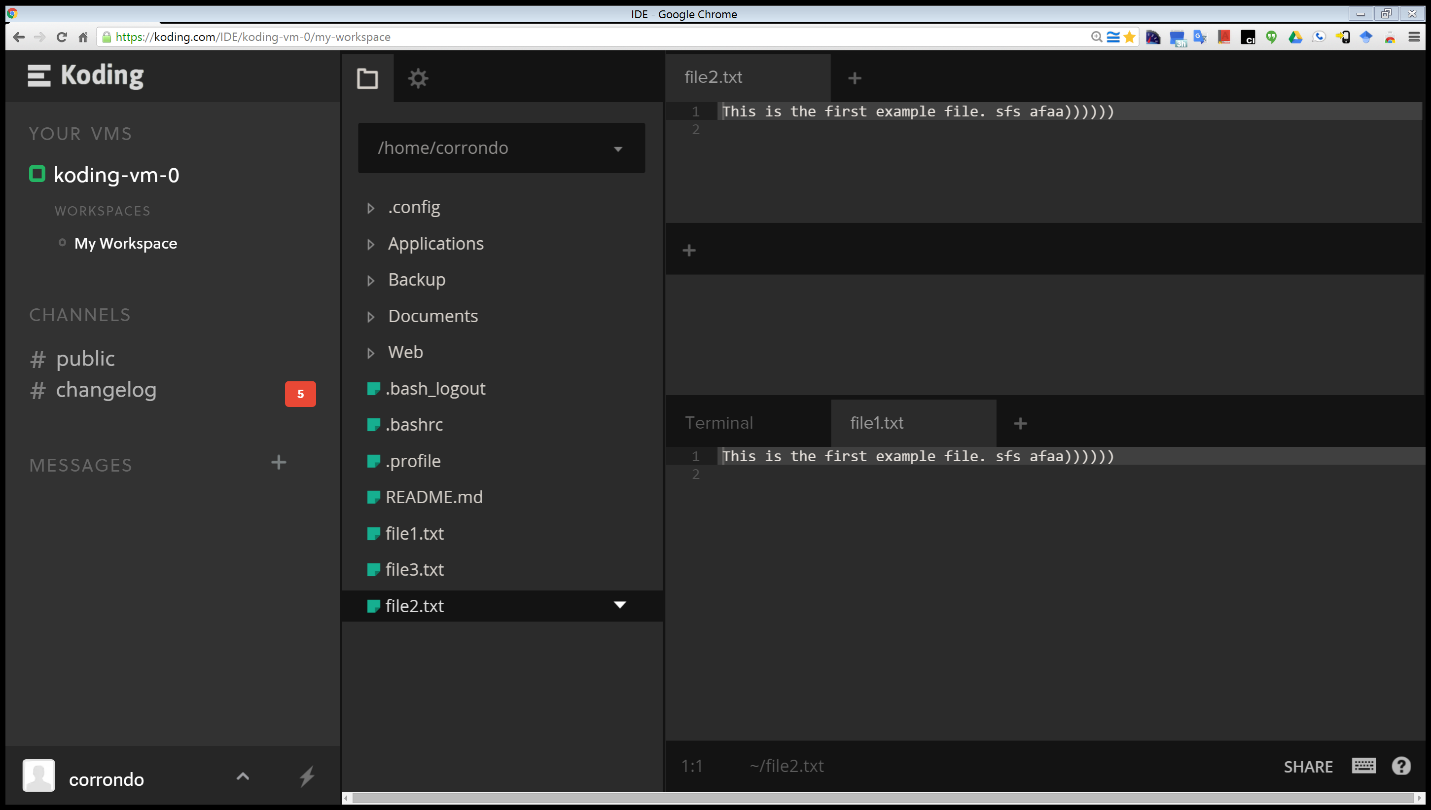
* Created file2.txt via **hard links**.

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?



* The inode values of file1.txt and file2.txt is **163214**.
* They are the **same**.
* They have the **same contents**.

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?

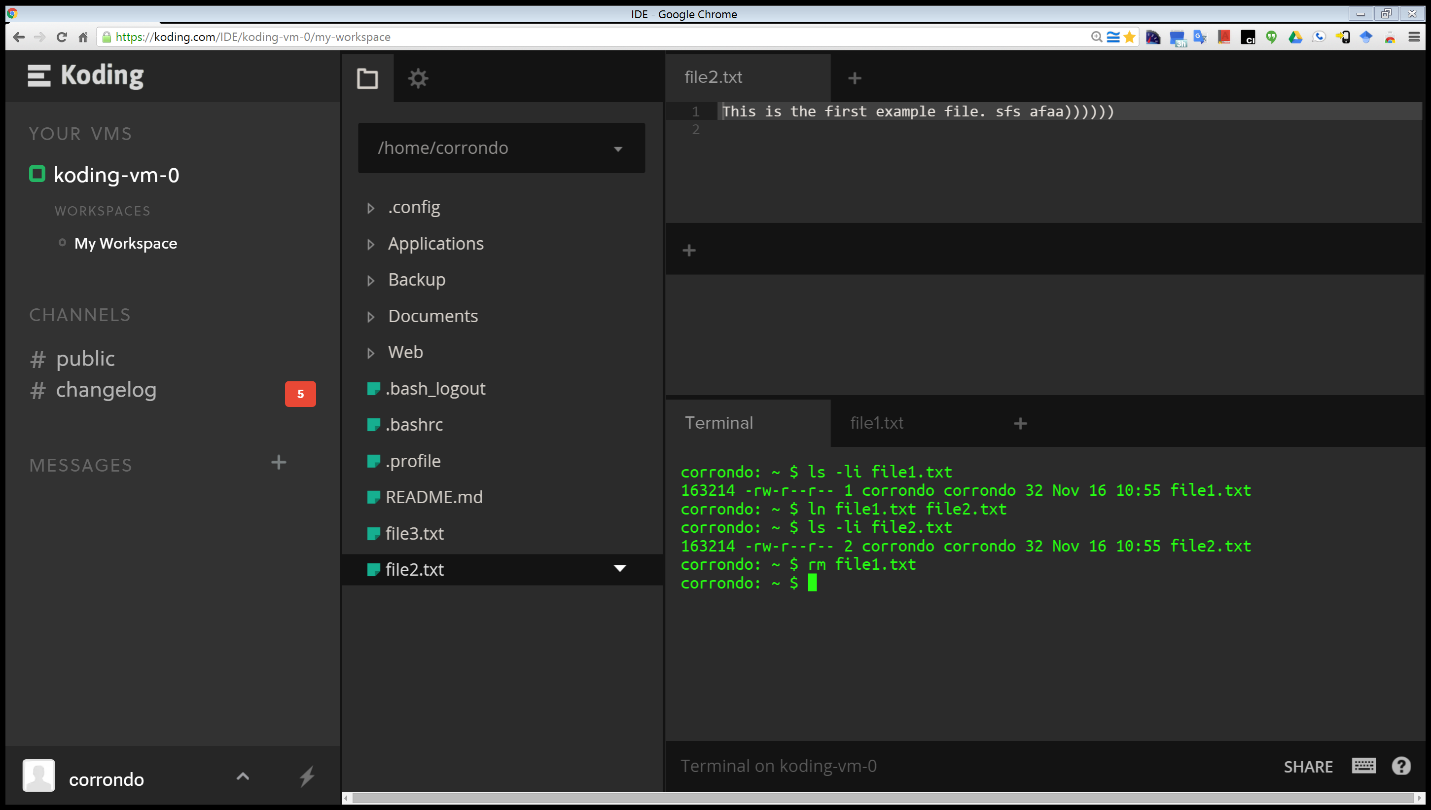


* After changing the contents of file2.txt the contents of file1.txt is the **same**.

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

rm file1.txt

Does file2.txt still exist as well?



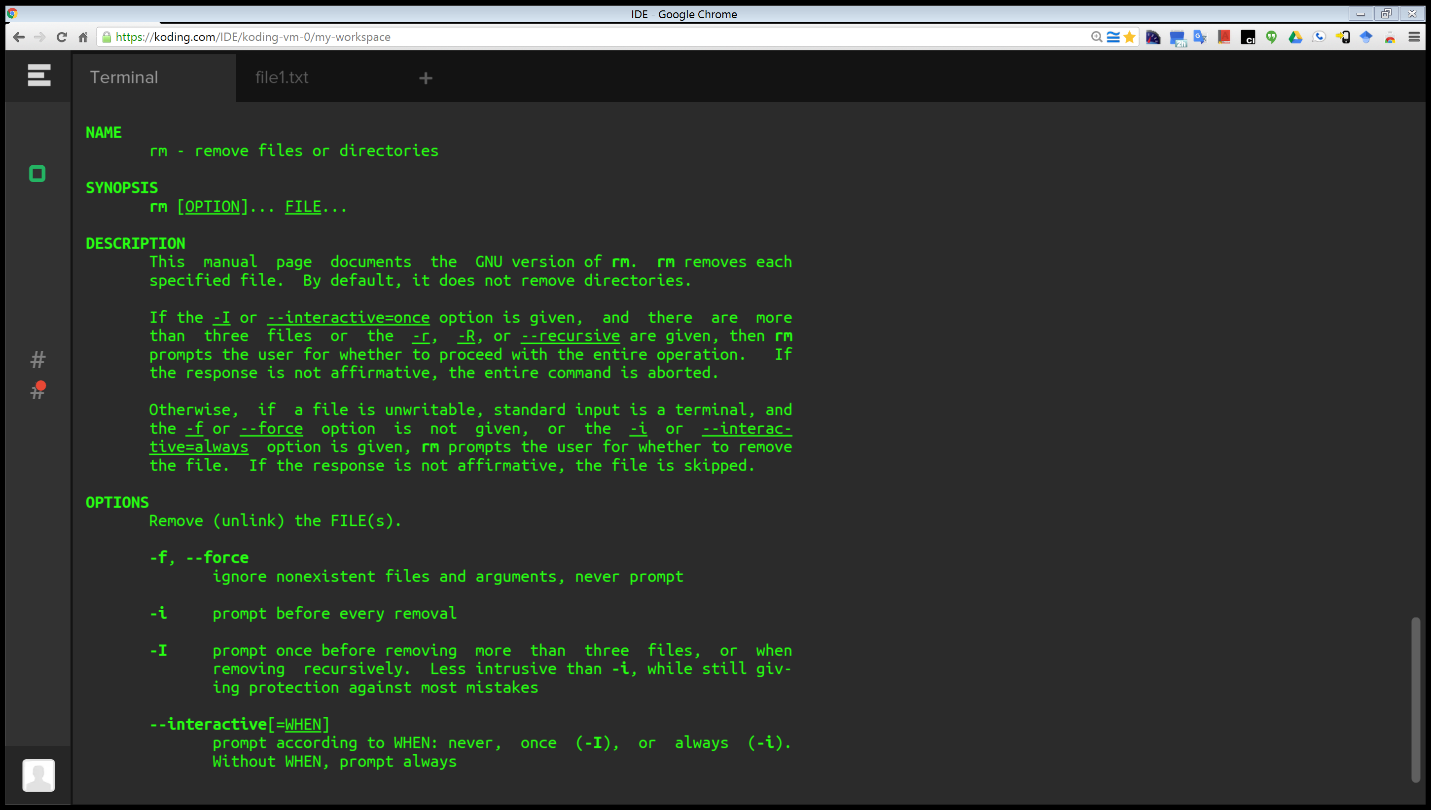
* After removing file1.txt the screenshot shows that file2.txt **still exist**.

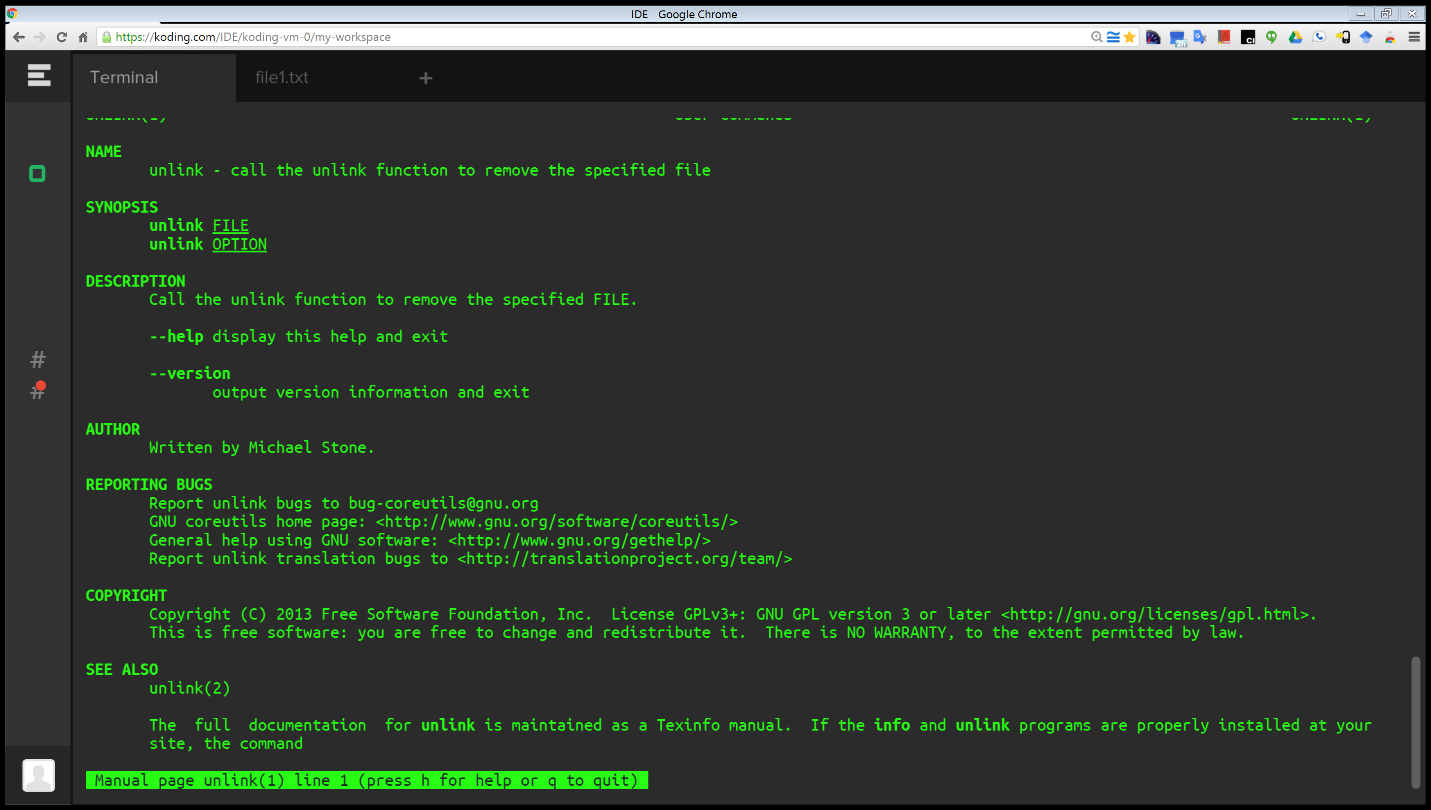
Now examine the man pages for both the rm and unlink commands.

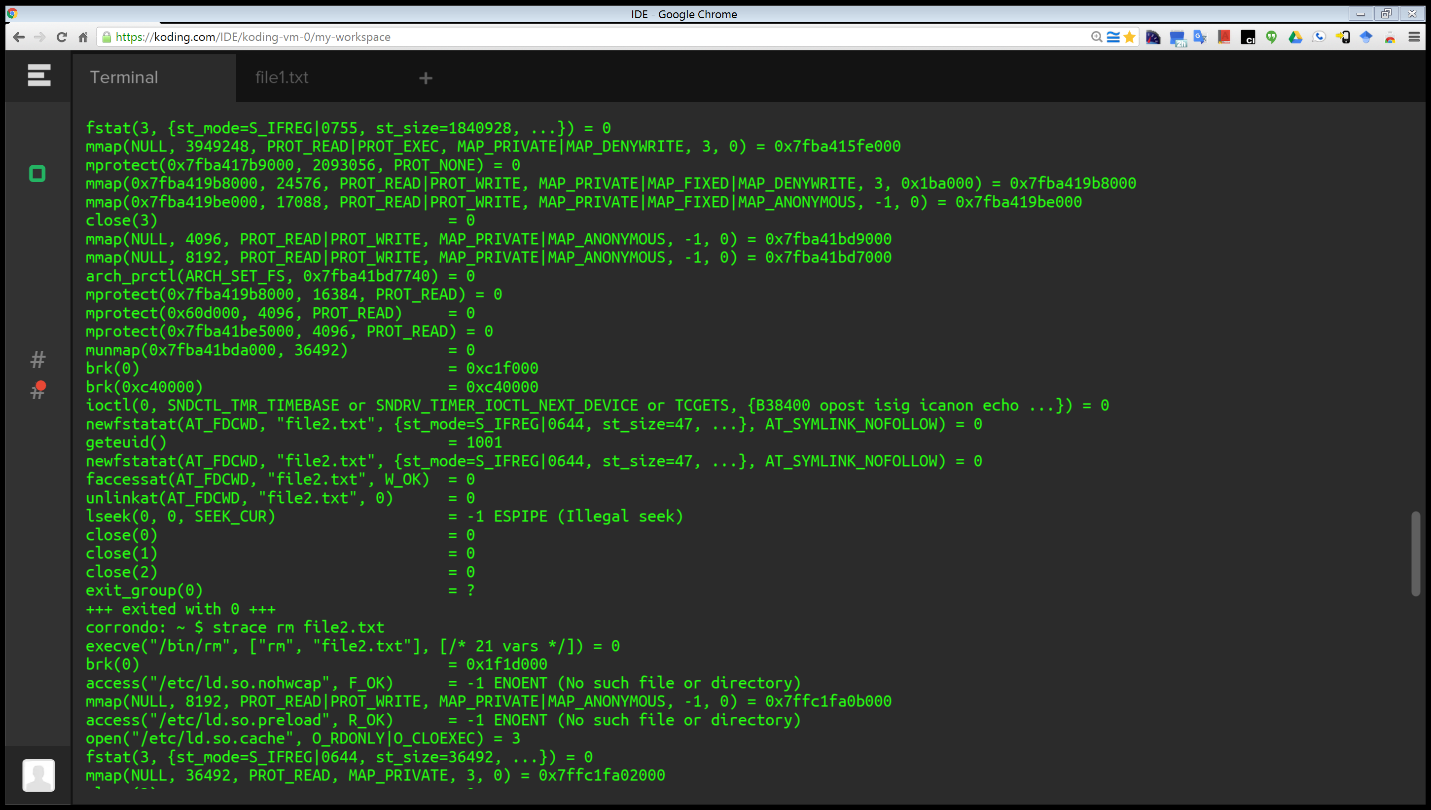
Afterwards, remove file2.txt by entering the command

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?







* The system call that is used for removing file2.txt is **unlink**.

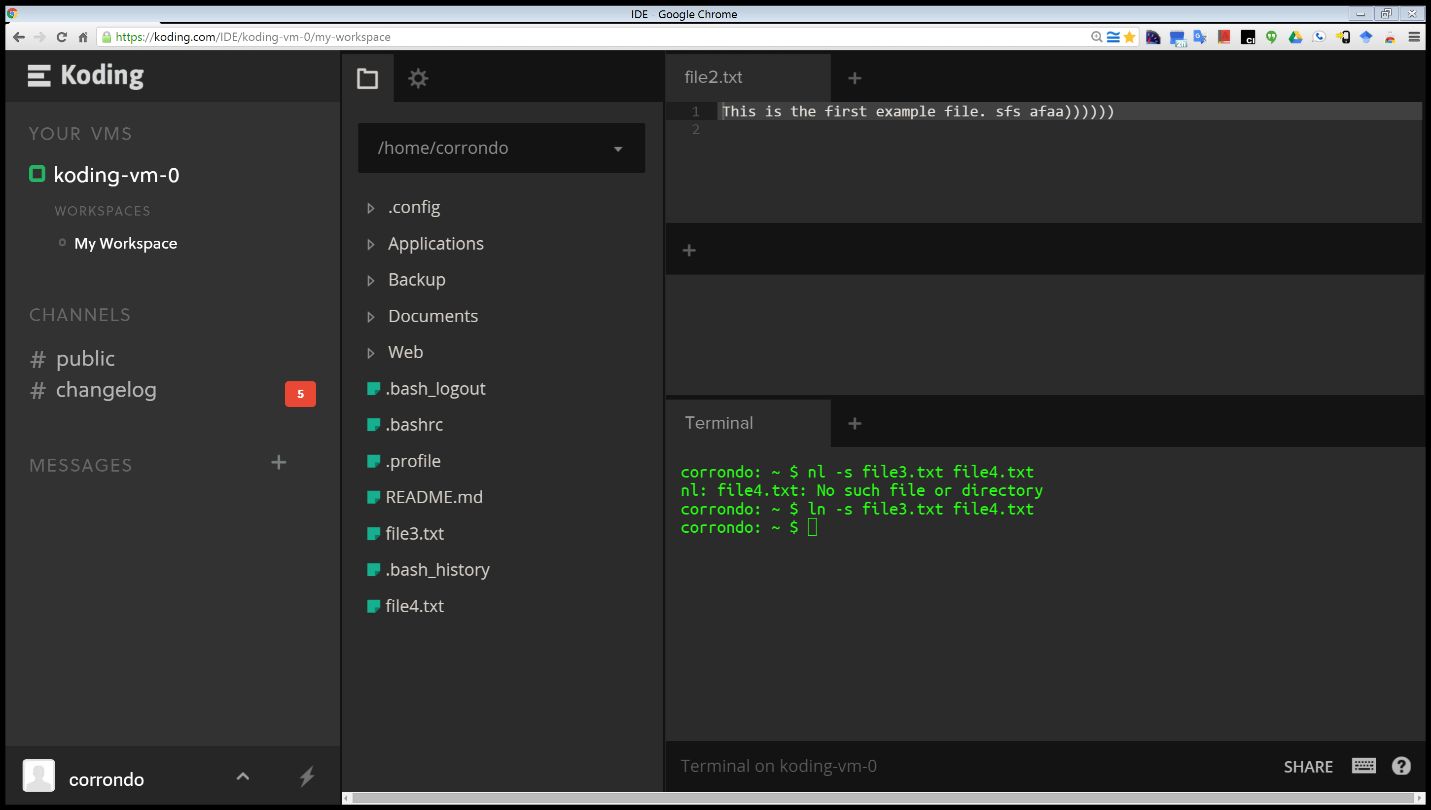
A soft link (or symbolic link) creates a new file that “points” to the name of the file it is linking to. In the source code available with this text, create a soft link to file3.txt by entering the following command:

ln -s file3.txt file4.txt

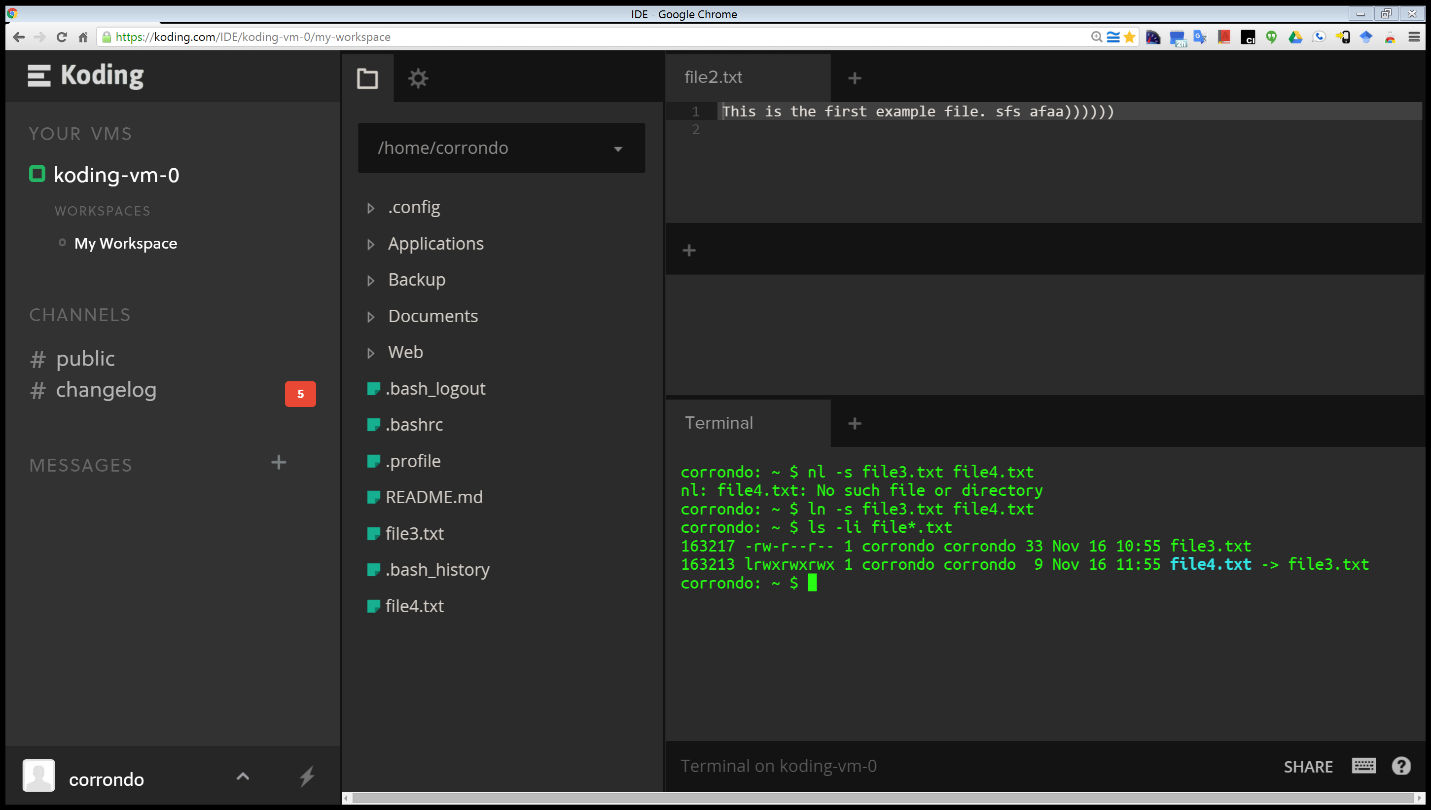
After you have done so, obtain the inode numbers of file3.txt and file4.txt using the command

ls -li file\*.txt

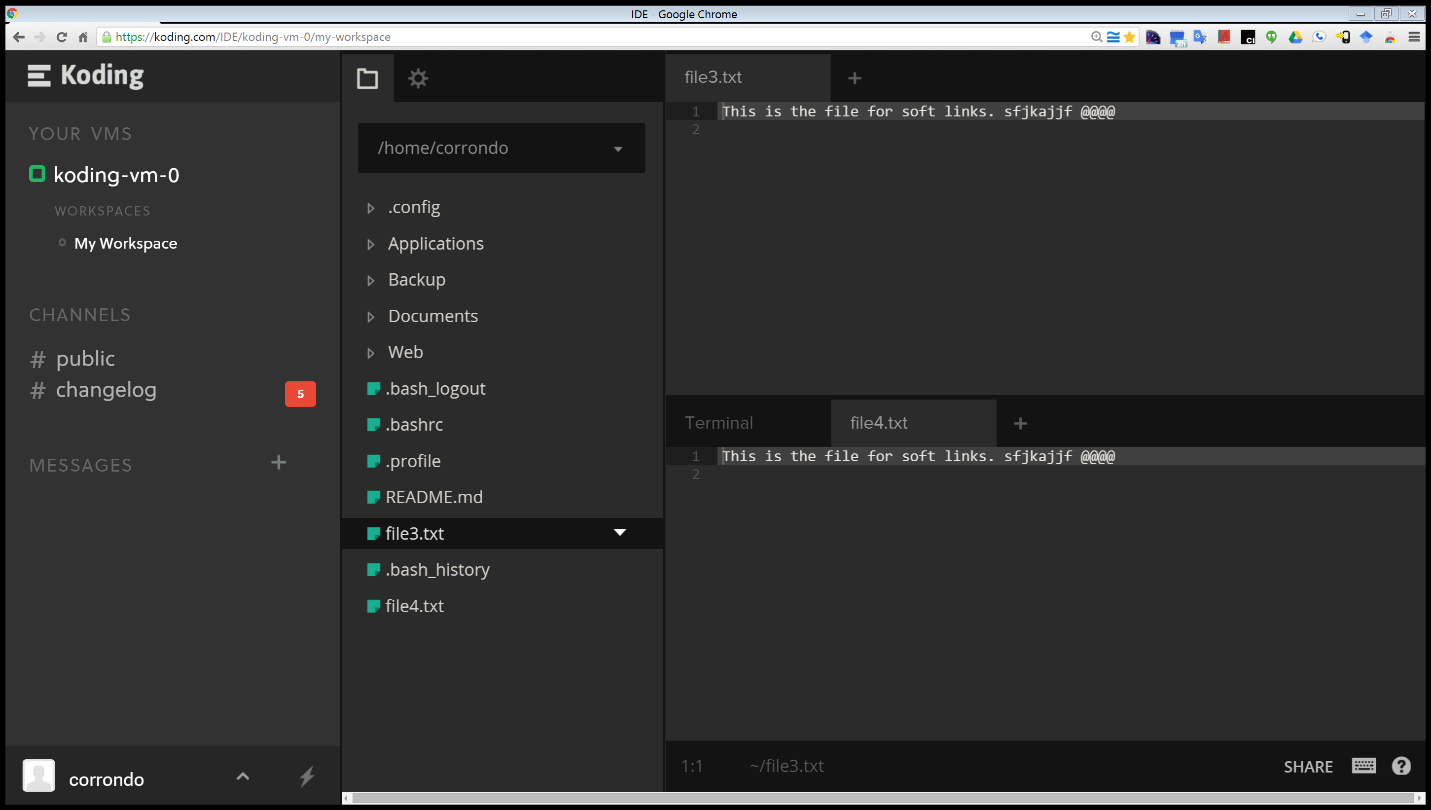
Are the inodes the same, or is each unique? Next, edit the contents of file4.txt. Have the contents of file3.txt been altered as well?



* Soft link **created**.

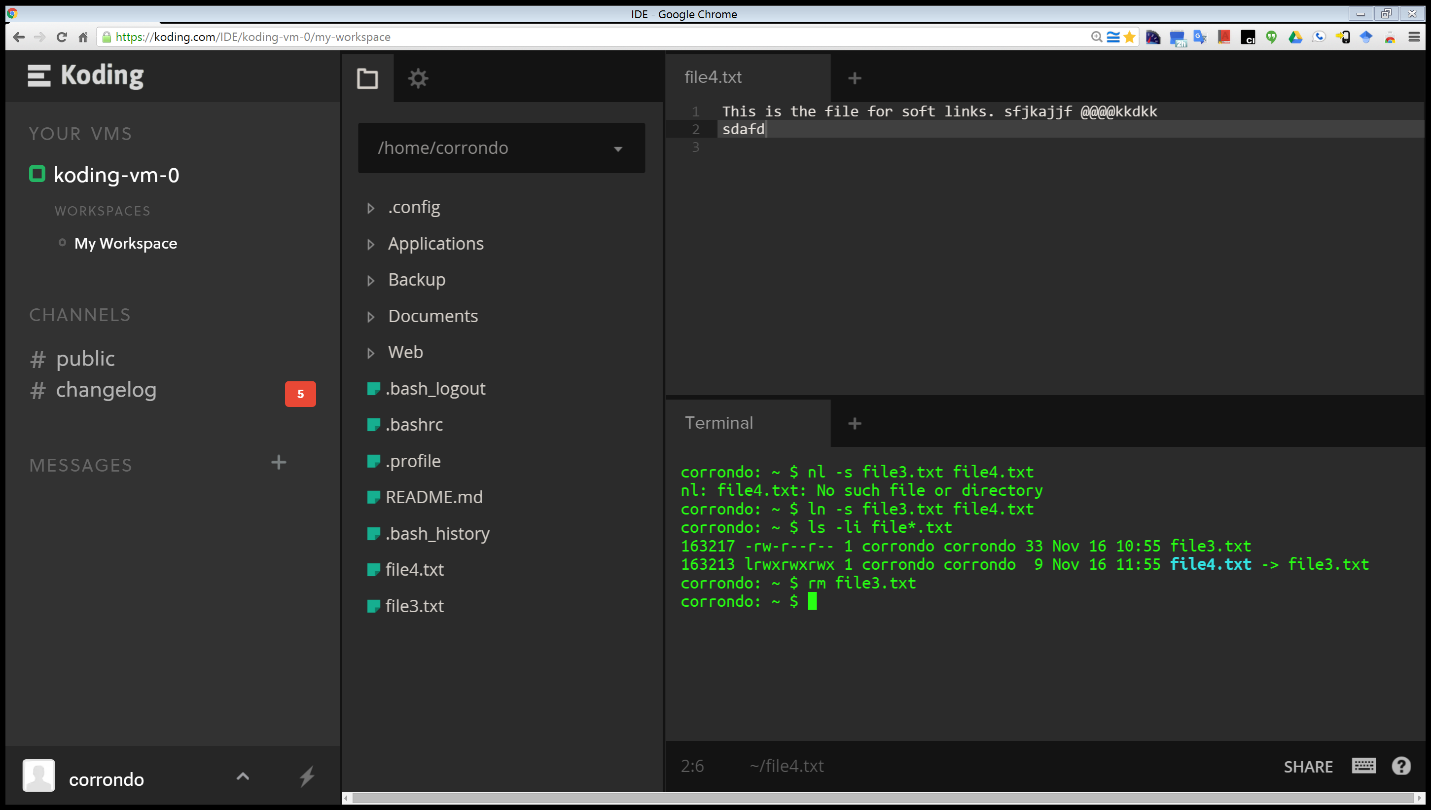


* Each inode is **unique**.



* After editing the contents of file4.txt the contents file3.txt has been **altered as well**.

Last, delete file3.txt. After you have done so, explain what happens when you attempt to edit file4.txt.



* After **deleting** file3.txt and **editing** file4.txt the changes made to file4.txt contents were saved and then in a spooky manner a **new** file3.txt with the **same** exactcontents as file4.txt was **automatically** **re-created**.

1. What are the various kinds of performance overheads associated with servicing an interrupt?

The various kinds of performance overheads associated with servicing an interrupt consist of:

* The cost of saving and restoring process state
* The cost of flushing the instruction pipeline and restoring the instructions into the pipeline when the process is restarted.

This is because during an interrupt the current executing process is interrupted and its state is stored in the correct process control block. To handle the interrupt the interrupt service routine is dispatched. After interrupt handling is completed, the state of the process is restored and the process is resumed.

1. Some DMA controllers support direct virtual memory access, where the targets of I/O operations are specified as virtual addresses and a translation from virtual to physical address is performed during the DMA. How does this design complicate the design of the DMA controller? What are the advantages of providing such a functionality?

Advantages:

* Direct virtual memory access allows the device to perform a transfer from two memory-mapped devices without the CPU intervening or main memory being used as a staging ground.
* The device only has to issue memory operations to the memory-mapped addresses of a target device
* Subsequent virtual address translation guarantee that the data is transferred to the appropriate device.

Disadvantage:

* Support for virtual address translation on addresses accessed by a DMA controller becomes necessary
* Requires the DMA controller to have an address-translation unit.
* The address translation results in both hardware and software costs
* May introduce coherence problems between the data structures maintained by the CPU for address translation and analogous structures used by the DMA controller.
* Aforementioned coherence issues increase system complexity plus require active handling.

References:

1. Operating System Concepts, Ninth Edition, Abraham Silberschartz, Peter bear Galvin, Greg Gagne. Entire Chapter 12-13 for each question in its respective section.
2. <https://kb.iu.edu/d/afjm>

I explored ways to structure file use, to allocate disk space, to recover freed space, to track the locations of data, and to interface other parts of the operating system to secondary storage. Throughout the chapter I noticed the concern for performance issues. Also, I leaned to describe the details of implementing local file systems and directory structures as well as to describe the implementation of remote file systems. The discussion about block allocation and free-block algorithms and trade-offs clarified much confusion related to the material. We discussed the I/O services provided by the operating system and the embodiment of such services in the application I/O interface. Explored the structure of an operating system’s I/O subsystem and discussed the principles and complexities of I/O hardware. I’m now better prepared to explain the performance aspects of I/O hardware and software.