Linux 文件时间戳解释：atime、mtime 和 ctime

什么时候“改变”不意味着“修改”？当我们谈论 Linux 文件时间戳时。在本指南中，我们将解释系统如何更新它们，以及如何自行更改它们。

atime、mtime 和 ctime 之间的区别

每个 Linux 文件都有三个时间戳：访问时间戳 (atime)、修改时间戳 (mtime) 和更改时间戳 (ctime)。

访问时间戳是最后一次读取文件的时间。这意味着有人使用程序来显示文件的内容或从中读取一些值。没有任何内容被编辑或添加到文件中。数据被引用但未更改。

修改后的时间戳表示上次修改文件内容的时间。程序或进程编辑或操作了文件。 “修改”表示文件中的某些内容被修改或删除，或者添加了新数据。

更改的时间戳不是指对文件内容所做的更改。相反，它是与文件相关的元数据发生更改的时间。例如，文件权限更改将更新更改的时间戳。

标准的 ext4 Linux 文件系统还在其内部文件系统结构中为文件创建时间戳分配空间，但这尚未实现。有时，此时间戳已填充，但您不能依赖其中的值。

时间戳剖析

Linux 时间戳包含一个数字而不是日期和时间。该数字是自 Unix 纪元以来的秒数，即 1970 年 1 月 1 日午夜 (00:00:00)，协调世界时 (UTC)。 Linux 时间戳中忽略了闰秒，因此它们与实时时间不同。

当 Linux 需要显示时间戳时，它会将秒数转换为日期和时间。这使得人类更容易理解。查看文件的计算机所在的位置和时区指导将秒数转换为日期和时间。它还确保月份使用正确的语言。

那么，一个时间戳可以存储多少秒呢？很多——准确地说是 2,147,483,647。这是一个很大的数字，但足够了吗？如果将其添加到 Unix 纪元，然后将其转换为日期和时间，则会得到 2038 年 1 月 19 日，星期二，凌晨 03:14:07。不过，在此之前我们需要一个不同的时间戳方案。

查看时间戳

当您将`-l`（长列表）选项与`ls`一起使用时，如下所示，您可以看到修改后的时间戳：

```
ls -l dp.c
```

如果要查看访问时间戳，请使用 `-lu`（访问时间）选项，如下所示：

```
ls -lu dp.c
```

最后，要查看更改时间戳，您可以使用 `-lc`（更改时间）选项；输入以下内容：

```
ls -lc dp.c
```

```
dave@howtogeek:~/work$ ls -l dp.c
-rwxrw-r-x 1 dave dave 2895 Apr 21  2019 dp.c
dave@howtogeek:~/work$
dave@howtogeek:~/work$ ls -lu dp.c
-rwxrw-r-x 1 dave dave 2895 Jan 20 11:30 dp.c
dave@howtogeek:~/work$
dave@howtogeek:~/work$ ls -lc dp.c
-rwxrw-r-x 1 dave dave 2895 Jan 20 11:30 dp.c
dave@howtogeek:~/work$
```

上面的时间戳显示文件的内容最后一次修改是在 2019 年 4 月 21 日。访问和更改的时间戳是相同的，因为文件是在 2020 年 1 月 20 日从另一台计算机复制到这台计算机上的，并且两个时间戳都是在那个时候更新的。

要同时查看所有时间戳，请使用 `stat` 命令，如下所示：

```
stat dp.c
```

```
dave@howtogeek:~/work$ stat dp.c
  File: dp.c
  Size: 2895            Blocks: 8          IO Block: 4096   regular fi
le
Device: 801h/2049d      Inode: 1581292     Links: 1
Access: (0765/-rwxrw-r-x)  Uid: ( 1000/    dave)   Gid: ( 1000/    dav
e)
Access: 2020-01-20 11:30:30.959001000 -0500
Modify: 2019-04-21 09:13:35.510830000 -0400
Change: 2020-01-20 11:30:30.959001000 -0500
 Birth: -
dave@howtogeek:~/work$
```

时区列在显示屏底部。如您所见，它们具有非常准确的小数秒组件。在每个时间戳的末尾，您还会看到 `-0500` 或 `-0400`。

这些是*时区偏移量*。文件系统以 UTC 格式记录时间戳，并在 `stat` 显示时将其转换为本地时区。我们用来研究这篇文章的计算机的配置就像它在美国东部标准时间 (EST) 区一样。

当 EST 生效时，该时区比 UTC 晚五个小时。然而，当东部夏令时 (EDT) 生效时，它比 UTC 晚四个小时。 2019 年 4 月，更改修改时间戳时，EDT 生效。这就是为什么其中两个时间戳有 5 小时的偏移量，而修改后的有 4 小时的偏移量。

偏移量和时区不会存储在任何地方。既没有 inode 也没有文件系统空间专门用于保存这些值。您必须使用时间戳（始终采用 UTC 时间）、显示文件的计算机的本地时区以及 DST 是否生效来即时计算这些。

您还会看到一个“出生”时间戳，这是为文件的创建日期保留的。这未实现，您会看到一个连字符“-”而不是时间戳。

更改时间戳

如果需要，您可以更改文件上的时间戳。您可以使用 `touch` 命令更改访问时间戳或修改时间戳，或同时更改两者：

```
touch -a dp.c
```

要设置新的访问时间戳，您可以使用 `-a`（访问时间）选项。此命令将访问时间戳设置为计算机的当前时间：

```
stat dp.c
```

```
dave@howtogeek:~/work$ touch -a dp.c
dave@howtogeek:~/work$ stat dp.c
  File: dp.c
  Size: 2895            Blocks: 8          IO Block: 4096   regular fi
le
Device: 801h/2049d      Inode: 1581292     Links: 1
Access: (0765/-rwxrw-r-x)  Uid: ( 1000/    dave)   Gid: ( 1000/    dav
e)
Access: 2020-01-22 08:43:36.712000000 -0500
Modify: 2019-04-21 09:13:35.510830000 -0400
Change: 2020-01-22 08:43:36.712000000 -0500
 Birth: -
dave@howtogeek:~/work$
```

访问时间戳已按预期更改。但是，更改的时间戳也已更新；这个是正常的。

要更改修改时间戳，可以使用 `-m`（修改时间）选项：

```
touch -m dp.c
```

```
stat dp.c
```

```
dave@howtogeek:~/work$ touch -m dp.c
dave@howtogeek:~/work$ stat dp.c
  File: dp.c
  Size: 2895            Blocks: 8          IO Block: 4096   regular fi
le
Device: 801h/2049d      Inode: 1581292     Links: 1
Access: (0765/-rwxrw-r-x)  Uid: ( 1000/    dave)   Gid: ( 1000/    dav
e)
Access: 2020-01-22 08:50:55.080000000 -0500
Modify: 2020-01-22 08:52:43.332000000 -0500
Change: 2020-01-22 08:52:43.332000000 -0500
 Birth: -
dave@howtogeek:~/work$
```

这一次，更新了修改和更改的时间戳。

如果要同时更改访问时间戳和修改时间戳，可以使用 `-d`（日期）选项。您还可以指定时间和日期——您不仅限于将时间戳更改为现在。

我们将使用以下命令将访问和修改时间戳设置为 2020 年 1 月 15 日的 10:30:45：

```
touch -d "2020-01-15 10:30:45" dp.c
```

```
stat dp.c
```

```
dave@howtogeek:~/work$ touch -d "2020-01-15 10:30:45" dp.c
dave@howtogeek:~/work$ stat dp.c
  File: dp.c
  Size: 2895            Blocks: 8          IO Block: 4096   regular fi
le
Device: 801h/2049d      Inode: 1581292     Links: 1
Access: (0765/-rwxrw-r-x)  Uid: ( 1000/    dave)   Gid: ( 1000/    dav
e)
Access: 2020-01-15 10:30:45.000000000 -0500
Modify: 2020-01-15 10:30:45.000000000 -0500
Change: 2020-01-22 09:00:07.020000000 -0500
 Birth: -
dave@howtogeek:~/work$
```

我们现在已经将访问和修改时间戳设置为过去的日期。更改的时间戳也更新为计算机的当前时间。

如果要将一个文件的时间戳设置为另一个文件的时间戳值，也可以使用 `-r`（参考）选项，如下所示：

```
touch dp.c -r dice_words.sl3
```

```
stat dp.c
```

```
dave@howtogeek:~/work$ touch dp.c -r dice_words.sl3
dave@howtogeek:~/work$ stat dp.c
  File: dp.c
  Size: 2895            Blocks: 8          IO Block: 4096   regular fi
le
Device: 801h/2049d      Inode: 1581292     Links: 1
Access: (0765/-rwxrw-r-x)  Uid: ( 1000/    dave)   Gid: ( 1000/    dav
e)
Access: 2020-01-22 07:30:16.416000000 -0500
Modify: 2019-03-20 10:15:57.602387000 -0400
Change: 2020-01-22 09:04:42.212000000 -0500
 Birth: -
dave@howtogeek:~/work$
```

然后，我们几乎回到了起点，混合了 `-0400` 和 `-0500` 时间戳。

让我们做一些只影响更改的时间戳的事情。我们将使用 `chmod` 命令为所有用户授予可执行文件执行权限：

```
chmod +x dp
```

```
stat dp
```

```
dave@howtogeek:~/work$ chmod +x dp
dave@howtogeek:~/work$ stat dp
  File: dp
  Size: 20920           Blocks: 48         IO Block: 4096   regular fi
le
Device: 801h/2049d      Inode: 1581291     Links: 1
Access: (0755/-rwxr-xr-x)  Uid: ( 1000/    dave)   Gid: ( 1000/    dav
e)
Access: 2020-01-22 08:30:15.000000000 -0500
Modify: 2020-01-22 08:30:15.000000000 -0500
Change: 2020-01-22 09:20:43.148000000 -0500
 Birth: -
dave@howtogeek:~/work$
```

更改的时间戳是唯一更新的时间戳。这是因为文件本身没有改变——既没有被访问也没有被修改。但是，有关文件的元数据*已*更改。

文件系统如何更新时间戳

挂载文件系统时，可以使用一些选项来指定该文件系统应如何操作或处理。这些存储在 `/etc/fstab` 文件中，该文件在启动时被读取和处理。您还可以设置选项来指示他们应该用来更新访问时间戳的方案。

以下是一些最常见的选项：

- **strictatime (strict atime)**：此选项会在每次访问文件时更新文件的访问时间戳。这种方法会产生开销，但某些服务器可以从这种方案中受益。它在台式机或笔记本电脑上没有什么优点。
- **noatime （无 atime）**：此选项完全禁止更新文件和目录的访问时间戳。但是，修改后的时间戳仍会更新。
- **nodiratime（无目录时间）**：此选项为要更新的文件启用访问时间戳，但为目录禁用它。
- **relatime（相对时间）**：只有当访问时间戳超过 24 小时，或者前一个早于当前修改或更改的时间戳时，此选项才会更新访问时间戳。这在访问时间戳更新过于频繁或根本不更新之间取得了良好的平衡。

让我们看看这台计算机的 `/etc/fstab` 文件，看看设置了哪些选项：

```
less /etc/fstab
```

```
dave@howtogeek:~/work$ less /etc/fstab
```

`/etc/fstab`文件给我们显示出来了，如下图。

```
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name dev
ices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda1 during installation
UUID=4a143d08-8695-475b-8243-b13b56050fc2 /               ext4    erro
rs=remount-ro 0       1
/swapfile                                 none            swap    sw
            0       0
/etc/fstab (END)
```

这是没有环绕的文件内容：

```
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda1 during installation
UUID=4a143d08-8695-475b-8243-b13b56050fc2 / ext4 errors=remount-ro 0 1
/swapfile none swap sw 0 0
```

只有两个条目，其中一个是交换文件，我们可以忽略它。另一个安装在文件系统的根目录（ / ）并且在安装时位于设备 `/dev/sda1` 上。这是第一个硬盘驱动器上的第一个分区，它恰好包含一个 `ext4` 文件系统。

传递给它的唯一选项是 `errors=remount-ro`，它告诉操作系统在尝试将其挂载为读写文件系统时出现错误时将其重新挂载为只读文件系统.

因此，没有提及如何处理访问时间戳。让我们深入挖掘，看看 `/proc/mounts` 能告诉我们什么。我们将从 `/proc/mounts` 输出到 `grep`。我们的搜索字符串将是“sda”，即硬盘驱动器标识符。

我们键入以下内容：

```
cat /proc/mounts | grep "sda"
```

```
dave@howtogeek:~/work$ cat /proc/mounts | grep "sda"
/dev/sda1 / ext4 rw,relatime,errors=remount-ro 0 0
dave@howtogeek:~/work$
```

现在我们看到以下选项：

- **rw**：文件系统将挂载为读写文件系统。
- **relatime**：文件系统将使用“相对时间”方案来更新访问时间戳。

那个是从哪里来的？那么，`relatime` 方案用于以下情况：

- 当使用**defaults** `/etc/fstab` 选项时。
- 当使用**relatime** `/etc/fstab` 选项时。
- `/etc/fstab` 中没有使用访问时间戳选项，并且您使用的是 Linux 内核 2.6.30 或更新版本。

`ext4` 文件系统的 `/etc/fstab` 条目没有指定任何访问时间戳更新选项，因此 Linux 做出了明智的选择并使用了 `relatime`。

时间戳很重要

时间戳为我们提供了一种查看文件何时被访问、修改或更改的简便方法。但是，更重要的是，它们提供了一种备份和同步软件的方法，以确定哪些文件需要备份。

每当您需要强行说服程序包含或忽略一个文件或一组文件时，操纵时间戳的能力将被证明是有用的。

Linux Commands	
Files	tar • pv • cat • tac • chmod • grep • diff • sed • ar • man • pushd • popd • fsck • testdisk • seq • fd • pandoc • cd • $PATH • awk • join • jq • fold • uniq • journalctl • tail • stat • ls • fstab • echo • less • chgrp • chown • rev • look • strings • type • rename • zip • unzip • mount • umount • install • fdisk • mkfs • rm • rmdir • rsync • df • gpg • vi • nano • mkdir • du • ln • patch • convert • rclone • shred • srm • scp • gzip • chattr • cut • find • umask • wc
Processes	alias • screen • top • nice • renice • progress • strace • systemd • tmux • chsh • history • at • batch • free • which • dmesg • chfn • usermod • ps • chroot • xargs • tty • pinky • lsof • vmstat • timeout • wall • yes • kill • sleep • sudo • su • time • groupadd • usermod • groups • lshw • shutdown • reboot • halt • poweroff • passwd • lscpu • crontab • date • bg • fg • pidof • nohup • pmap
Networking	netstat • ping • traceroute • ip • ss • whois • fail2ban • bmon • dig • finger • nmap • ftp • curl • wget • who • whoami • w • iptables • ssh-keygen • ufw • arping • firewalld

RELATED: ***Best Linux Laptops for Developers and Enthusiasts***