# Understand UNIX / Linux Inodes Basics with Examples

*by* HIMANSHU ARORA *on* JANUARY 16, 2012

Several countries provides a unique identification number (for example, social security number in the USA) to the people who live in that country. This makes it easier to identify an individual uniquely. This makes it easier to handle all the paper work necessary for an individual by various government agencies and financial institutions.

Similar to the social security number, there is a concept of Inode numbers which uniquely exist for all the files on Linux or *nix systems.

## Inode Basics

An Inode number points to an Inode. An Inode is a data structure that stores the following information about a file :

- Size of file
- Device ID
- User ID of the file
- Group ID of the file
- The file mode information and access privileges for owner, group and others
- File protection flags
- The timestamps for file creation, modification etc
- link counter to determine the number of hard links
- Pointers to the blocks storing file's contents

Please note that the above list is not exhaustive. Also, the name of the file is not stored in Inodes (We will come to it later).

When a file is created inside a directory then the file-name and Inode number are assigned to file. These two entries are associated with every file in a directory. The user might think that the directory contains the complete file and all the extra information related to it but this might not be the case always. So we see that a directory associates a file name with its Inode number.

When a user tries to access the file or any information related to the file then he/she uses the file name to do so but internally the file-name is first mapped with its Inode number stored in a table. Then through that Inode number the corresponding Inode is accessed. There is a table (Inode table) where this mapping of Inode numbers with the respective Inodes is provided.

## Why no file-name in Inode information?

As pointed out earlier, there is no entry for file name in the Inode, rather the file name is kept as a separate entry parallel to Inode number. The reason for separating out file name from the other information related to same file is for maintaining hard-links to files. This means that once all the other information is separated out from the file name then we can have various file names which point to same Inode.

For example :

```
$ touch a


$ ln a a1


$ ls -al
drwxr-xr-x 48 himanshu himanshu 4096 2012-01-14 16:30 .
drwxr-xr-x 3 root root 4096 2011-03-12 06:24 ..
-rw-r--r-- 2 himanshu family 0 2012-01-14 16:29 a
-rw-r--r-- 2 himanshu family 0 2012-01-14 16:29 a1
```

In the above output, we created a file ‘a’ and then created a hard link a1. Now when the command ‘ls -al’ is run, we can see the details of both ‘a’ and ‘a1’. We see that both the files are indistinguishable. Look at the second entry in the output.

This entry specifies number of hard links to the file. In this case the entry has value '2' for both the files.

Note that Hard links cannot be created on different file systems and also they cannot be created for directories.

## When are Inodes created?

As we all now know that Inode is a data structure that contains information of a file. Since data structures occupy storage then an obvious question arises about when the Inodes are created in a system? Well, space for Inodes is allocated when the operating system or a new file system is installed and when it does its initial structuring. So this way we can see that in a file system, maximum number of Inodes and hence maximum number of files are set.

Now, the above concept brings up another interesting fact. A file system can run out of space in two ways :

- No space for adding new data is left
- All the Inodes are consumed.

Well, the first way is pretty obvious but we need to look at the second way. Yes, its possible that a case arises where we have free storage space but still we cannot add any new data in file system because all the Inodes are consumed. This may happen in a case where file system contains very large number of very small sized files. This will consume all the Inodes and though there would be free space from a Hard-disk-drive point of view but from file system point of view no Inode available to store any new file.

The above use-case is possible but less encountered because on a typical system the average file size is more than 2KB which makes it more prone to running out of hard disk space first. But, nevertheless there exists an algorithm which is used to create number of Inodes in a file system. This algorithm takes into consideration the size of the file system and average file size. The user can tweak the number of Inodes while creating the file system.

## Commands to access Inode numbers

Following are some commands to access the Inode numbers for files :

## 1) Ls -i Command

As we explained earlier in our Unix LS Command: 15 Practical Examples article, the flag -i is used to print the Inode number for each file.

```
$ ls -i
1448240 a 1441807 Desktop 1447344 mydata 1441813 Pictures 1442737 testfile 1448
1448240 a1 1441811 Documents 1442707 my_ls 1442445 practice 1442739 test.py
1447139 alpha 1441808 Downloads 1447278 my_ls_alpha.c 1441810 Public 1447099 Uns
1447478 article_function_pointer.txt 1575132 google 1447274 my_ls.c 1441809 Temp
1442390 chmodOctal.txt 1441812 Music 1442363 output.log 1448800 testdisk.log 157
```

See that the Inode number for 'a' and 'a1' are same as we created 'a1' as hard link.

## 2) Df -i Command

df -i command displays the inode information of the file system.

```
$ df -i
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/sda1            1875968  293264 1582704   16% /
none                  210613     764  209849    1% /dev
none                  213415       9  213406    1% /dev/shm
none                  213415      63  213352    1% /var/run
none                  213415       1  213414    1% /var/lock
/dev/sda2            7643136  156663 7486473    3% /home
```

The flag -i is used for displaying Inode information.

## 3) Stat Command

Stat command is used to display file statistics that also displays inode number of a file

```
$ stat a
File: `a'
Size: 0 Blocks: 0 IO Block: 4096 regular empty file
Device: 805h/2053d Inode: 1448240 Links: 2
Access: (0644/-rw-r--r--) Uid: ( 1000/himanshu) Gid: ( 1001/ family)
Access: 2012-01-14 16:30:04.871719357 +0530
Modify: 2012-01-14 16:29:50.918267873 +0530
Change: 2012-01-14 16:30:03.858251514 +0530
```

## Example Usage Scenario of an Inode number

1. Suppose there exist a file name with some special character in it. For example: “ab*
2. Try to remove it normally using rm command, you will not be able to remove it.
3. However using the inode number of this file you can remove it.

Lets see these steps in this example :

1) Check if the file exists:

```
$ ls -i
1448240 a 1447274 my_ls.c
1448240 a1 1442363 output.log
```

```
1448239 "ab* 1441813 Pictures
1447139 alpha
```

So we have a file with name “ab* in this directory

2) Try to remove it normally:

```
$ rm "ab*
> ^C
$ rm "ab*
> ^C
$
```

See that I tried couple of times to remove the file but could not.

3) Remove the file using Inode number:

As we discussed earlier in our find command examples article, you can search for a file using inode number and delete it.

```
$ find . -inum 1448239 -exec rm -i {} \;
rm: remove regular empty file `./"ab*'? y
```

```
$ ls -i
1448240 a 1447274 my_ls.c
1448240 a1 1442363 output.log
1447139 alpha 1441813 Pictures
```

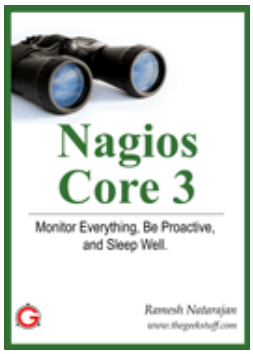
So we used the find command specifying the Inode number of the file we need to delete. The file got deleted. Though we could have deleted the file otherwise also by using the command rm \”ab* instead of using the complicated find command example above but still I used it to demonstrate one of the use of Inode numbers for users.

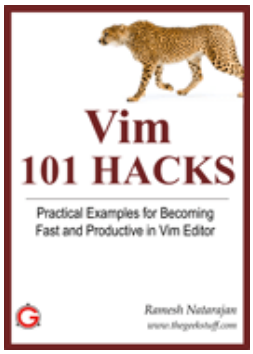
Post Add your comment

## If you enjoyed this article, you might also like..

1. 50 Linux Sysadmin Tutorials
2. 50 Most Frequently Used Linux Commands (With Examples)
3. Top 25 Best Linux Performance Monitoring and Debugging Tools
4. Mommy, I found it! – 15 Practical Linux Find Command Examples
5. Linux 101 Hacks 2nd Edition eBook Free

- Awk Introduction – 7 Awk Print Examples
- Advanced Sed Substitution Examples
- 8 Essential Vim Editor Navigation Fundamentals
- 25 Most Frequently Used Linux IPTables Rules Examples
- Turbocharge PuTTY with 12 Powerful Add-Ons






Comments on this entry are closed.

### BachiTux

January 16, 2012, 12:27 am

Excellent guide, boys! Thanks!!!

∞

### Anonymous

January 16, 2012, 4:38 am

Thanks, Now I have a clearer understanding of Inodes.

∞

**Prakashkumar**

January 16, 2012, 5:09 am

Hi,

Nice info. i amazed the way you comparing the things, and you make the understanding more faster without compromising anything

∞

**Johan Volkers**

January 16, 2012, 6:21 am

a simple
rm ‘ab*’
will remove the file ab* also

∞

**brad**

January 16, 2012, 10:20 am

Is there an easy way to recover or undelete a deleted inode? The answer is no, but i don’t really understand why. It seems like a simple thing to keep deleted inodes in a salvage location until purged or until the file is overwritten.

∞

**vinay jhedu**

January 16, 2012, 11:46 am

This is really great article , well explained . I am reading your articles regularly now .

∞

**Tracy Hearst**

January 16, 2012, 2:33 pm

Does the inode information also get backed up?

When backing up from a ext4 to FAT file systems, information like permissions get lost. Is this the reason?

How does one preserve how files are hard linked when transferring files to another file systems?

∞

**jalal hajigholamali**
January 16, 2012, 8:18 pm

Thanks

∞

**linshibo**
January 17, 2012, 8:20 pm

about the example：you can remove it by this： rm \”ab*

∞

**Gareth Williams**
January 18, 2012, 11:11 am

Just a word of warning if you have multiple partitions! I know the above find command to delete an inode works and prompts for user confirmation and is safe to use if used correctly, it is generally better to use an absoulte path such as “/home” as apposed to current working directory “.” The reason being that inode numbers are unique but partition specific. So if you filesystem is split into multiple partitions e.g. /boot /home /root /var /usr /tmp… etc, each could all have a file with an inode number which is the same e.g. inode 12345. So if the file you want to delete is “/home/test/test.txt” and it has inode number 12345 and you are currently in directory “/var” and you run “find . -inum 12345 -exec rm -i {} \;” it will attempt to delete a file with inode number 12345 on the /var parition than if your current working directory was /home. Ultimately this depends on whether you have multiple partitions but always best to be safe 🙂

∞

**Srinivas**

January 19, 2012, 12:05 am

Hi

Can somebody explain to me why we cannot create a hard link on folder? as well as why cannot create a hard link that span across file systems? I did lot of search on Google but didn't find any meaningful explanation

Srinivas

∞

**Gareth Williams**

January 19, 2012, 4:46 am

Srinivas, for the reasons mentioned in my previous comment you cannot create a hard link across multiple filesystems. The reason being a hard link points to a specific inode and different filesystems have different inode tables therefore if it was possible it would point to the wrong file. For example if you have two seperate filesystems e.g. /boot and /tmp and you try to hard link file /boot/initrd.img-2.6.32-5-686 which has inode 123456 to /tmp/hardlink. As /tmp is on a different filesystem its inode 123456 will point to something completely different or nothing at all hence it will not work. This is why we have soft links which can span multiple filesystems and act like shortcuts as apposed to inode pointers!

∞

**Srinivas**

January 19, 2012, 11:48 pm

Thanks Gareth for your good explanation. I was sensed the same however would like to confirm. However am still getting answer to primary question of why can't we create against a folder which resides on same file system?

Thanks you very much for replying to my question

Srinivas

∞

**Srinivas**

January 19, 2012, 11:50 pm

Please read as “However am still not getting answer to my primary question” due to typo mistake

Srinivas

∞

**cc**

January 25, 2012, 1:53 am

Gareth, great coverage of an under-appreciated topic. thanks

Srinivas, pls read Gareth’s comments for the answers to your questions

∞

**Srinivas**

January 25, 2012, 9:05 am

Right, however am still looking answer to my 2nd question.

“Why we cannot create hard link for folder on same file system?”

appreciated if any body answer to above question …

Srinivas

∞

**Gareth Williams**

January 25, 2012, 9:46 am

Srinivas, my understanding with regards to not being able to hard link to directories is not because it is impossible to implement, infact it is possible and was allowed in some distributions in the past but I believe because of potential security implications the kernel no longer allows them. Probably coupled by the fact that symbolic links can handle this across multiple filesystems why bother?

If you look at the man page for the “ln”, it should mention the “-d -F” arguments which are/were used for linking directories, although as mentioned “note: will probably fail due to system restrictions, even for the superuser”.

If this functionality were allowed it is open to abuse were people could create infinite loops either accidently or maliciously. Imagine if you created a couple of hard links to folders in such a way that they formed a circular reference. Then if you was to run a command against one of those directories it would run around in circles forever. Remember hard linking a file is like creating another file in a directory without using twice the space, no big deal. Hard linking a directory on the other hand for commands like “find, updatedb” or anything that recurses through a directory structure could end up all over the place! They are not as simple to work with as files.

Remember symbolic links (soft links) are distinguishable from the files/folders in which the refer too as they have a seperate inode and an application can easily identify them and choose not to follow them. You can see all your symbolic links using the following command “find /etc -type l -exec ls -li {} \;”

Hard links on the other hand are much less distinguishable apart from sharing an inode with something else. If you were writing an application how would you determine which hard links to follow and which not too? You can see all files which have a hard link using the following command “find / -links +2 -type f -exec ls -li {} \;”

I am sure the above is not the only reason why this functionality is not used these days but hope this has highlighted the potential flaws and likely why this is no longer used. I would imagine trying to resolve any potential implications of using hard links for directories is negated by the actual lack of demand given that symbolic links handle this task. I am curious are you asking this question for your understanding or do you have a problem that you are trying to overcome?

Hope this helps

∞

**Srinivas**

January 25, 2012, 11:21 am

Thanks Gareth for your excellent explanation. I asked this question out of my curiosity as I never got a satisfactory answer for this question in the past. I did lot

of googleing, but didn't got much.

I knew hard links are very limited use these days and we can achieve similar or more functionality with symlinks.

I agree with you that, hard links are difficult to spot and manage as they point to same inode and share a single entry in the file system.

Although hard links has some advantages over soft links, particular space saving, am not sure how much they really useful in this modern computing.

one advantage even today is having different names for same executable. we are still using hard links in some way without even we knowing

```
$ find /bin -links +2 -exec ls -li {} \;
423498 -rwxr-xr-x 3 root root 62872 Jan 14 2010 /bin/gunzip
423498 -rwxr-xr-x 3 root root 62872 Jan 14 2010 /bin/gzip
423498 -rwxr-xr-x 3 root root 62872 Jan 14 2010 /bin/zcat
```

in above example zcat,gzip,gunzip pointing same executable with different names.

Srinivas

∞

**Musab**

March 6, 2012, 10:36 am

bravo himanshu gr8 wrk my friend
thank you very explainable

∞

**Shilpi**

September 11, 2012, 4:17 am

Its really good tutorial

∞

**harkamal**

November 7, 2012, 10:29 am

when we name a file in unix , where is it’s file name stored ?

∞

**nithin shetty**

August 12, 2013, 11:05 pm

You can remove the file ab* like this

rm -f ab* #by escaping the *

∞

**Erhan S.**

October 3, 2013, 10:08 am

Neat info. thanks!

∞

**Anonymous**

October 5, 2013, 4:56 am

Thanks a lot for the simple and clear explanation!

∞

**Florian**

November 8, 2013, 2:13 pm

Hi Himanshu and thank you for this article.

I was wondering:
How to know the inodes+filenames size of a ramfs?
e.g. I can create 10 million empty files in /ramfs-dir/ and if I type du -b /ramfs-dir/ it will return 0 while in fact, some serious space is used in the ram for these files.

THANKS!

∞

**Naresh**

January 7, 2014, 11:19 am

I observe that the inodes are full; now I am not able to create any file; how can I recover by cleaning this up?

∞

**Shubham**

April 7, 2014, 5:17 am

Very clean , precise and easy to understand. Thumbs up!!

∞

**poorvi**

January 20, 2015, 1:27 am

Pls explain how inodes are allocated with
example

∞

**pritee**

March 3, 2015, 12:01 pm

It’s really simple and easy to understand the concept of inode… Similarly if there are more articles to understand the concept of Perl and Unix then please help…..

∞

**Anonymous**

September 26, 2015, 10:57 pm

Hi Srinivas,
having said the use of Hard links through the example of gzip. can you explain me

why, despite all the files pointing to the same file, why can't they be used interchangeably?

ex) gzip file
results in file.gz
but why can't we use same gzip command to unzip, but instead have to use gunzip, though they are the same.

is the gzip binary written in such a way that, the usage , as in , the command itself is considered as an argument and then the result depends on what command we run ?

∞

**Hani**
September 30, 2015, 11:32 pm

Great explanations and I love the hints you give every time you can, very helpful! Thanks 🙄

∞

**sara arif**
November 3, 2015, 8:04 am

realy helpful thnkyu

∞

**P. M. Verma**
April 8, 2016, 12:35 am

Under Inode Basics,
You have mentioned "The timestamps for file creation".
I read somewhere that Linux does not store file creation time.

∞

**Abhimanyu**
June 8, 2016, 10:55 am

It is a pleasure to go through your website. I am new to this and you made it very easy for me..Thanks so much..

One quick question, when we do ls -al , it shows the files with Inode Info.
what is the first "." signifies.

∞

**Shivangi**
December 24, 2016, 5:59 am

It was really nice explaination. It helped me .

∞

**Udit**
January 9, 2017, 1:28 am

Nice. well expalined.

∞

**Priyabrata Pattnaik**
March 26, 2017, 1:51 am

How to clear the Inode usage. For example when I type the command df -h it shows inode usage is 100%. How to clear the inode space after deleting some files.

∞

Next post: What is an IP Address? IPv4 and IPv6 Address Examples
Previous post: TCP/IP Attacks – ARP Cache Poisoning Fundamentals Explained

RSS | Email | Twitter | Facebook

## EBOOKS

Free

Linux 101 Hacks 2nd Edition eBook - Practical Examples to Build a Strong Foundation in Linux

Bash 101 Hacks eBook - Take Control of Your Bash Command Line and Shell Scripting

Sed and Awk 101 Hacks eBook - Enhance Your UNIX / Linux Life with Sed and Awk

Vim 101 Hacks eBook - Practical Examples for Becoming Fast and Productive in Vim Editor

Nagios Core 3 eBook - Monitor Everything, Be Proactive, and Sleep Well



## POPULAR POSTS

15 Essential Accessories for Your Nikon or Canon DSLR Camera

12 Amazing and Essential Linux Books To Enrich Your Brain and Library

50 UNIX / Linux Sysadmin Tutorials

50 Most Frequently Used UNIX / Linux Commands (With Examples)

How To Be Productive and Get Things Done Using GTD

30 Things To Do When you are Bored and have a Computer

Linux Directory Structure (File System Structure) Explained with Examples

Linux Crontab: 15 Awesome Cron Job Examples

Get a Grip on the Grep! – 15 Practical Grep Command Examples

Unix LS Command: 15 Practical Examples

15 Examples To Master Linux Command Line History

Top 10 Open Source Bug Tracking System

Vi and Vim Macro Tutorial: How To Record and Play

Mommy, I found it! -- 15 Practical Linux Find Command Examples

15 Awesome Gmail Tips and Tricks

15 Awesome Google Search Tips and Tricks

RAID 0, RAID 1, RAID 5, RAID 10 Explained with Diagrams

Can You Top This? 15 Practical Linux Top Command Examples

Top 5 Best System Monitoring Tools

Top 5 Best Linux OS Distributions

How To Monitor Remote Linux Host using Nagios 3.0

Awk Introduction Tutorial – 7 Awk Print Examples

How to Backup Linux? 15 rsync Command Examples

The Ultimate Wget Download Guide With 15 Awesome Examples

Top 5 Best Linux Text Editors

Packet Analyzer: 15 TCPDUMP Command Examples

The Ultimate Bash Array Tutorial with 15 Examples

3 Steps to Perform SSH Login Without Password Using ssh-keygen & ssh-copy-id

Unix Sed Tutorial: Advanced Sed Substitution Examples

UNIX / Linux: 10 Netstat Command Examples

The Ultimate Guide for Creating Strong Passwords

6 Steps to Secure Your Home Wireless Network

Turbocharge PuTTY with 12 Powerful Add-Ons

## CATEGORIES

Linux Tutorials

Vim Editor

Sed Scripting

Awk Scripting

Bash Shell Scripting

Nagios Monitoring

OpenSSH

IPTables Firewall

Apache Web Server

MySQL Database

Perl Programming

Google Tutorials

Ubuntu Tutorials

PostgreSQL DB

Hello World Examples

C Programming

C++ Programming

DELL Server Tutorials

Oracle Database

VMware Tutorials

## About The Geek Stuff

My name is **Rameshan Natarajan**. I will be posting instruction guides, how-to, troubleshooting tips and tricks on Linux, database, hardware, security and web. My focus is to write articles that will either teach you or help you resolve a problem. Read more about Ramesh Natarajan and the blog.

## Contact Us

**Email Me :** Use this Contact Form to get in touch me with your comments, questions or suggestions about this site. You can also simply drop me a line to say hello!.

Follow us on Twitter

Become a fan on Facebook

## Support Us

Support this blog by purchasing one of my ebooks.

Bash 101 Hacks eBook

Sed and Awk 101 Hacks eBook

Vim 101 Hacks eBook

Nagios Core 3 eBook