6.4 File Statuses

We've already seen part of a file status in our previous chapters without knowing it. One of these statuses is the file's mode. We've already seen how to access this with the `ls -l` command, but there's really a little more happening. A file's "status" is kept in an **i-node**. i-nodes are files that contain meta-data of the file such as its mode, (which we've already seen), its file size, name, ID, and a number of other bits of information.

To access this information, we can use the `stat` command:

```
stat file
```

> File: ‘file’
>
> Size: 0 Blocks: 0 IO Block: 4096 regular empty file
>
> Device: 831h/2097d Inode: 293493374 Links: 1
>
> Access: (0644/-rw-r--r--) Uid: (401451/ mjlny2) Gid: (401451/ UNKNOWN)
>
> Access: 2020-07-05 15:15:36.677271695 -0500
>
> Modify: 2020-07-05 15:15:36.677271695 -0500
>
> Change: 2020-07-05 15:15:36.677271695 -0500
>
> Birth: -

Mode

We've already talked pretty extensively about mode, however, we've really only worked with permissions ourselves. When working on massive servers, there are often a number of groups. Groups are defined by the system administrator who has super user privileges. Groups are defined at the root level at:

```
/etc/group
```

If you `cat` this file, you can print out all of the user groups within your system. The numbers next to the group names are the associated IDs.

To see your own group within a system, you can use the `groups` command or the `lid` command, however, the `lid` command requires super user privileges to execute.

```
groups username
```

In practice we can see:

```
groups mjlny2
```

> mjlny2 : groups: cannot find name for group ID 401451
>
> 401451 umumslfaculty@um.umsystem.edu irdataaccess@stl.umsl.edu umslcognossr@stl.umsl.edu

The output of groups shows all of your associated groups by name. If the group doesn't have a name, only the associated ID will be shown.

Changing a File's Group

To change the group of a file to one of your associated groups, you can use the `chgrp` command:

```
chgrp groupid filename
```

In practice:

```
ls -l somefile
```

> -rw-r--r-- 1 mjlny2 401451 0 Jul 5 15:40 somefile

In the above, the group id for *somefile* is 401451 (it has no associated name). Changing the group to another with the `chgrp` command looks like:

```
chgrp umumslfaculty@um.umsystem.edu somefile && ls -l somefile
```

> -rw-r--r-- 1 mjlny2 umumslfaculty@um.umsystem.edu 0 Jul 5 15:40 somefile

Now the associated group to *somefile* is no longer `401451` but is now `umumslfaculty@um.umsystem.edu`. Naturally, the groups that you will see on your own machine / server will be different than the examples displayed here.

Changing a File's Owner

Just as you can change the group of a file with `chgrp` you can change the owner of a file with `chown`. Changing the owner of a file, however, requires that you have super user privileges, which will be discussed later. The general look of the command however, is:

```
chown newOwnerId filename
```

File Access Steps:

When a process attempts to access a file/directory access is determined by a series of steps from the most restrictive to the least restrictive:

1. Process is started by user. Process userId matches the user who started it.
2. Process accessing file:
    1. Does process userId match the file's userId
        1. If yes: check mode for user's access privileges.
    2. Does process's associated groups match the File's groups:
        1. If yes: check the mode for the group's access privilieges.
    3. Check the mode for other's access privileges.

Groups and Modes

It is possible to elevate another user to have access to other modes with `setuid` and `setgid`. Typically this is done for one time services, or to get around permissions for a very specific reason.

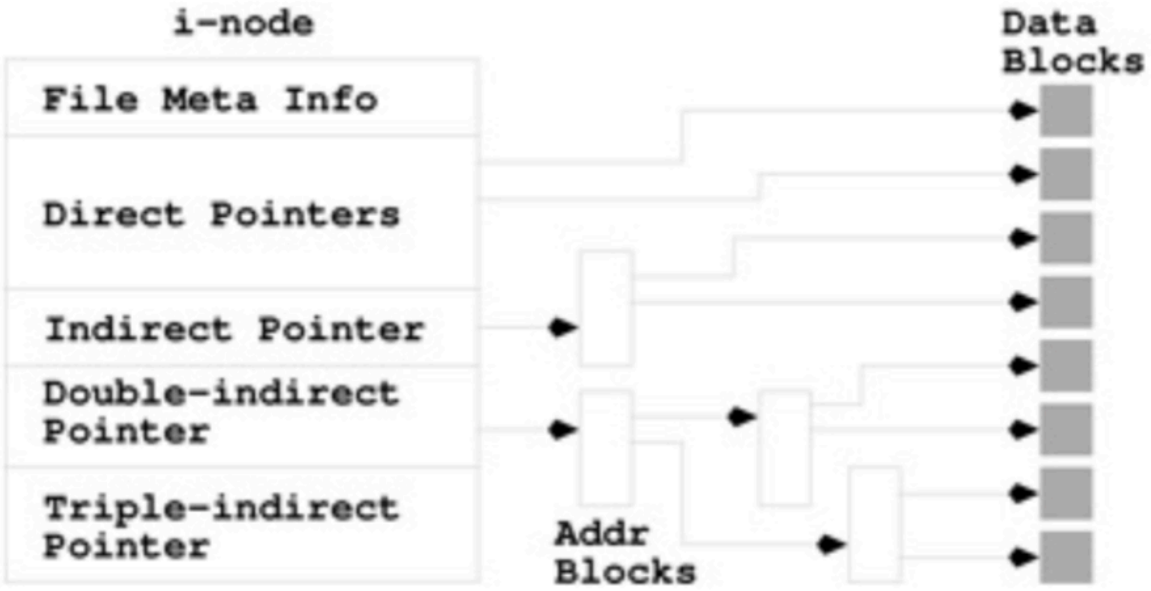
Creating a group can be done with the `groupadd` command, but again, to do so you need super user privileges. When adding a group, you're ultimately just appending a group to the group list in `/etc/group`.

6.5 File System Implementation

Linux machines most often have at least one file system. Each file system is broken up into blocks where we can store our files. Files are stored in blocks on a filesystem in byte sizes of 1024, 2048, 4096, etc.

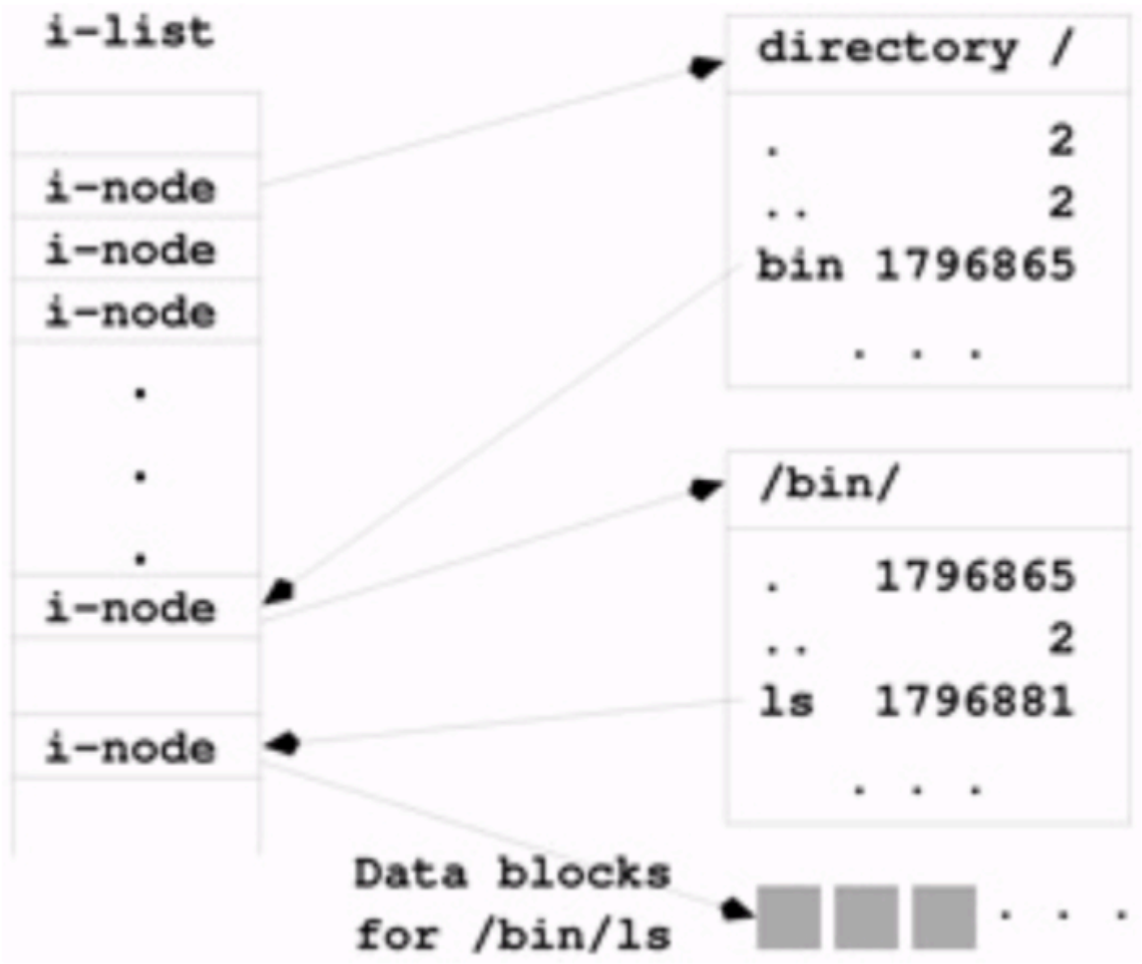
These blocks all have addresses of where they are in the system, their statuses, etc. by use of i-nodes. All i-nodes exist within a list: **i-list**. For each file, the i-node has information on the file's metadata, its mode, but also, where specifically the file's data are stored in the system (i.e. on what blocks the file data actually exists).

The file's i-nodes can have 12 direct pointers to the blocks where the file's data are, indirect pointers, double indirect pointers, and triple indirect pointers. The indirect pointers are essentially pointing to other i-nodes that then point to the file's blocks, double indirect point to indirect pointers, and triple indirect point to double indirect pointers. The point of doing this is for significantly larger files.



> *-Wang, P (Ed). (2018). Mastering Modering Linux (pp. 190) Boca Raton: CRC Press*

When navigating the system, what's happening behind the scenes is that the shell is interacting with us in a relatively human readable manner, but then the operating system is taking those commands and accessing the i-list, finding the associated i-node, and then working from there to access either another i-node or specific data blocks:



> *-Wang, P (Ed). (2018). Mastering Modering Linux (pp. 190) Boca Raton: CRC Press*

Mounted File Systems

The i-list is a representation of your entire filesystem as a linear array. Linux by default has its root filesystem, but other systems can can be added (or removed) to the filesystem list by **mounting** (and **unmounting**). Super users can mount new drives with the `mount` command, and unmount with `unmount`.

While only a super user has access to mount and unmount filesystems, you can view all mounted filesystems by using `mount` with no additional arguments. You can additionally see other status by using the `df` command to see a filesystem, its size, how much space is used/available and where it's mounted.

File systems as a whole are considered a **super block**, or a block that hosts metadata for the filesystem like the ilist length, size, mode permissions etc.