7/14/2019

**INODE STRUCTURE**

**OS CIA-1 component 3**

Submitted By :- J Leo Clinton

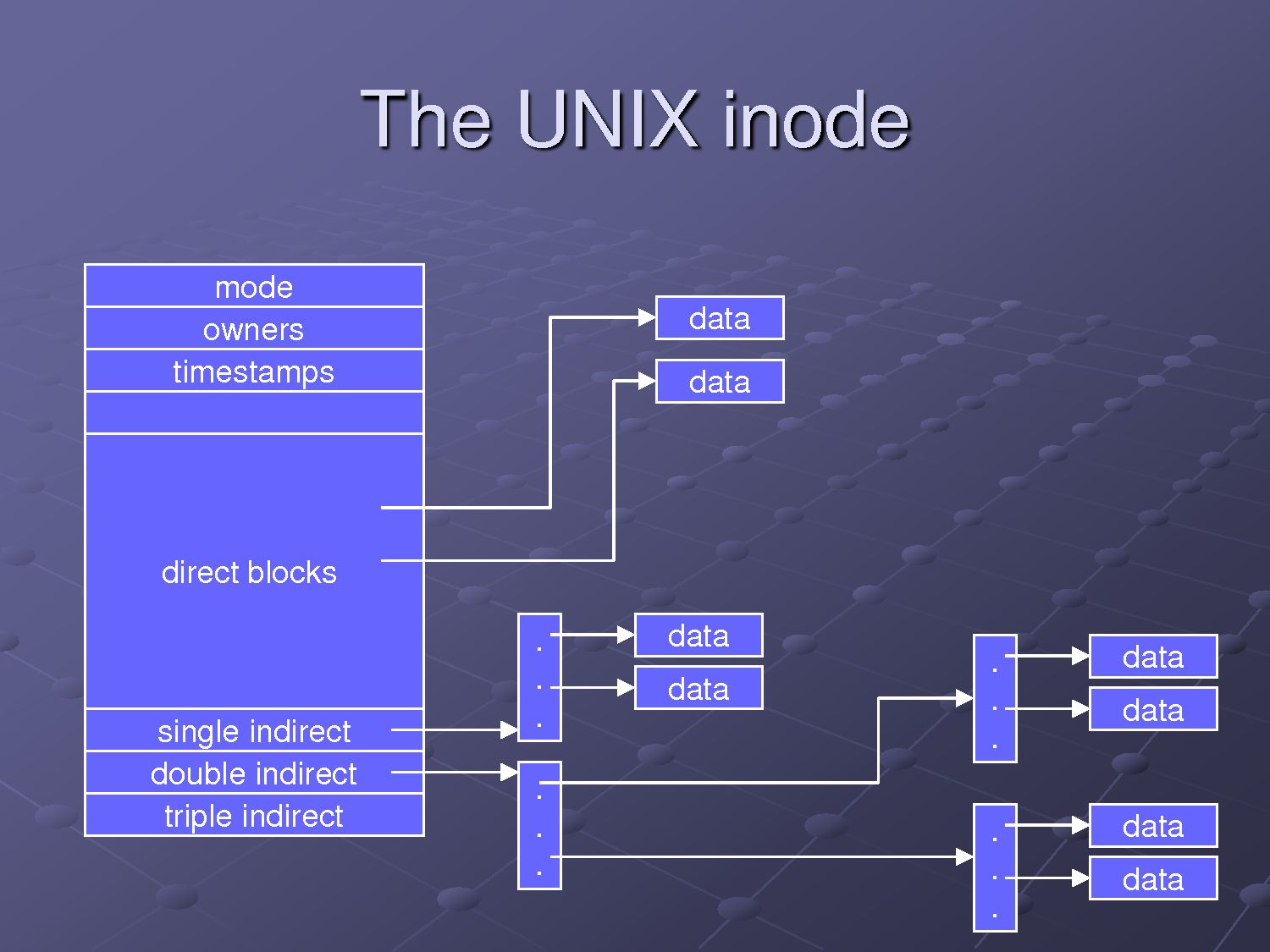
1847228

**INODE Structure in LINUX**

**INODE**

Every Linux file or directory has an inode, and this inode contains all of the file’s metadata .

For example, the inode contains a list of all the blocks in which a file is stored, the

 owner information for that file, permissions and all other attributes that are set for the file. In a sense, you could say that a file really is the inode, and names are attached to these inodes to make it easier for humans to work with them.

**INODE Number**

An inode is an entry in inode table, containing information i.e the meta data about a regular file and directory. An inode is a data structure on a traditional Unix-style file system such as ext3 or ext4.

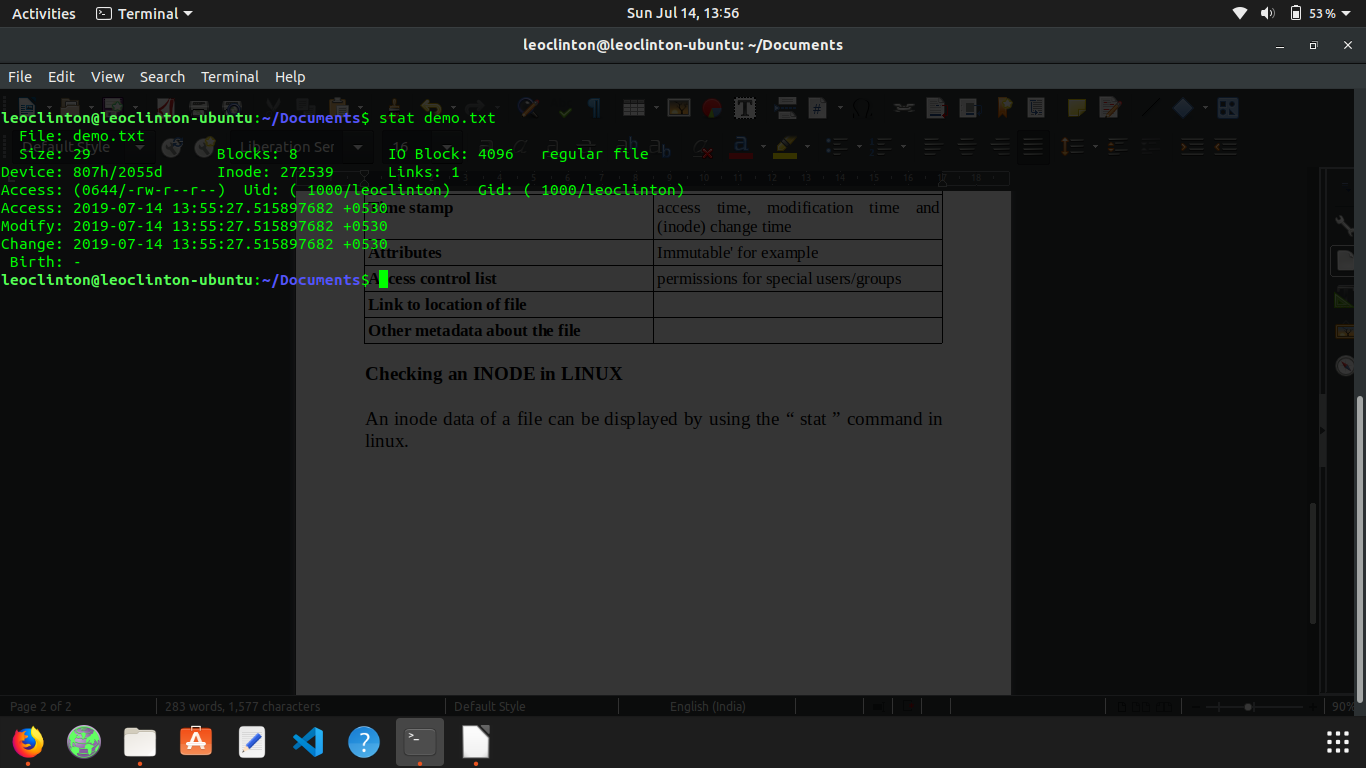
Linux extended filesystems such as ext2 or ext3 maintain an array of these inodes: the inode table. This table contains list of all files in that filesystem. The individual inodes in inode table have a unique number (unique to that filesystem), the inode number.

The inode contains the following information :-

|  |  |
| --- | --- |
| File type | regular file, directory, pipe,link. |
| Permissions to that file | read, write, execute |
| Link count | The number of hard link relative to an inode |
| User ID | owner of file |
| Group ID | group owner |
| Size of file | or major/minor number in case of some special files |
| Time stamp | access time, modification time and (inode) change time |
| Attributes | Immutable' for example |
| Access control list | permissions for special users/groups |
| Link to location of file |  |
| Other metadata about the file |  |

**Checking an INODE in LINUX**

An inode data of a file can be displayed by usingthe “ stat ” command in linux.



## Direct Block Pointers :-

In an ext2 file system an inode consists of only 15 block pointers. The first 12 block pointers are called as Direct Block pointers. Which means that these pointers point to the address of the blocks containing the data of the file.

12 Block pointers can point to 12 data blocks. So in total the Direct Block pointers can address only 48K(12 \* 4K) of data. Which means if the file is only of 48K or below in size, then inode itself can address all the blocks

## Indirect Block Pointers

whenever the size of the data goes above 48k(by considering the block size as 4k), the 13th pointer in the inode will point to the very next block after the data(adjacent block after 48k of data), which inturn will point to the next block address where data is to be copied.

Now as we have took our block size as 4K, the indirect block pointer, can point to 1024 blocks containing data(by taking the size of a block pointer as 4bytes, one 4K block can point to 1024 blocks because 4 bytes \* 1024 = 4K).

## Double indirect Block Pointers

Now if the size of the file is above 4MB + 48K then the inode will start using Double Indirect Block Pointers, to address data blocks. Double Indirect Block pointer in an inode will point to the block that comes just after 4M + 48K data, which intern will point to the blocks where the data is stored.

Double Indirect block pointer also is inside a 4K block as every blocks are 4K, Now block pointers are 4 bytes  in size, as mentioned previously, so Double indirect block pointer can address 1024 Indirect Block pointers(which means 1024 \* 4M =4G). So with the help of a double indirect Block Pointer the size of the data can go upto 4G.

## Triple Indirect Block Pointers

Now this triple Indirect Block Pointers can address upto 4G \* 1024 = 4TB, of file size. The fifteenth block pointer in the inode will point to the block just after the 4G of data, which intern will point to 1024 Double Indirect Block Pointers.