Linux Professional Institute

LPIC-1

جلسه هفتم: آرشیو کردن فایل‌ها، مدیریت لینک‌ها و مجوز فایل‌ها

در این جلسه:

ویدئو اول:

- یادآوری دستور tar
- آشنای با دستور cpio
- آشنایی با دستور dd
- آشنایی با هاردلینک و سافت‌لینک
- آشنایی با مفهوم مالکیت فایل‌ها و دایرکتوری‌ها
- آشنایی با چگونگی تغییر مالک فایل و دایرکتوری با دستور chown

ویدئو دوم:

- آشنایی با چگونگی تغییر گروه فایل و دایرکتوری با دستور chgrp
- آشنایی با مفهوم مجوزها
- آشنایی با مجوزهای ویژه
- آشنایی با چگونگی تغییر مجوز با دستور chmod
- آشنایی با چگونگی تغییر مجوزهای پیش‌فرض با دستور umask

فهرست مطالب

مقدمه

جلسه‌ی قبل ابتدا در مورد پارتیشن‌بندی هارددیسک‌ها و استفاده از دستورهای مربوط به مدیریت فایل‌ها و دایرکتوری‌ها صحبت کردیم. سپس در مورد دلیل نیاز به فشرده‌سازی اطلاعات صحبت کردیم و با برخی از ابزارهای فشرده‌سازی در لینوکس آشنا شدیم. در نهایت، در مورد مفهوم آرشیو صحبت کردیم و چگونگی آرشیو فایل‌ها و دایرکتوری‌ها با ابزار `tar` را یاد گرفتیم. در این جلسه با چندین ابزار دیگر برای آرشیو فایل‌ها آشنا می‌شویم و سپس در مورد مدیریت مجوزهای دسترسی در لینوکس صحبت می‌کنیم.

کپی و آرشیو فایل‌ها با استفاده از *cpio*

ابزار `cpio` از نظر مفهومی بسیار شبیه به ابزار `tar` می‌باشد. یعنی ما با استفاده از این ابزار می‌توانیم چندین فایل را در قالب یک فایل، که به آن فایل آرشیو می‌گوییم، ذخیره کنیم. ابزار `cpio` که مخفف copy in and out می‌باشد، یکی از ابزارهای بسیار قوی برای ایجاد آرشیوها می‌باشد و در بک‌آپ‌گیری از کل سیستم، از آن استفاده می‌شود. این ابزار، می‌تواند خروجی خود را بر روی یک دایرکتوری یا بر روی یک دستگاه جانبی بریزد (دقیقا مانند `tar`). قبل از این که در مورد چگونگی اسفاده از `cpio` صحبت کنیم، بیایید ابتدا یک سری فایل ساختگی ایجاد کنیم. اگر به خاطر داشته باشید، دستور `touch` می‌توانست فایل‌های ساختگی ایجاد کند. پس:

```
[root@localhost ~]# touch File1.txt File2.txt File3.txt
[root@localhost ~]# ls -1
File1.txt
File2.txt
File3.txt
```

همانطور که می‌بینید، ما با استفاده از `touch`، سه فایل خالی با نام‌های مشخص شده ایجاد کردیم. حال وقت آن رسیده که به سراغ کار با ابزار `cpio` برویم. توجه داشته باشید که ابزار `cpio` از نظر ساختاری، کمی متفاوت از دستورهایی که تا الان یاد گرفتیم می‌باشد و ممکن است در ابتدا طریقه‌ی استفاده از این دستور کمی شما را گیج کند.

برای ایجاد یک فایل آرشیو توسط `cpio`، ابتدا باید لیستی از فایل‌هایی که میخواهیم آرشیو کنیم را درون `cpio` پایپ کرده و سپس با استفاده از redirectorها، خروجی `cpio` را درون یک فایل بریزیم:

```
[root@localhost ~]# ls -1 | cpio -ov > file.cpio
File1.txt
File2.txt
File3.txt
file.cpio
1 block
```

همانطور که می‌بینید، از آنجا که ما می‌خواستیم فایل‌هایی که به تازگی ایجاد کردیم را آرشیو کنیم، ابتدا با استفاده از `ls`، لیستی از فایل‌های مورد نظر را به دست آوردیم. سپس خروجی `ls` را درون دستور `cpio` پایپ کردیم و بدین طریق، به `cpio` گفتیم که چه فایل‌هایی را درون آرشیو قرار دهد. آپشن `o` به `cpio` می‌گوید که باید یک فایل آرشیو ایجاد کند و آپشن `v` به این دستور می‌گوید که گزارشی از عملکرد خود را در خروجی به ما ارائه دهد.

بیایید از صحت ایجاد فایل آرشیو اطمینان حاصل کنیم:

```
[root@localhost ~]# ls -l
total 4
-rw-r--r--. 1 root root   0 Jul 11 10:24 File1.txt
-rw-r--r--. 1 root root   0 Jul 11 10:24 File2.txt
-rw-r--r--. 1 root root   0 Jul 11 10:24 File3.txt
-rw-r--r--. 1 root root 512 Jul 11 10:29 file.cpio
```

همانطور که می‌بینید، فایل `file.cpio` ایجاد شده و در این محل قرار گرفته است. اما از کجا می‌توانیم مطمئن باشیم که فایل‌های ما به درستی درون این فایل آرشیو قرار گرفته‌اند؟

ما می‌توانیم محتویات یک فایل آرشیو `cpio` را بدون اکسترکت کردن مشاهده کنیم. برای این کار:

```
[root@localhost ~]# cpio -itvI file.cpio
-rw-r--r--   1 root     root            0 Jul 11 10:24 File1.txt
-rw-r--r--   1 root     root            0 Jul 11 10:24 File2.txt
-rw-r--r--   1 root     root            0 Jul 11 10:24 File3.txt
-rw-r--r--   1 root     root            0 Jul 11 10:29 file.cpio
1 block
```

همانطور که می‌بینید، با استفاده از آپشن‌های `itvI`، برنامه‌ی `cpio` محتویات موجود در فایل آرشیو را بدون اکسترکت کردن به ما نشان داد. آپشن `i` هنگام قرار گرفتن در کنار `t`، به `cpio` می‌گوید که محتویات فایل آرشیو را نمایش دهد. آپشن `t` سبب می‌شود که محتویات آرشیو در خروجی برایمان لیست شود. آپشن `v` سبب می‌شود که محتویات آرشیو با اطلاعات بیشتری نظیر زمان ایجاد، صاحب فایل و... به ما نمایش داده شود و در نهایت آپشن `I` جهت معرفی فایل آرشیوی که باید محتویاتیش به ما نشان داده شود، به کار می‌رود.

بیایید یک فایل آرشیو دیگر نیز با `cpio` ایجاد کنیم. این بار، بیایید کلیه‌ی فایل‌هایی که در دایرکتوری `/etc` وجود دارند و دارای پسوند `.conf` هستند را آرشیو کنیم. برای این کار:

```
[root@localhost ~]# ls /etc/*.conf
/etc/asound.conf  /etc/ld.so.conf      /etc/mke2fs.conf    /etc/sudo.conf
/etc/dracut.conf  /etc/libaudit.conf   /etc/nsswitch.conf  /etc/sudo-ldap.conf
/etc/e2fsck.conf  /etc/libuser.conf    /etc/resolv.conf    /etc/sysctl.conf
/etc/host.conf    /etc/locale.conf     /etc/rsyncd.conf    /etc/tcsd.conf
/etc/kdump.conf   /etc/logrotate.conf  /etc/rsyslog.conf   /etc/vconsole.conf
/etc/krb5.conf    /etc/man_db.conf     /etc/sestatus.conf  /etc/yum.conf
[root@localhost ~]# ls /etc/*.conf | cpio -ov > etc.cpio
...
/etc/sudo-ldap.conf
/etc/sysctl.conf
/etc/tcsd.conf
/etc/vconsole.conf
/etc/yum.conf
77 blocks
```

همانطور که می‌بینید ما با استفاده از globbing، از `ls` خواستیم که کلیه‌ی فایل‌های موجود در مسیر `/etc` که دارای پسوند `.conf` هستند را به ما نمایش دهد. سپس خروجی دستور `ls` را درون `cpio` پایپ کردیم و با استفاده از آپشن‌های `ov` به این دستور گفتیم که برای ما یک فایل آرشیو ایجاد کند. سپس با استفاده از ریدایرکتور `>`، به `cpio` گفتیم که STDOUT خود را درون فایل `etc.cpio` بریزد.

حال بیایید به محتویات فایل آرشیو شده توسط `cpio` نگاهی بیاندازیم:

```
[root@localhost ~]# cpio -itvI etc.cpio
...
-rw-r--r--   1 root     root          449 Aug  9  2019 /etc/sysctl.conf
-rw-------   1 tss      tss          7046 Aug  3  2017 /etc/tcsd.conf
```

```
-rw-r--r--   1 root     root           37 Mar 20 10:47 /etc/vconsole.conf
-rw-r--r--   1 root     root          970 Aug  8  2019 /etc/yum.conf
77 blocks
```

همانطور که می‌بینید، cpio محتویات مورد نظر ما را به درستی آرشیو کرده است. اگر با دقت بیشتری به خروجی نگاه کنید، می‌بینید که cpio هنگام آرشیو کردن، Absolute Path هر فایل را نیز نگهداری می‌کند. به عبارت دیگر، اگر ما این فایل آرشیو را اکسترکت کنیم، محتویات این فایل، دقیقا در دایرکتوری /etc اکسترکت می‌شوند. این امر باعث می‌شود که در بسیاری از مواقع، از ابزار cpio جهت ایجاد یک image یا بک‌آپ کامل از سیستم استفاده کنیم.

حال بیایید در مورد چگونگی اکسترکت کردن یک فایل cpio صحبت کنیم. جهت اکسترکت کردن یک آرشیو cpio، به شکل زیر عمل می‌کنیم:

```
[root@localhost ~]# cpio -ivI etc.cpio
...
cpio: /etc/vconsole.conf not created: newer or same age version exists
/etc/vconsole.conf
cpio: /etc/yum.conf not created: newer or same age version exists
/etc/yum.conf
77 blocks
```

همانطور که می‌بینید، در حال حاضر این دستور به ما پیغام خطا می‌دهد. قبل از صحبت در مورد پیغام خطا، بیایید آپشن‌های به کار رفته را بررسی کنیم. آپشن i به cpio می‌گوید که فایل را اکسترکت کند (چون اینجا آپشن t نداریم). آپشن v سبب می‌شود که این دستور گزارشی از عملکرد خود در خروجی را به ما بدهد و با استفاده از آپشن I، فایل آرشیوی که باید اکسترکت شود را به cpio می‌دهیم.

حال بیایید در مورد پیغام خطا صحبت کنیم. همانطور که گفتیم، آرشیوهای cpio، مسیر Absolute Path هر فایل را درون خود نگهداری می‌کنند. یعنی در اینجا، فایلی نظیر /etc/yum.conf، دقیقا در مسیر /etc/yum.conf اکسترکت خواهد شد. این، بر خلاف عملکرد ابزاری مانند tar می‌باشد که فایل‌ها و دایرکتوری‌های آرشیو شده را در مسیر کنونی اکسترکت می‌کرد. دلیل این خطا، این است که فایل‌های موجود در /etc/*.conf، هیچ تغییری در مقایسه با فایل‌های موجود در آرشیو نکرده‌اند و در نتیجه، cpio فایل‌های آرشیو شده را جایگزین آن فایل‌ها نمی‌کند.

برای این که cpio فایل‌های درون خود را جایگزین فایل‌های موجود در /etc/ کند، می‌توانیم آپشن u- را به این دستور اضافه کنیم:

```
[root@localhost ~]# cpio -iuvI etc.cpio
...
/etc/sudo-ldap.conf
/etc/sysctl.conf
/etc/tcsd.conf
/etc/vconsole.conf
/etc/yum.conf
77 blocks
```

همانطور که می‌بینید، این بار به ما پیغام خطایی داده نشد و cpio فایل‌های موجود در آرشیو را جایگزین فایل‌های اصلی موجود در /etc کرد.

البته ما می‌توانیم از cpio بخواهیم که بدون توجه به Absolute Path، فایل‌های موجود درون آرشیو را در مکان کنونی اکتسرکت کند. برای این کار باید آپشن --no-absolute-filenames را به این دستور اضافه کنیم:

```
[root@localhost ~]# cpio -idvI  etc.cpio --no-absolute-filenames
cpio: Removing leading `/' from member names
...
etc/vconsole.conf
etc/yum.conf
77 blocks
```

همانطور که می‌بینید، ما به آپشن -ivI که از آن جهت اکتسرکت فایل‌ها استفاده می‌کردیم، دو آپشن جدید اضافه کردیم. آپشن --no-absolute-filenames به cpio می‌گوید که با حذف علامت / موجود در ابتدای نام فایل‌های موجود درون آرشیو، آنها را از حالت Absolute Path بودن خارج کند. یعنی در اینجا، فایل‌ها به جای اکتسرکت شدن در /etc، در etc (یعنی یک دایرکتوری در موقعیت کنونی با نام etc) اکتسرکت خواهند شد. از آنجایی که ما در دایرکتوری کنونی، دایرکتوری به نام etc نداریم، از آپشن d نیز استفاده کردیم. این آپشن به cpio می‌گوید که در صورت عدم وجود دایرکتوری جهت اکتسرکت، آن دایرکتوری را ایجاد کند.

در نهایت، لازم است عنوان کنیم که بر خلاف tar، دستور cpio نمی‌تواند به صورت اتوماتیک از ابزارهای فشرده‌سازی استفاده کند و ما مجبوریم خودمان خروجی cpio را درون ابزارهای فشرده‌سازی پایپ کنیم. ما در جلسه‌ی قبل به صورت کامل در مورد ابزارهای فشرده‌سازی و چگونگی استفاده از آنها صحبت کردیم، اما به طور کلی:

```
[root@localhost ~]# ls -1 File?.txt | cpio -ov | gzip -v > comp-file.cpio.gz
File1.txt
File2.txt
File3.txt
1 block
 86.3%
[root@localhost ~]# ls -1
comp-file.cpio.gz
File1.txt
File2.txt
File3.txt
```

همانطور که می‌بینید، طبق معمول ما لیستی از فایل‌هایی که می‌خواهیم آرشیو کنیم را با استفاده از دستور ls، درون cpio پایپ کردیم و چون می‌خواستیم یک فایل آرشیو ایجاد کنیم، از آپشن‌های -ov استفاده کردیم. سپس به جای ریدایرکت کردن خروجی cpio به یک فایل، خروجی آن را درون دستور gzip جهت فشرده‌سازی فایل آرشیو، پایپ کردیم و در نهایت با استفاده از ریدایرکتور >، خروجی gzip را درون یک فایل به نام comp-file.cpio.gz ریختیم. این فایل، یک فایل آرشیو cpio می‌باشد که توسط gzip فشرده‌سازی شده است.

برای این که این فایل را از حالت فشرده خارج کنیم و سپس آن را اکسترکت کنیم، می‌توانیم به صورت زیر عمل کنیم:

```
[root@localhost ~]# gunzip -c comp-file.cpio.gz | cpio -iuv
File1.txt
File2.txt
File3.txt
1 block
```

همانطور که می‌بینید، ما ابتدا با استفاده از دستور gunzip، اقدام به خارج کردن فایل از حالت فشرده کردیم. با استفاده از آپشن -c، به gunzip گفتیم که خروجی خود را روی STDOUT بریزد، سپس ما خروجی gunzip

را داخل برنامه‌ی cpio پایپ کردیم. آپشن‌های iuv را قبلا توضیح دادیم پس به توضیح بیشتر آن نمی‌پردازیم.
توجه کنید که ما مجبور به استفاده از gzip نیستیم و می‌توانیم از bzip2 یا xz نیز استفاده کنیم.

تکثیر با استفاده از *dd*

ابزار dd به ما امکان می‌دهد که از کلیه‌ی داده‌های موجود روی یک دستگاه ذخیره‌سازی، بک‌آپ بگیریم. از این ابزار، معمولا برای ایجاد کپی‌های Low-Level از دستگاه‌های ذخیره‌سازی (هارددیسک، یک پارتیشن روی هارددیسک و...) استفاده می‌شود. به عبارت دیگر، این ابزار یک المثنی از یک دستگاه ذخیره‌سازی ایجاد می‌کند؛ این یعنی هنگام تکثیر یک پارتیشن، نه تنها فایل‌های موجود درون آن را تکثیر می‌کند، بلکه کلیه‌ی فضای خالی آن و حتی فایل‌سیستم آن را نیز تکثیر می‌کند. به طور کلی، از دستور dd به صورت زیر استفاده می‌کنیم:

```
dd if=INPUT_DEVICE of=OUTPUT_DEVICE [operands]
```

همانطور که می‌بینید، برای استفاده از این دستور، کافی است در مقابل if، دیوایس‌فایل دستگاهی که میخواهیم از آن المثنی ایجاد کنیم را مشخص و در مقابل of، دستگاه یا مکانی که می‌خواهیم المثنی روی آن قرار گیرد را مشخص کنیم.

بیایید با استفاده از dd، یک کپی از پارتیشن /boot سیستم خود ایجاد کنیم. برای این کار، باید دیوایس‌فایل پارتیشن /boot را به dd بدهیم. پس:

```
[root@localhost ~]# lsblk
NAME            MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda               8:0    0   20G  0 disk
├─sda1            8:1    0    1G  0 part /boot
└─sda2            8:2    0   19G  0 part
  ├─centos-root 253:0    0   17G  0 lvm  /
  └─centos-swap 253:1    0    2G  0 lvm  [SWAP]
sdb               8:16   1  1.9G  0 disk /mnt
sr0              11:0    1  942M  0 rom
```

همانطور که می‌بینید، دیوایس‌فایل پارتیشن /boot در موقعیت /dev/sda1 قرار دارد. پس برای ایجاد یک المثنی از این پارتیشن:

```
[root@localhost ~]# dd if=/dev/sda1 of=boot-backup
2097152+0 records in
2097152+0 records out
1073741824 bytes (1.1 GB) copied, 13.2951 s, 80.8 MB/s
```

همانطور که می‌بینید، ما ابتدا دستور dd را وارد کرده، سپس در مقابل if، موقعیت دستگاهی که میخواهیم از آن المثنی ایجاد کنیم را وارد کردیم و سپس در مقابل of، نام فایل المثنی که باید توسط dd ایجاد شود را مشخص کردیم. از آنجایی که ما میخواستیم این فایل در موقعیت کنونی ایجاد شود، در مقابل of، فقط نام فایل المثنی را نوشتیم. اگر میخواستیم این المثنی در یک دستگاه یا دایرکتوری دیگر قرار گیرد، کافی بود مسیر آن دستگاه یا دایرکتوری را به همراه نام فایل المثنی، مشخص می‌کردیم.

حال بیایید از صحت ایجاد این المثنی مطمئن شویم:

```
[root@localhost ~]# ls -lh
total 1.0G
-rw-r--r--. 1 root root 1.0G Jul 13 12:44 boot-backup
```

همانطور که می‌بینید، ما الان یک فایل با نام boot-backup داریم که دقیقا هم‌حجم با پارتیشن /boot می‌باشد.

همانطور که قبلا گفتیم، دستور dd فضای خالی پارتیشن‌ها را نیز کپی می‌کند و دلیل هم حجم بودن فایل المثنی با پارتیشن /boot، همین امر می‌باشد.

حال بیایید ببینیم این فایل از چه نوعی می‌باشد. در جلسه‌ی قبل گفتیم که با استفاده از دستور file، می‌توانیم نوع یک فایل را پیدا کنیم. پس:

```
[root@localhost ~]# file boot-backup
boot-backup: SGI XFS filesystem data (blksz 4096, inosz 512, v2 dirs)
```

همانطور که می‌بینید، دستور file به ما می‌گوید که فایل boot-backup، یک فایل دارای فایل‌سیستم XFS می‌باشد که سایز هر بلوک موجود در آن، ٤٠٩٦ بایت می‌باشد. اگر به خاطر داشته باشید، گفتیم که dd فایل‌سیستم یک پارتیشن را نیز کپی می‌کند، و دلیل این که این فایل یک فایل‌سیستم دارد، این امر می‌باشد.

اگر توجه کرده باشید، وقتی که از dd استفاده می‌کنیم، سیستم گزارشی از عملکرد خود را در خروجی به ما نشان نمی‌دهد و به مدت زمان زیادی، کنترل سیستم را از دست ما میگیرد. اگر به سراغ کپی کردن دستگاه‌های دارای حجم بالا برویم، این امر کمی آزاردهنده می‌شود. پس بیایید این مشکل را حل کنیم.

برای حل این مشکل، کافی است پس از مشخص کردن ورودی و خروجی، از آپشن status به صورت زیر استفاده کنیم:

```
[root@localhost ~]# dd if=/dev/sdb of=AnotherBackup status=progress
2028851712 bytes (2.0 GB) copied, 166.526302 s, 12.2 MB/s
3964928+0 records in
3964928+0 records out
2030043136 bytes (2.0 GB) copied, 166.649 s, 12.2 MB/s
```

همانطور که می‌بینید، ما این بار سعی کردیم از پارتیشن /dev/sdb که دیوایس‌فایل یک فلش‌مموری بود المثنی ایجاد کنیم. این بار پس از مشخص کردن نام فایل خروجی، آپشن status را اضافه کردیم و در مقابل آن عبارت progress را نوشتیم. این باعث شد که dd در STDOUT، به صورت کامل فرآیند و سرعت ایجاد این المثنی را به ما نشان دهد.

نکته: تا به اینجا ما دستگاه‌هایی که روی سیستم مانت شده بودند را به عنوان ورودی به dd دادیم و خواستیم که dd از آنها المثنی ایجاد کند. این امر، کار خطرناکی است و عقلانی نمی‌باشد. شدیدا پیشنهاد می‌شود که قبل از ایجاد المثنی از پارتیشن‌ها (یا حداقل پارتیشن‌های سیستمی)، آنها را unmount کرده و سپس به سراغ ایجاد المثنی بروید.

ما می‌توانیم از دستور dd برای پاک‌کردن کامل یک هارددیسک نیز استفاده کنیم. ممکن است بدانید که هنگام پاک‌کردن فایل‌ها از روی یک هارددیسک، داده‌ی اصلی مربوط به هر فایل از روی هارددیسک پاک نمی‌شود، بلکه لینکی که از سیستم‌عامل به فایل وجود داشته پاک می‌شود و تا زمانی که داده‌های جدید جایگزین داده‌های قبلی موجود بر روی سکتورها نشوند، داده‌های قبلی قابل بازیابی یا Recovery خواهند بود. یکی از دلایلی که پاک‌کردن یک فایل چند ثانیه طول می‌کشد، اما کپی کردن یک فایل جدید چندین دقیقه، همین امر می‌باشد.

در بسیاری از اوقات، ما نمیخواهیم اطلاعاتی که از روی دیسک پاک کرده‌ایم، قابل بازیابی باشند. ما در لینوکس، یک‌سری دیوایس‌فایل مخصوص داریم. یکی از این دیوایس‌فایل‌ها، /dev/zero می‌باشد. این دیوایس‌فایل، به هر تعدادی که بخواهیم به ما صفر در خروجی می‌دهد. ما می‌توانیم با استفاده از این دیوایس‌فایل و دستور dd، برای از بین بردن کامل اطلاعات موجود برروی هارددیسک استفاده کنیم. به عبارت دیگر، ما می‌توانیم با مشخص

کردن دیوایس‌فایل /dev/zero به عنوان ورودی dd و مشخص کردن پارتیشن یا هارددیسکی که میخواهیم کاملا پاک شود به عنوان خروجی dd، کلیه اطلاعات موجود بر روی هارددیسک مورد نظر را طوری پاک کنیم که به هیچ عنوان قابل بازیابی نباشند. پس:

```
[root@localhost ~]# dd if=/dev/zero of=/dev/sdb1 status=progress
5368402432 bytes (5.4 GB) copied, 408.124874 s, 13.2 MB/s
dd: writing to ‘/dev/sdb1’: No space left on device
10485761+0 records in
10485760+0 records out
5368709120 bytes (5.4 GB) copied, 408.188 s, 13.2 MB/s
```

همانطور که می‌بینید، ما در اینجا اقدام به پاک‌کردن کامل اطلاعات پارتیشن /dev/sdb1 کردیم. کاری که این دستور انجام می‌دهد این است که روی کلیه‌ی بلوک یا سکتورهای هارددیسک مشخص شده، عدد صفر را می‌نویسد (به دلیل استفاده از /dev/zero به عنوان if). جالب است بدانید که اجرای یک باره‌ی این دستور کافی نیست و معمولا باید تا ۱۰ بار روی کلیه‌ی سکتورهای یک پارتیشن صفر را بنویسیم تا هارددیسک به صورت کامل پاک شود. طبیعی است که ما معمولا برای پاک‌کردن کامل هارددیسک به سراغ سایر ابزارها می‌رویم و معمولا از dd برای این کار استفاده نمی‌کنیم.

اگر به خاطر داشته باشید، جلسه‌ی قبل با استفاده از dd و /dev/zero، تعدادی فایل ساختگی با حجم دلخواه ایجاد کردیم. بیایید این دستورها را بار دیگر به خاطر آوریم و آن را توضیح دهیم:

```
[root@localhost ~]# dd if=/dev/zero of=testfile bs=2G count=1
```

ما تا به اینجا می‌دانیم که بخش if و of این دستور چه معنایی دارند، اما بیایید در مورد bs و count صحبت کنیم. bs، مخفف block size می‌باشد و ماکزیمم سایز هر بلوک هنگام خواندن یا نوشتن را مشخص می‌کند. count، تعداد بلوک‌هایی که از ورودی باید کپی شوند را مشخص می‌کند. یعنی ما در اینجا می‌گوییم که یک بلوک (count=1) که دارای سایز ۲ گیبی‌بایت می‌باشد را ایجاد کند و آن را در قالب فایل testfile، ذخیره کند.

نکته: ما دیوایس‌فایل‌های مخصوص دیگری مانند /dev/zero نیز داریم که معروف‌ترین آنها /dev/random و /dev/urandom می‌باشند. این دیوایس‌فایل‌ها، به تعداد مورد نظر در خروجی عدد تصادفی تولید می‌کنند. ما می‌توانیم به جای استفاده از /dev/zero در هر کدام از سناریوهای بالا، از /dev/random و /dev/urandom استفاده کنیم.

مدیریت لینک‌ها

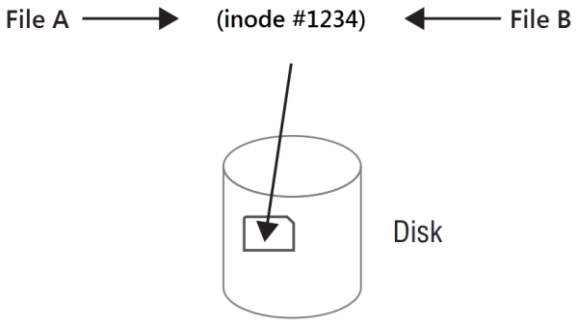
لینک‌ها، یکی از روش‌هایی می‌باشند که از طریق آن می‌توانیم به یک فایل یا دایرکتوری، چندین هویت بدهیم. لینک‌ها در لینوکس، عملکردی نظیر Shortcutها در ویندوز را دارند. در لینوکس، ما دو نوع لینک داریم و در این بخش، با این دو نوع لینک آشنا می‌شویم. قبل از صحبت در مورد لینک‌ها، باید به صورت خیلی مختصر و کلی در مورد مفهوم inodeها در لینوکس صحبت کنیم.

به طور خیلی ساده، در سیستم‌های لینوکسی، inode ساختمان داده‌ای می‌باشد که یک سری اطلاعات در مورد فایل‌ها و دایرکتوری‌های موجود در هارددیسک را درون خود نگه داری می‌کند. این اطلاعات، شامل permissionها (مالک فایل، مجوز خواندن/نوشتن و...)، نوع فایل و... می‌باشند. اما نکته‌ای که الان می‌خواهیم به آن توجه کنیم این است که inodeها، موقعیت هر فایل (شماره‌ی سکتور یا بلوک) بر روی هارددیسک را نیز

درون خود دارند. کرنل لینوکس با توجه به inodeها، می‌تواند فایل‌ها و دایرکتوری‌ها را پیدا کند. هر فایل یا دایرکتوری که روی سیستم داشته باشیم، یک شماره‌ی inode منحصر به فرد دارد که سیستم با نگاه کردن به آن شماره، می‌تواند اطلاعات مربوط به آن فایل یا دایرکتوری را پیدا کند.

هاردلینک‌ها

هاردلینک‌ها، فایل‌ها یا دایرکتوری‌هایی هستند که دارای یک شماره‌ی inode یکسان می‌باشند، اما نام متفاوتی دارند. داشتن شماره‌ی inode یکسان بدین معنی است که همه‌ی آن فایل‌ها و دایرکتوری‌ها، به یک نقطه (سکتور یا بلوک) بر روی هارددیسک اشاره می‌کنند. داشتن نام‌های متفاوت باعث می‌شود که بتوانیم به روش‌های متفاتی به فایل دسترسی پیدا کنیم. تصویر ۱ چگونگی عملکرد هاردلینک‌ها را به خوبی نشان می‌دهد. پس هاردلینک‌ها در ساده‌ترین حالت، دو فایل هستند که نام‌های متفاوتی دارند، اما یک شماره‌ی inode یکسان دارند و وجود این شماره‌ی inode یکسان، باعث می‌شود که هر دو فایل، یک موقعیت برروی هارددیسک داشته باشند، یا به عبارت دیگر، به یک فایل اشاره کنند. پس ما دو فایل با دو نام متفاوت داریم که از نظر فیزیکی، یک فایل هستند.



تصویر ۱ - ساختار هاردلینک‌ها

اگر دو فایل به نام A و B داشته باشیم که به هم هاردلینک شده باشند، هر تغییری در فایل A ایجاد کنیم، در فایل B نیز ایجاد می‌شود. به همین شکل، هر تغییری در فایل B ایجاد کنیم، در فایل A نیز ایجاد می‌شود. در چنین حالتی اگر کسی فایل A را پاک کند، هنوز فایل B در سر جای خود باقی می‌ماند و عکس این قضیه نیز صادق است. این امر بسیار کاربردی می‌باشد، چون ما می‌توانیم به آن، به عنوان یک بک‌آپ نگاه کنیم؛ بک‌آپی که همزمان با فایل اصلی، آپدیت می‌شود.

برای ایجاد هاردلینک، از دستور `ln` استفاده می‌کنیم. فرض کنید می‌خواهیم یک هاردلینک به فایل `/etc/postfix/main.cf` بزنیم:

```
[root@localhost ~]# ln /etc/postfix/main.cf postfix-main.cf
```

همانطور که می‌بینید، ما ابتدا دستور `ln` را وارد کردیم، سپس فایلی که می‌خوستیم به آن هاردلینک بزنیم را وارد کرده (`/etc/postfix/main.cf`) و پس از آن، نام فایل جدیدی که می‌خواستیم به فایل اول هاردلینک شود را مشخص کردیم (`postfix-main.cf`). توجه کنید که این فایل نباید از قبل وجود داشته باشد. بیایید از صحت ایجاد این هاردلینک مطمئن شویم:

```
[root@localhost ~]# ls
postfix-main.cf
```

اما از کجا بدانیم که این فایل، به /etc/postfix/main.cf هاردلینک شده است؟ بیایید به inodeهای این دو فایل نگاهی بیاندازیم. برای پیدا کردن inode یک فایل، می‌توانیم از دستور ls -i استفاده کنیم. پس:

```
[root@localhost ~]# ls -i postfix-main.cf
50803061 postfix-main.cf
[root@localhost ~]# ls -i /etc/postfix/main.cf
50803061 /etc/postfix/main.cf
```

در ستون اول خروجی ls، می‌توانیم شماره‌ی inode این فایل‌ها را ببینیم و همانطور که مشاهده می‌کنید، این دو فایل دارای شماره‌ی inode یکسان می‌باشند. ما می‌توانیم با استفاده از دستور ls -l نیز، تعداد هاردلینک‌های زده شده به یک inode را مشاهده کنیم. ابتدا برای مقایسه، یک فایل خالی ایجاد می‌کنیم و سپس دستور ls -l را اجرا می‌کنیم:

```
[root@localhost ~]# touch test-file
[root@localhost ~]# ls -l
total 28
-rw-r--r--. 2 root root 27176 Oct 30  2018 postfix-main.cf
-rw-r--r--. 1 root root     0 Jul 15 13:06 test-file
```

همانطور که می‌بینید، در ستون مربوط به لینک‌های فایل postfix-main.cf، عدد ۲ مشاهده می‌شود. این یعنی به inode مربوط به این فایل، ۲ عدد لینک زده شده است. در حالی که در ستون مربوط به لینک‌های فایل test-file، عدد ۱ مشاهده می‌شود. این یعنی به inode مربوط به این فایل، فقط ۱ لینک زده شده است.

با توجه به حرفهایی که زدیم، می‌دانیم که اگر این فایل را باز کنیم، دقیقا مانند فایل /etc/postfix/main.cf می‌باشد. همچنین اگر فایل postfix-main.cf را با ادیتور vi باز کنیم و یک خط نوشته به ابتدای آن اضافه کنیم، این تغییرات در فایل /etc/postfix/main.cf نیز قابل مشاهده خواهد بود. بیایید این کار را امتحان کنیم:

```
[root@localhost ~]# vi postfix-main.cf
#oh this is just a test line, baby!
# Global Postfix configuration file. This file lists only a subset
# of all parameters. For the syntax, and for a complete parameter
# list, see the postconf(5) manual page (command: "man 5 postconf").
#
...
```

همانطور که می‌بینید ما این فایل را با vi باز کرده و خط مشخص شده را به آن اضافه کردیم. توجه کنید که قبل از نوشتن خط مورد نظر، از علامت # استفاده کنید، وگرنه تنظیمات postfix به هم می‌ریزد.

حال بیایید ببینیم که این تغییرات در فایل /etc/postfix/main.cf نیز اعمال شده یا نه:

```
[root@localhost ~]# head -5 /etc/postfix/main.cf
#oh this is just a test line, baby!
# Global Postfix configuration file. This file lists only a subset
# of all parameters. For the syntax, and for a complete parameter
# list, see the postconf(5) manual page (command: "man 5 postconf").
#
```

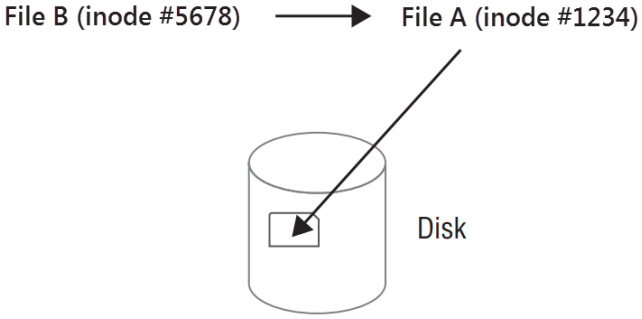
همانطور که می‌بینید، ما با استفاده از دستور head، پنج خط اول موجود در فایل /etc/postfix.main.cf را مشاهده کردیم و دیدیم که طبق انتظار، متنی که در هاردلینک postfix-main.cf اضافه کرده بودیم، در این فایل نیز وجود دارد.

به طور کلی، هنگام ایجاد هاردلینک، باید موارد زیر را در نظر داشته باشیم:

- قبل از استفاده از دستور ln، فایلی که می‌خواهیم به آن هاردلینک بزنیم (فایل اولی که به دستور ln می‌دهیم) باید وجود داشته باشد.
- فایلی که قرار است به عنوان هاردلینک به فایل اصلی عمل کند (فایل دومی که به دستور ln می‌دهیم)، نباید وجود داشته باشد.
- فایل اصلی و کلیه‌ی هاردلینک‌های زده شده به فایل اصلی، دارای شماره‌ی inode یکسان می‌باشند.
- فایل اصلی و کلیه‌ی هاردلینک‌های زده شده به فایل اصلی، دارای محتوای یکسان می‌باشند.
- فایل اصلی و کلیه‌ی هاردلینک‌های زده شده به فایل اصلی، می‌توانند در دایرکتوری‌های متفاوت قرار گیرند.
- فایل اصلی و کلیه‌ی هاردلینک‌های زده شده به فایل اصلی، باید حتما درون یک پارتیشن یکسان قرار داشته باشند.

سافت‌لینک‌ها

به طور کلی، Soft Link یا Symbolic Linkها یک نوع فایل خاص می‌باشند. در واقع اگر فایل A یک فایل موجود بر روی یک پارتیشن باشد که در یک بلوک خاص ذخیره شده است و فایل B به فایل A یک سافت‌لینک زده باشد، فایل B به هیچ موقعیت یا بلوکی روی هارددیسک اشاره نمی‌کند، بلکه مستقیما به فایل A اشاره می‌کند، یا به عبارت دیگر، فایل B، به جای این که به موقعیت فایل A بر روی پارتیشن فایل A اشاره کند، به نام فایل A اشاره می‌کند. این یعنی فایل B، شماره‌ی inode متفاوتی از A خواهد داشت، تقریبا هیچ فضایی اشغال نخواهد کرد و همچنین می‌تواند بر روی پارتیشن یا هارددیسک متفاوتی از فایل A قرار داشته باشد. حتی اگر B به یک فایل ۱۰۰ ترابایتی سافت‌لینک بزند، خود B فضای خیلی کمی از هارددیسک را اشغال می‌کند. تصویر ۲ عملکرد سافت‌لینک‌ها را به زیبایی توصیف می‌کند:



تصویر ۲- ساختار سافت‌لینک‌ها

اگر فایل A را حذف کنیم، فایل B، اصطلاحا تبدیل به یک لینک مرده یا Dead Link خواهد شد و دیگر هیچ کاربردی نخواهد داشت، اما تبدیل به یک ریسک امنیتی خواهد شد.

برای ایجاد سافت‌لینک‌ها از دستور ln به همراه آپشن s- استفاده می‌کنیم. فرض کنید می‌خواهیم به همان فایل /etc/postfix/main.cf سافت لینک بزنیم. پس:

```
[root@localhost ~]# ln -s /etc/postfix/main.cf soft-main.cf
```

حال بیایید از صحت ایجاد این فایل مطمئن شویم:

```
[root@localhost ~]# ls -l
total 0
lrwxrwxrwx. 1 root root 20 Jul 16 14:02 soft-main.cf -> /etc/postfix/main.cf
```

همانطور که می‌بینید، اینبار خروجی ls متفاوت است. این جا در مقابل فایل soft-main.cf که سافت‌لینک ما می‌باشد، یک فلش به فایل /etc/postfix/main.cf می‌بینیم. این در واقع همان چیزی است که تصویر ۲ به ما نشان می‌دهد. بیایید شماره‌ی inode این فایل اصلی و فایل سافت‌لینک را بررسی کنیم:

```
[root@localhost ~]# ls -i soft-main.cf
33575022 soft-main.cf
[root@localhost ~]# ls -i /etc/postfix/main.cf
50803061 /etc/postfix/main.cf
```

همانطور که می‌بینید شماره‌ی inode این دو فایل کاملا با هم متفاوت است. بیایید به حجم این دو فایل نیز نگاهی بیاندازیم:

```
[root@localhost ~]# du -h soft-main.cf
0       soft-main.cf
[root@localhost ~]# du -h /etc/postfix/main.cf
28K     /etc/postfix/main.cf
```

همانطور که می‌بینید، soft-main.cf، بر خلاف فایل اصلی (/etc/postfix.main.cf)، هیچ حجمی ندارد. اگر ما به فایل soft-main.cf محتوایی اضافه کنیم، آن محتوا در فایل /etc/postfix/main.cf ذخیره می‌شود. ما تست این امر را به خودتان می‌سپاریم.

در اینجا باید به یک نکته‌ی مهم دیگر نیز اشاره کنیم. به خروجی ls -l توجه کنید:

```
[root@localhost ~]# ls -l
total 0
lrwxrwxrwx. 1 root root 20 Jul 16 14:02 soft-main.cf -> /etc/postfix/main.cf
[root@localhost ~]# ls -l /etc/postfix/main.cf
-rw-r--r--. 1 root root 27231 Jul 16 14:10 /etc/postfix/main.cf
```

اگر توجه کنید، در ستون تعداد لینک، همچنان عدد ۱ مشاهده می‌شود. دلیل این امر این است که این ستون فقط تعداد لینک‌های زده شده به یک شماره‌ی inode خاص را نشان می‌دهد. از آنجایی که شماره‌ی inode فایل سافت‌لینک با شماره‌ی inode فایل اصلی متفاوت می‌باشد، عدد موجود در این ستون، ۱ می‌باشد. به عبارت دیگر، ما با نگاه کردن به این ستون، نمی‌توانیم تعداد لینک‌های زده شده به یک فایل را پیدا کنیم.

به طور کلی برای ایجاد سافت‌لینک، باید موارد زیر را در نظر داشته باشیم:

- قبل از استفاده از دستور ln -s، فایلی که می‌خواهیم به آن سافت‌لینک بزنیم (فایل اولی که به دستور ln -s می‌دهیم یا فایل اصلی) باید وجود داشته باشد.
- فایلی که قرار است به عنوان سافت‌لینک به فایل اصلی عمل کند (فایل دومی که به دستور ln -s می‌دهیم)، نباید وجود داشته باشد.
- فایل اصلی و کلیه‌ی سافت‌لینک‌های زده شده به فایل اصلی، دارای شماره‌ی inode متفاوت می‌باشند.
- فایل اصلی و سافت‌لینک‌های زده شده به فایل اصلی، دارای داده‌ی یکسان نیستند؛ چرا که سافت‌لینک داده‌ای بر روی دیسک ذخیره نمی‌کند، بلکه صرفا ما را به فایل اصلی ارجاع می‌دهد. این یعنی اگر

فایل اصلی پاک شود، سافت‌لینک هیچ چیزی درون خود نخواهد داشت.

- فایل اصلی و کلیه‌ی سافت‌لینک‌های زده شده به فایل اصلی، می‌توانند در دایرکتوری‌های متفاوت قرار گیرند.
- فایل اصلی و کلیه‌ی سافت‌لینک‌های زده شده به فایل اصلی، می‌توانند در پارتیشن‌های متفاوت قرار داشته باشند.

مدیریت مالکیت فایل‌ها

در سیستم‌های لینوکسی، مالکیت فایل‌ها دارای دو جنبه می‌باشد؛ بدین شکل که هر فایل، یک صاحب یا Owner دارد و از آنجایی که هر صاحب می‌تواند عضو یک گروه یا Group باشد، هر فایل یک گروه نیز دارد. این دو جنبه، باعث می‌شوند که ما سه لایه دسترسی برای هر فایل داشته باشیم:

- **لایه‌ی دسترسی Owner یا مالک فایل:** مجوزهای مالک فایل (خواندن، نوشتن، اجرا) در برخورد با فایل را مشخص می‌کنند.
- **لایه‌ی دسترسی Group یا گروه:** مجوزهای اعضای گروه مالک فایل (خواندن، نوشتن، اجرا) در هنگام برخورد با فایل را مشخص می‌کند. این مجوزها می‌توانند دقیقا مانند مجوزهای مالک فایل باشند، یا می‌توانند با آن تفاوت داشته باشند.
- **لایه‌ی دسترسی Others یا سایرین:** مجوزهای (خواندن، نوشتن، اجرا) کاربرانی که نه صاحب فایل هستند و نه عضوی از گروه فایل هستند را مشخص می‌کند.

ابتدا بیایید در مورد چگونگی پیدا کردن مالک و همچنین گروه فایل صحبت کنیم.

بررسی مالکین فایل

با استفاده از آپشن `l-` دستور `ls`، می‌توانیم مالک و گروه یک فایل را مشاهده کنیم:

```
[root@localhost ~]# ls -l /etc/postfix/
total 148
-rw-r--r--. 1 root root 20876 Oct 30  2018 access
-rw-r--r--. 1 root root 11883 Oct 30  2018 canonical
-rw-r--r--. 1 root root 10106 Oct 30  2018 generic
-rw-r--r--. 1 root root 21545 Oct 30  2018 header_checks
-rw-r--r--. 1 root root 27231 Jul 16 14:10 main.cf
-rw-r--r--. 1 root root  6105 Oct 30  2018 master.cf
-rw-r--r--. 1 root root  6816 Oct 30  2018 relocated
-rw-r--r--. 1 root root 12549 Oct 30  2018 transport
-rw-r--r--. 1 root root 12696 Oct 30  2018 virtual
```

ستون اول این خروجی، مجوزهای دسترسی به این فایل را مشخص می‌کند. ما در حال حاضر کاری با این ستون نداریم و جلوتر آن را توضیح می‌دهیم. در حال حاضر، ستون سوم و چهارم برای ما مهم هستند. ستون سوم، نشان دهنده‌ی Username مالک فایل می‌باشد و ستون چهارم، نشان دهنده‌ی گروه فایل می‌باشد.

بسیاری از توزیع‌های لینوکسی، اعم از CentOS، هر کاربر را به یک گروه که دقیقا هم‌نام با Username کاربر می‌باشد اختصاص می‌دهد. این امر، باعث می‌شود که فایل‌ها به صورت اشتباهی بین همه‌ی اعضای یک گروه به اشتراک گذاشته نشوند. پس دلیل این که در هر دو ستون عبارت `root` مشاهده می‌شود، این است که مالک فایل که کاربر `root` می‌باشد، عضوی از گروه `root` می‌باشد.

نکته: هر کاربری که یک فایل یا دایرکتوری را ایجاد کند، به صورت پیش‌فرض، مالک آن فایل یا دایرکتوری می‌شود. همچنین گروهی که کاربر عضو آن بوده، به عنوان گروه مرتبط با آن فایل، انتخاب می‌شوند.

تغییر مالک فایل

برای تغییر مالک یک فایل، از دستور chown استفاده می‌کنیم. به طور کلی، استفاده از این دستور به صورت زیر می‌باشد:

```
chown [OPTIONS] NEWOWNER FILENAMES
```

همانطور که می‌بینید، برای استفاده از این دستور، کافی است این دستور را به همراه آپشن‌های مورد نظر وارد کرده، سپس Username مالک جدید را وارد کرده و سپس نام فایل یا فایل‌هایی که می‌خواهیم مالکشان را عوض کنیم را وارد کنیم.

بیایید این دستور را روی فایل /etc/postfix/main.cf امتحان کنیم:

```
[root@localhost ~]# ls -l /etc/postfix/main.cf
-rw-r--r--. 1 root root 27231 Jul 16 14:10 /etc/postfix/main.cf
[root@localhost ~]# chown postfix /etc/postfix/main.cf
```

همانطور که می‌بینید، مالک فایل /etc/postfix/main.cf، در ابتدا کاربر root بود. سپس با استفاده از chown، ارائه‌ی نام کاربر postfix و نام فایل، مالک این فایل را از root به postfix تغییر دادیم. بیایید از صحت این امر مطمئن شویم:

```
[root@localhost ~]# ls -l /etc/postfix/main.cf
-rw-r--r--. 1 postfix root 27231 Jul 16 14:10 /etc/postfix/main.cf
```

دستور chown چندین آپشن دارد، اما اکثر آنها کاربردی نمی‌باشند. از این میان، شاید بتوان گفت آپشن R- به‌دردبخور می‌باشد، چرا که این آپشن می‌تواند مالک کلیه‌ی فایل‌های موجود در یک دایرکتوری را به صورت recursive تغییر دهد. ما بررسی عملکرد این آپشن را به خودتان می‌سپاریم.

تغییر گروه یک فایل

برای تغییر گروهی که یک فایل به آن تعلق دارد، از دستور chgrp استفاده می‌کنیم. استفاده از این دستور، بسیاری شبیه به استفاده از دستور chown می‌باشد:

```
chgrp [OPTIONS] NEWGROUP FILENAMES
```

همانطور که می‌بینید، ما برای تغییر گروه یک فایل یا دایرکتوری، دستور chgrp را با آپشن‌های مورد نظر وارد کرده، سپس نام گروه جدیدی که میخواهیم فایل به آن تعلق پیدا کند را وارد می‌کنیم و سپس نام فایل یا فایل‌هایی که میخواهیم به گروه جدید ملحق شوند را وارد می‌کنیم.

بیایید این دستور را روی یک فایل ساختگی امتحان کنیم:

```
[root@localhost ~]# touch test-file.txt
[root@localhost ~]# ls -l test-file.txt
-rw-r--r--. 1 root root 0 Jul 18 14:04 test-file.txt
[root@localhost ~]# chgrp postfix test-file.txt
```

همانطور که می‌بینید، ما ابتدا با استفاده از touch، یک فایل ساختگی به نام test-file.txt ساختیم و سپس مالکین آن فایل را با استفاده از دستور ls -l بررسی کردیم. می‌بینید که فایل ابتدا متعلق به کاربر root و گروه root بوده است. سپس با استفاده از دستور chgrp، گروهی که فایل به آن متعلق بود را به گروه

postfix تغییر دادیم. بیایید از صحت این امر مطمئن شویم:

```
[root@localhost ~]# ls -l
total 0
-rw-r--r--. 1 root postfix 0 Jul 18 14:04 test-file.txt
```

این دستور نیز آپشن‌های زیادی دارد که زیاد کاربردی نیستند؛ با این حال، آپشن R- این دستور می‌تواند کارآمد باشد، چرا که به صورت recursive، گروه کلیه‌ی فایل‌های موجود در یک دایرکتوری را تغییر می‌دهد.

نکته: ما می‌توانیم با دستور chown که در بخش قبل یاد گرفتیم، هم مالک فایل و هم گروه فایل را تغییر دهیم. برای این کار، کافی است به صورت زیر عمل کنیم:

```
[root@localhost ~]# touch another-file.txt
[root@localhost ~]# ls -l
total 0
-rw-r--r--. 1 root root 0 Jul 18 14:22 another-file.txt
[root@localhost ~]# chown postfix:postfix another-file.txt
[root@localhost ~]# ls -l
total 0
-rw-r--r--. 1 postfix postfix 0 Jul 18 14:22 another-file.txt
```

همانطور که می‌بینید، برای عوض کردن همزمان مالک و گروه فایل، کافی است هنگام اعمال نام مالک جدید به دستور chown، یک علامت : پس از نام مالک جدید قرار دهیم و در مقابل آن نام گروه جدید را بنویسیم. یا به عبارت دیگر:

```
chown [options] NEW_OWNER:NEW_GROUP FILENAMES
```

مدیریت مجوز دسترسی به فایل‌ها

همانطور که قبلا دیدیم، هنگام استفاده از دستور ls -l، لینوکس اطلاعات متفاوتی در مورد یک فایل، اعم از مجوزهای دسترسی آن، به ما می‌دهد:

```
[root@localhost ~]# ls -l
total 0
-rwxrw-r--. 1 root root 0 Jul 19 22:14 test-file.txt
```

ما تا به اینجا چندین بار در مورد خروجی این دستور و مفهوم هر ستون در آن صحبت کرده‌ایم، اما بار دیگر نیز به صورت مختصر تک‌تک ستون‌های خروجی این دستور را توضیح می‌دهیم. در تصویر ۳، مفهوم هر کدام از خروجی‌های دستور ls -l را مشاهده می‌کنید. ما قبلا در مورد همه‌ی ستون‌ها، به جز ستون اول و دوم صحبت کرده‌ایم و در این بخش می‌خواهیم مفهوم دو ستون اول این خروجی را توضیح دهیم.

test-file.txt	Jul 19 22:14	0	root	root	1	rwxrw-r--.	-
نام فایل	آخرین زمان اعمال تغییرات روی فایل	حجم فایل (بایت)	گروه فایل	مالک فایل	تعداد هاردلینک‌ها	مجوزهای دسترسی به فایل	کد مشخص کننده‌ی نوع فایل

تصویر ۳- مفهوم هر کدام از ستون‌های خروجی دستور ls -l

کدهای مشخص‌کننده‌ی نوع فایل

ابتدا بیایید با کدهای مشخص کننده‌ی نوع فایل آشنا شویم. این کدها به شرح زیر می‌باشند:

جدول ۱- کدهای مشخص کننده‌ی نوع فایل و مفهوم آنها

کد	مفهوم
-	فایل، یک فایل باینری، فایل متنی، فایل image یا یک فایل فشرده می‌باشد.
d	فایل، در واقع یک دایرکتوری می‌باشد.
l	فایل، یکسافت‌لینک به یک فایل دیگر می‌باشد.
p	فایل، یک پایپ می‌باشد که از آن برای ایجاد ارتباط بین دو پراسس استفاده می‌شود.
s	فایل، یک فایل سوکت می‌باشد که عملکردی نظیر فایل پایپ دارد، با این تفاوت که می‌تواند ارتباطات بین پراسس‌ها را از روی شبکه و... نیز برقرار کند.
b	فایل، یک دیوایس‌فایل بلوکی می‌باشد، مثل دیوایس‌فایل هارددیسک‌ها و...
c	فایل، یک دیوایس‌فایل کاراکتری می‌باشد، مثل کیبورد و...

برای مثال، بیایید نوع فایل‌های زیر را با استفاده از کد آنها بررسی کنیم:

```
brw-rw----. 1 root disk 8 Jul 21 12:53 sda1
drwxr-xr-x. 2 root root 6 Jul 21 12:55 Directory
-rw-r--r--. 1 root root 0 Jul 21 12:55 file.txt
```

همانطور که می‌بینید، فایل اول یک دیوایس‌فایل بلوکی می‌باشد، فایل دوم یک دایرکتوری می‌باشد و فایل سوم، یک فایل متنی می‌باشد.

مجوزهای دسترسی به فایل

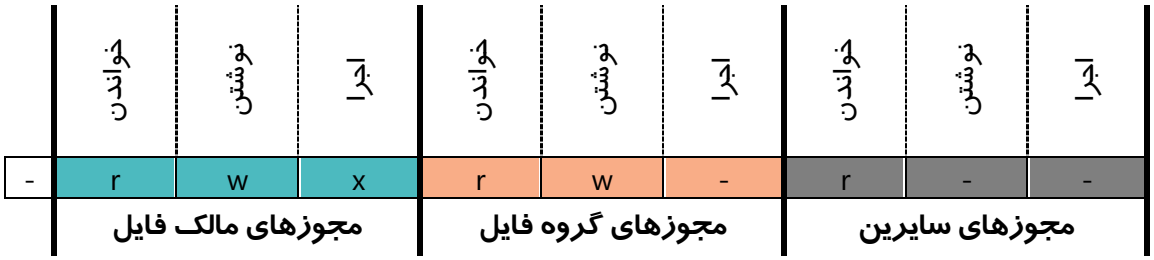
لینوکس از سه نوع مجوز دسترسی استفاده می‌کند. این مجوزها، مجوز خواندن، مجوز نوشتن و مجوز اجرا می‌باشند. توجه کنید که مفهوم مجوزها در فایل‌ها با مجوزها در دایرکتوری‌ها، کمی متفاوت می‌باشد:

جدول ۲- تفاوت مفهوم مجوزها در فایل‌ها و دایرکتوری‌ها

مجوز	فایل	دایرکتوری
خواندن (read)	قابلیت مشاهده‌ی محتویات فایل را به کاربر دارای این مجوز می‌دهد.	قابلیت لیست کردن فایل‌های موجود درون دایرکتوری را به کاربر دارای مجوز می‌دهد.
نوشتن (write)	قابلیت تغییر محتویات فایل را به کاربر دارای این مجوز می‌دهد.	قابلیت ایجاد، جابه‌جایی (یا تغییر نام) و پاک کردن فایل‌های موجود درون دایرکتوری را به کاربر دارای این مجوز می‌دهد.
اجرا (execute)	قابلیت اجرای فایل به عنوان یک اسکریپت یا یک فایل باینری را به کاربر دارای این مجوز می‌دهد.	به کاربر داری این مجوز، امکان می‌دهد که بتواند موقعیت کنونی خود را به این دایرکتوری تغییر دهد (دایرکتوری را باز کند).

ما می‌توانیم به هر کدام از لایه‌های دسترسی (مالک فایل، گروه فایل، سایرین)، مجوزهای خواندن (مشخص شده با کد r)، نوشتن (مشخص شده با کد w) و اجرا (مشخص شده با کد x) متفاوتی اختصاص دهیم. این یعنی که ما می‌توانیم ۹ مجوز متفاوت به هر فایل یا دایرکتوری موجود در لینوکس اختصاص دهیم؛ یعنی ۳ مجوز برای لایه‌ی مالک فایل (خواندن، نوشتن و اجرا) ، ۳ مجوز برای گروه فایل (خواندن، نوشتن و اجرا) و ۳ مجوز برای سایرین (خواندن، نوشتن و اجرا).

در تصویر ٤، مفهوم کد، لایه و ترتیب اختصاص هر مجوز که در سیستم‌های لینوکسی به فایل‌ها و دایرکتوری‌ها اختصاص داده می‌شود را می‌بینیم. ما گفتیم که در خروجی `ls -l`، پس از کد مشخص کننده‌ی نوع فایل، مجوز دسترسی فایل قرار گرفته است. پس:



تصویر ٤- مفهوم رشته‌ی مشخص کننده‌ی مجوز فایل

همانطور که در تصویر ٤ می‌بینید، سه کاراکتر اول موجود در رشته‌ی مجوز فایل، نشان دهنده‌ی مجوزهای دسترسی مالک فایل می‌باشند. این مجوزها، به ترتیب مجوز خواندن (r)، مجوز نوشتن (w) و مجوز اجرا (x) می‌باشند.

سه کاراکتر بعدی، مشخص کننده‌ی مجوزهای دسترسی گروه فایل می‌باشند. این سه کاراکتر نیز به ترتیب مجوز خواندن (r)، مجوز نوشتن (w) و مجوز اجرا (x) را مشخص می‌کنند و خط تیره نیز به معنای عدم وجود مجوز می‌باشد. کاربری که مالک فایل نباشد، اما عضوی از گروه این فایل باشد، هنگام برخورد با فایل، این مجوزهای دسترسی را خواهد داشت.

سه کاراکتر نهایی، مشخص کننده‌ی مجوزهای دسترسی سایر کاربران می‌باشد. این یعنی کاربرانی که نه مالک فایل هستند و نه عضوی از گروه فایل، این مجوزها را هنگام برخورد با فایل خواهند داشت. به این لایه مجوز، بعضا Other Permissions یا World Permissions نیز می‌گویند. در اینجا هم مجوز خواندن (r)، نوشتن (w) و اجرا (x) با همان ترتیب قبلی موجود می‌باشد.

تغییر مجوزهای یک فایل با استفاده از *chmod*

ما می‌توانیم با استفاده از دستور chmod، مجوزهای یک فایل را تغییر دهیم. دستور chmod در ابتدا می‌تواند کمی گیج کننده باشد، چرا که می‌تواند مجوزها را در دو فرمت متفاوت تغییر دهد. این فرمت‌ها، فرمت‌های Symbolic و Octal می‌باشند.

استفاده از *chmod* در حالت Symbolic

در حالت symbolic، مجوزهای هر فایل یا دایرکتوری را ابتدا با مشخص کردن کد لایه‌ی مجوز و سپس مشخص کردن کد مجوز مورد نظر تغییر می‌دهیم. کد مربوط به لایه‌ها و کد مربوط به هر مجوز را به ترتیب در جدول ۳ و جدول ٤ مشاهده می‌کنید:

جدول ۳-کد مربوط به هر لایه‌ی مجوز

کد	لایه
u	مالک فایل
g	گروه فایل
o	سایرین
a	اعمال بر روی همه‌ی لایه‌ها

جدول ٤- کد مربوط به هر مجوز

کد	مجوز
r	مجوز خواندن
w	مجوز نوشتن
x	مجوز اجرا
-	عدم وجود مجوز

اعمال مجوزها، با مشخص کردن کد مربوط به لایه، استفاده از علامت‌های +، - یا = و سپس مشخص کردن کد مربوط به مجوز، صورت می‌پذیرد. معنای هر کدام از علامت‌های +، - و = به شرح زیر می‌باشد:

- اگر بخواهیم یک مجوز را به لایه‌ای از فایل اضافه کنیم، از علامت + بین کد لایه مورد نظر و کد مجوز استفاده می‌کنیم.
- اگر بخواهیم یک مجوز را از لایه‌ای از فایل حذف کنیم، از علامت - بین کد لایه مورد نظر و کد مجوز استفاده می‌کنیم.
- اگر بخواهیم مجوزی که به فایل میدهیم، جایگزین مجوزهای قبلی لایه‌ی مورد نظر شوند، از علامت = بین کد لایه‌ی مورد نظر و کد مجوز استفاده می‌کنیم.

به احتمال زیاد تا الان خیلی گیج شده‌اید. پس بهتر است با ذکر چندین مثال این قضیه را بهتر درک کنیم.

بیایید ابتدا با استفاده از touch، یک فایل به نام test-file.txt ایجاد کنیم:

```
[root@localhost ~]# touch test-file.txt
```

حال بیایید نگاهی به مجوزهای این فایل بیاندازیم:

```
[root@localhost ~]# ls -l
total 0
-rw-r--r--. 1 root root 0 Jul 19 13:07 test-file.txt
```

همانطور که می‌بینید، این فایل یک فایل معمولی می‌باشد (-)، صاحب این فایل مجوز خواندن و نوشتن دارد (rw-)، کاربران عضو گروه این فایل، فقط مجوز خواندن دارند (r--) و سایر کاربران نیز فقط مجوز خواندن دارند (r--).

ابتدا بیایید مجوز نوشتن را از صاحب فایل، سلب کنیم. برای این کار:

```
[root@localhost ~]# chmod u-w test-file.txt
```

همانطور که می‌بینید، ما ابتدا کد لایه‌ی مورد نظر را به chmod دادیم؛ از آنجایی که می‌خواستیم مجوزهای لایه‌ی مالک فایل دچار تغییر شود، از کد u استفاده کردیم. سپس چون میخواستیم یک مجوز را از فایل حذف

کنیم، از علامت - استفاده کردیم. سپس کد مجوزی که میخواستیم حذف کنیم را به chmod دادیم؛ از آنجایی که میخواستیم مجوز نوشتن را سلب کنیم، از کد w استفاده کردیم و در نهایت نام فایلی که میخواستیم این تغییرات روی آن صورت گیرد را مشخص کردیم.

حال بیایید به مجوزهای این فایل نگاهی بیاندازیم:

```
[root@localhost ~]# ls -l
total 0
-r--r--r--. 1 root root 0 Jul 19 13:07 test-file.txt
```

همانطور که می‌بینید، chmod دقیقا کاری که می‌خواستیم را انجام داده و مجوز خواندن را از صاحب فایل، سلب کرده است.

بیایید این بار مجوز نوشتن را به اعضای گروه این فایل بدهیم. برای این کار:

```
[root@localhost ~]# chmod g+w test-file.txt
```

اینجا هم مثل قبل، ابتدا مشخص کردیم که می‌خواهیم کدام لایه از مجوزها دچار تغییر شوند (در اینجا، g یا لایه‌ی گروه فایل)، سپس نوع تغییر را مشخص کردیم (در اینجا، + یا اضافه کردن مجوز) و در نهایت کد مجوزی که باید اضافه شود را مشخص کردیم (در اینجا، w یا مجوز نوشتن). حال بیایید به مجوزهای این فایل نگاهی بیاندازیم:

```
[root@localhost ~]# ls -l
total 0
-r--rw-r--. 1 root root 0 Jul 19 13:07 test-file.txt
```

همانطور که می‌بینید، مجوز دلخواه ما به درستی اعمال شده است.

حال بیایید کاری کنیم که سایر کاربران، مجوز نوشتن و اجرای این فایل را داشته باشند، اما مجوز خواندن فایل را نداشته باشند. ما می‌توانیم این کار را به چند روش انجام دهیم. شاید اولین روشی که به نظرتان برسد، سلب مجوز خواندن از سایرین و سپس اعطای مجوز نوشتن و اجرا کردن به آنها باشد. اما برای این کار باید دو دستور وارد کنیم. اینجاست که باید به سراغ استفاده از علامت = برویم. همانطور که قبلا هم گفتیم، علامت = مجوزهایی که به لایه‌ی مورد نظر می‌دهیم را جایگزین مجوزهای قبلی می‌کند، یا به عبارت دیگر، مجوزهای کنونی لایه‌ی مشخص شده را تبدیل به - کرده و مجوزهای جدید که مشخص می‌کنیم را به جای آن قرار می‌دهد. پس:

```
[root@localhost ~]# chmod o=wx test-file.txt
```

همانطور که می‌بینید، با استفاده از علامت =، به دستور chmod گفتیم که مجوزهای wx را جایگزین مجوزهای قبلی لایه‌ی سایرین در فایل test-file.txt کند. حال بیایید به مجوزهای این فایل نگاهی بیاندازیم:

```
[root@localhost ~]# ls -l
total 0
-r--rw--wx. 1 root root 0 Jul 19 13:07 test-file.txt
```

همانطور که می‌بینید، مجوزهای مورد نظر ما جایگزین مجوزهای قبلی این لایه شده‌اند.

از این مثال، می‌توانیم یک نکته‌ی دیگر نیز یاد بگیریم. ما هنگام استفاده از chmod، می‌توانیم چندین کد مجوز را پشت هم قرار دهیم. یعنی مثلا اگر بخواهیم به گروه یک فایل مجوز خواندن و نوشتن را اعطا کنیم، می‌توانیم دستور را به صورت chmod g+rw وارد کنیم.

استفاده از *chmod* در حالت Octal

در حالت Octal، ۹ کاراکتر مجوز (یعنی همان rw-r--r-- و...) را با ۳ رقم هشت‌هشتی بیان می‌کنیم. رقم اول، بیانگر مجوزهای مالک فایل، رقم دوم بیانگر مجوزهای گروه فایل و رقم سوم، بیانگر مجوزهای سایرین می‌باشد. رقم مربوط به هر کاراکتر مجوز را در جدول می‌بینیم:

جدول ۵- رقم مربوط به هر کاراکتر مجوز

مقدار رقم	کد مجوز	معنا
0	---	هیچ مجوزی اعطا نشده است.
1	--x	فقط مجوز اجرا
2	-w-	قفط مجوز خواندن
3	-wx	مجوز نوشتن و اجرا
4	r--	فقط مجوز خواندن
5	r-x	مجوز خواندن و اجرا
6	rw-	مجوز خواندن و نوشتن
7	rwx	مجوز خواندن، نوشتن و اجرا

اگر بخواهیم جدول بالا را از دید دیگری بیان کنیم، می‌گوییم که هر کدام از مجوزها، یک مقدار رقمی دارند. یعنی:

- مجوز خواندن مقدار ٤ را دارد.
- مجوز نوشتن مقدار ۲ را دارد.
- مجوز اجرا مقدار ۱ را دارد.

اگر بخواهیم به یک فایل مجوز خواندن را بدهیم، از رقم ٤ استفاده می‌کنیم. اگر بخواهیم به یک فایل مجوز خواندن و نوشتن بدهیم، از حاصل جمع اعداد ۲ (رقم مربوط به مجوز نوشتن) و ٤ (رقم مربوط به مجوز خواندن) که رقم ۶ می‌باشد، استفاده می‌کنیم.

توجه کنید که هنگام استفاده از chmod در حالت Octal، باید سه رقم به chmod بدهیم:

- رقم اول مشخص کننده‌ی مجوزهای صاحب فایل می‌باشد.
- رقم دوم مشخص کننده‌ی مجوزهای گروه فایل می‌باشد.
- رقم سوم مشخص کننده‌ی مجوزهای سایرین می‌باشد.

به عبارت دیگر، در حالت Octal، نمی‌توانیم فقط مجوز یک لایه را مشخص کنیم و باید مجوز هر سه لایه را همزمان مشخص کنیم.

بیایید یک فایل دیگر با touch ایجاد کرده وچگونگی عملکرد حالت Octal را بررسی کنیم:

```
[root@localhost ~]# touch new-file.txt
[root@localhost ~]# ls -l
total 0
-rw-r--r--. 1 root root 0 Jul 20 12:07 new-file.txt
```

بیایید با استفاده از روش Octal، کاری کنیم که صاحب این فایل، مجوز خواندن، نوشتن و اجرا را داشته باشد، گروه این فایل مجوز خواندن و اجرا را داشته باشد و سایرین هیچ مجوزی نداشته باشند. پس:

- اگر به جدول ۵ نگاه کنید، می‌بینید که رقم مربوط به اعطای مجوز خواندن، نوشتن و اجرا، ۷ می‌باشد (۴+۲+۱)؛ پس اولین رقم ما، که مجوز مالک فایل را مشخص می‌کند، باید ۷ باشد.
- رقم مربوط به اعطای مجوز خواندن و اجرا، ۵ می‌باشد (۴+۱)؛ پس دومین رقم اعمالی به chmod، که مجوز گروه فایل را مشخص می‌کند، باید ۵ باشد.
- رقم مربوط به اعطای هیچ مجوزی، ۰ می‌باشد، پس سومین رقم اعمالی به chmod، باید ۰ باشد.

پس ما این سه رقم را در کنار هم قرار داده و chmod را به صورت زیر اجرا می‌کنیم:

```
[root@localhost ~]# chmod 750 new-file.txt
```

حال بیایید نگاهی به مجوزهای این فایل بیاندازیم:

```
[root@localhost ~]# ls -l
total 0
-rwxr-x---. 1 root root 0 Jul 20 12:07 new-file.txt
```

همانطور که می‌بینید، مجوزها دقیقا همانطور که می‌خواستیم اعمال شدند.

نکته: بسیاری از کاربران هنگام مواجهه با مسائل مربوط به مجوز، سریعا به فایل یا دایرکتوری، مجوز ۷۷۷ را می‌دهند. پرواضح است که این امر می‌تواند از نظر امنیتی سیستم را تحت تاثیر قرار دهد. ما همیشه باید فقط بیت‌های مجوزی که به آنها نیاز داریم را روی یک فایل یا دایرکتوری قرار دهیم.
به طور کلی، اگر در سیستم فایلی را دیدید که مجوزهای ۷۷۷ را دارد، بهتر است به آن فایل مشکوک شوید. نکته‌ی قابل توجه در اینجا این است که سافت‌لینک‌ها، دارای مجوزهای ۷۷۷ می‌باشند. این امر طبیعی است و فقط اشاره به مجوز خود فایل سافت‌لینک می‌باشد و تاثیری روی مجوزهای دسترسی به فایل اصلی ندارد. نکته‌ی جالب این است که اگر سعی کنید مجوز سافت‌لینک را تغییر دهید، مجوزی فایلی که به آن سافت‌لینک زده شده (فایل اصلی) دچار تغییر می‌شود.

تغییر مجوزهای ویژه

ما تا به اینجا با ۹ مجوز، یا ۹ بیت مجوز، آشنا شده‌ایم؛ اما در سیستم‌های لینوکسی، سه بیت مجوز دیگر نیز داریم که به آنها مجوزهای ویژه می‌گویند. لینوکس از این مجوزهای ویژه برای مدیریت و کنترل دقیق‌تر فایل‌ها و دایرکتوری‌ها استفاده می‌کند. در این بخش می‌خواهیم با این سه بیت مجوز ویژه آشنا شویم.

بیت مجوز SUID

مجوز SUID یا Set User ID بر روی فایل‌های قابل اجرا (یعنی فایل‌های Executable، مثل اسکریپت‌ها، یا دستورها و...) اعمال می‌شود. این مجوز به لینوکس می‌گوید که به جای اجرای فایل با مجوزهای مربوط به کاربر اجرا کننده (یعنی مجوزهای گروه یا سایرین)، فایل را با مجوزهای مالک فایل اجرا کند. به عبارت دیگر، این مجوز به لینوکس می‌گوید که فارغ از این که چه کسی فایل دارای SUID را اجرا می‌کند (کاربران هم‌گروه با فایل یا سایر کاربران)، فایل را با مجوزهای مالک فایل اجرا کند.
شاید درک اهمیت این قضیه برایتان دشوار باشد. پس بیایید با یک مثال به درک بهتری از عملکرد SUID برسیم. فرض کنید کاربر A، مالک دو فایل به نام‌های Script و File می‌باشد. فایل Script، یک فایل اسکریپت

با مجوزهای rwxr-xr-x می‌باشد و فایل File، یک فایل متنی با مجوزهای rw-r--r-- می‌باشد. فایل Script به محض اجرا، به سراغ نوشتن یک سری اطلاعات بر روی File می‌رود.

همانطور که گفتیم، همه‌ی کاربران مجوز اجرای فایل اسکریپت را دارند. در اینجا اگر کاربری به جز مالک Script این فایل را اجرا کند چه اتفاقی می‌افتد؟ گفتیم که فایل Script به محض اجرا به سراغ نوشتن بر روی فایل File می‌رود. اما فایل File، مجوز نوشتن را فقط به کاربر A داده است. پس در چنین حالتی اگر کاربری غیر از کاربر A (مالک فایل) به سراغ اجرای Script برود، سیستم به او پیغام خطا می‌دهد، چون اسکریپت با مجوزهای سایرین اجرا شده و File مجوز نوشتن به سایرین را نمی‌دهد.

در اینجا، مجوز SUID می‌تواند ما را نجات دهد. ما اگر مجوز SUID را بر روی فایل Script اعمال کنیم، لینوکس این اسکریپت را با مجوزهای مالک فایل اجرا می‌کند. در چنین حالتی، فایل File، فکر می‌کند که کاربر A (مالک فایل) این اسکریپت را اجرا کرده و در نتیجه به او اجازه‌ی نوشتن را می‌دهد.

به عنوان مثالی دیگر، می‌توان به دستور passwd اشاره کرد. در سیستم‌های لینوکسی، همه‌ی کاربران باید بتوانند رمز ورود خود به سیستم را در هر زمانی که می‌خواهند عوض کنند. در سیستم‌های لینوکسی، رمز کاربران به صورت hash شده در فایلی به نام /etc/shadow ذخیره می‌شود. کاربران برای تغییر رمز خود، از دستوری به نام passwd استفاده می‌کنند. دستور passwd مانند یک اسکریپت می‌باشد، یعنی این دستور رمز مورد نظر هر کاربر را می‌گیرد، آن را hash می‌کند و در پشت صحنه، به سراغ آپدیت‌کردن فایل /etc/shadow می‌رود. در اینجا ما سناریویی نظیر مثال قبل داریم؛ فایل /etc/shadow یک فایل ویژه می‌باشد که فقط کاربر root امکان اعمال تغییرات و حتی مشاهده‌ی آن را دارد. در چنین حالتی باید چه کنیم؟ شاید فکر کنید بهتر باشد که به همه‌ی کاربران مجوز نوشتن بر روی فایل /etc/shadow را بدهیم و خیال خود را راحت کنیم. مشکلی که اینجا وجود دارد این است که اگر ما به همه‌ی کاربران اجازه‌ی نوشتن روی این فایل را بدهیم، آنها می‌توانند رمز سایر کاربران را نیز تغییر دهند و اگر به آنها اجازه‌ی این کار را ندهیم، نمی‌توانند رمز خود را تغییر دهند. پس در اینجا باید چه کنیم؟

در اینجا نیز به سراغ SUID می‌رویم. اگر مجوز SUID را به فایل اجرایی دستور passwd بدهیم، هر کسی این دستور را اجرا کند، سیستم فکر می‌کند root (مالک passwd) آن دستور را اجرا کرده است و در نتیجه، /etc/shadow هم فکر می‌کند که root در حال اعمال تغییرات می‌باشد و به او، اجازه‌ی اعمال تغییرات را می‌دهد.

بیایید نگاهی به مجوزهای اعمال شده بر روی فایل اجرایی این دستور بیاندازیم:

```
[root@localhost ~]# which passwd
/usr/bin/passwd
[root@localhost ~]# ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 27856 Aug  9  2019 /usr/bin/passwd
```

همانطور که می‌بینید، مجوز SUID بر روی این فایل اعمال شده استو. در فایل‌هایی که مجوز SUID بر روی آنها اعمال شده باشد، بیت مربوط به مجوز SUID با حرف s، جایگزین بیت مجوز اجرا (x) در بخش مجوزهای مالک فایل می‌شود.

نکته: اگر SUID روی فایلی قرار گیرد که در آن مالک فایل مجوز اجرا ندارد، این مجوز با حرف S مشخص می‌شود . عملا استفاده از SUID در چنین حالتی مفهومی نخواهد داشت.

برای این که به یک فایل مجوز SUID را بدهیم، می‌توانیم از chmod استفاده کنیم. به طوری که:

- در حالت Symbolic، کاراکتر s را به مجوزهای مالک فایل اضافه می‌کنیم:

```
[root@localhost ~]# chmod u+s myapp
[root@localhost ~]# ls -l
total 0
-rwSr--r--. 1 root root 0 Jul 25 10:49 myapp
```

همانطور که می‌بینید، در قسمت مجوز اجرای لایه‌ی مالک این فایل، کاراکتر S مشاهده می‌شود و دلیل آن این است که مالک این فایل، مجوز اجرا را نداشته است.

- در حالت Octal، یک رقم به ابتدای ۳ رقم مجوز اضافه می‌کنیم. به طوری که اولین رقم، مشخص کننده‌ی آن مجوز خاص می‌باشد و رقم دوم، سوم و چهارم، به ترتیب مجوز مالک، گروه و سایرین را مشخص می‌کنند. از آنجایی که در اینجا می‌خواهیم مجوز خاص SUID را به سیستم بدهیم، از عدد ٤ به عنوان رقم اول استفاده می‌کنیم:

```
[root@localhost ~]# chmod 4751 myapp
[root@localhost ~]# ls -l
total 0
-rwsr-x--x. 1 root root 0 Jul 25 10:49 myapp
```

همانطور که می‌بینید، این بار به جای مجوز اجرا (x) در لایه‌ی مالک فایل، کاراکتر s را مشاهده می‌کنیم.

بیت مجوز SGID

مجوز SGID یا Set Group ID را می‌توانیم از دو جنبه بررسی کنیم:

- اگر SGID روی یک فایل اعمال شود، آن فایل با مجوزهای گروه فایل اجرا می‌شود. وجود یا عدم وجود این مجوز را می‌توان با نگاه کردن به مجوزهای لایه‌ی گروه یک فایل بررسی کرد. این بیت مجوز، جایگزین مجوز اجرا در مجوزهای گروه فایل می‌شود. لازم به ذکر است که در اینجا نیز اگر گروه فایل مجوز اجرا نداشته باشد، کاراکتر S به جای مجوز اجرا قرار می‌گیرد و عملا استفاده از SGID مفهومی نخواهد داشت.
- اگر SGID روی یک دایرکتوری اعمال شود، کلیه‌ی فایل‌های ایجاد شده در آن دایرکتوری، به جای هم‌گروه شدن با کاربر ایجاد کننده‌ی فایل، با خود دایرکتوری هم‌گروه می‌شوند. بدین شکل، همه‌ی کاربران هم‌گروه با دایرکتوری، می‌توانند فایل‌های موجود در دایرکتوری را با مجوزهای مشخص شده برای لایه‌ی گروه آن فایل، اجرا کنند. استفاده از SGID در کارهایی نظیر File Sharing کاربرد دارد.

برای این که به یک فایل یا دایرکتوری مجوز SGID بدهیم، می‌توانیم از دستور chmod استفاده کنیم:

- در حالت Symbolic، کاراکتر s را به مجوزهای گروه دایرکتوری (یا فایل) اضافه می‌کنیم:

```
[root@localhost ~]# ls -l
total 0
drwxr-xr-x. 2 root root 6 Jul 29 12:53 my_directory
[root@localhost ~]# chmod g+s my_directory/
[root@localhost ~]# ls -l
total 0
drwxr-sr-x. 2 root root 6 Jul 29 12:53 my_directory
```

همانطور که می‌بینید، پس از اعمال SGID، در قسمت مجوز اجرای لایه‌ی گروه این فایل، کاراکتر s مشاهده می‌شود و این یعنی که این دایرکتوری، مجوز SGID دارد.

- در حالت Octal، یک رقم به ابتدای ۳ رقم مجوزی که قبلا یاد گرفتیم، اضافه می‌کنیم. در این حالت، اولین رقم، مشخص کننده‌ی آن مجوز خاص می‌باشد و رقم دوم، سوم و چهارم، به ترتیب مجوز مالک، گروه و سایرین را مشخص می‌کنند. از آنجایی که در اینجا می‌خواهیم مجوز خاص SGID را به سیستم بدهیم، از عدد ۲ به عنوان رقم اول استفاده می‌کنیم:

```
[root@localhost ~]# ls -l
total 0
drwxr-xr-x. 2 root root 6 Jul 29 12:57 my_directory
[root@localhost ~]# chmod 2775 my_directory/
[root@localhost ~]# ls -l
total 0
drwxrwsrx. 2 root root 6 Jul 29 12:57 my_directory
```

همانطور که می‌بینید، پس از اعمال SGID، در قسمت مجوز اجرای لایه‌ی گروه این دایرکتوری، کاراکتر s مشاهده می‌شود.

Sticky Bit

Sticky Bit مجوزی است که روی دایرکتوری‌ها اعمال می‌شود. این مجوز، باعث می‌شود که فایل‌های موجود درون دایرکتوری، فقط توسط مالک فایل قابل حذف باشند یا به عبارت دیگر، این مجوز باعث می‌شود که هیچ‌کسی به جز مالک فایل، نتواند فایل موجود در آن دایرکتوری را پاک کند. معمولا از Sticky Bit بر روی دایرکتوری‌هایی که بین یک گروه به اشتراک گذاشته شده استفاده می‌شود. در این حالت اعضای گروه مجوز خواندن یا نوشتن روی فایل‌های داخل دایرکتوری را خواهند داشت، اما فقط مالک فایل می‌تواند آن فایل را پاک کند. دایرکتوری /tmp در سیستم‌های لینوکسی، دارای مجوز Sticky Bit می‌باشد:

```
[root@localhost ~]# ls -ld /tmp/
drwxrwxrwt. 11 root root 233 Jul 26 11:14 /tmp/
```

همانطور که می‌بینید، مجوز Sticky Bit با کاراکتر t مشخص می‌شود و جایگزین مجوز اجرا در لایه‌ی سایرین می‌شود.

برای این که Sticky Bit را به یک دایرکتوری بدهیم، می‌توانیم از دستور chmod استفاده کنیم:

- در حالت Symbolic، به مجوزهای لایه‌ی سایرین، حرف t را اضافه می‌کنیم:

```
[root@localhost ~]# chmod o+t my_directory/
[root@localhost ~]# ls -l
total 0
drwxr-xr-t. 2 root root 6 Jul 26 11:36 my_directory
```

همانطور که می‌بینید، پس از اعمال Sticky Bit، در قسمت مجوز اجرای لایه‌ی سایرین این دایرکتوری، کاراکتر t مشاهده می‌شود . این یعنی که این دایرکتوری مجوز Sticky Bit دارد.

- در حالت Octal، یک رقم به ابتدای ۳ رقم مجوزی که قبلا یاد گرفتیم، اضافه می‌کنیم. در این حالت، اولین رقم، مشخص کننده‌ی آن مجوز خاص می‌باشد و رقم دوم، سوم و چهارم، به ترتیب مجوز مالک، گروه و سایرین را مشخص می‌کنند. از آنجایی که در اینجا می‌خواهیم مجوز خاص Sticky Bit را به دایرکتوری بدهیم، از عدد ۱ به عنوان رقم اول استفاده می‌کنیم:

```
[root@localhost ~]# chmod 1775 my_directory/
[root@localhost ~]# ls -l
total 0
drwxrwxr-t. 2 root root 6 Jul 26 11:37 my_directory
```

همانطور که می‌بینید، پس از اعمال Sticky Bit، در قسمت مجوز اجرای لایه‌ی سایرین این دایرکتوری، کاراکتر t مشاهده می‌شود.

نکته: استفاده‌ی نابه‌جا از مجوزهای ویژه‌ی SUID و GUID می‌تواند سبب به وجود آمدن ضعف‌های امنیتی در سیستم شود. با این که برخی از برنامه‌ها نظیر passwd، برای عملکرد صحیح، به داشتن این مجوزهای ویژه نیاز دارند، اما اکثر برنامه‌ها نیازی به داشتن این مجوزها ندارد و ما نباید هیچ وقت بدون داشتن یک دلیل قانع کننده، این مجوزها را روی یک فایل یا دایرکتوری قرار دهیم. Sticky Bit به دلیل طبیعتی که دارد به اندازه‌ی SUID و GUID خطرناک نیست، اما باز هم باید یک دلیل مناسب برای قرار دادن آن روی یک دایرکتوری داشته باشیم.

مشخص‌کردن مجوزهای پیش‌فرض

اگر دقت کرده باشید، زمانی که یک فایل یا یک دایرکتوری جدید ایجاد می‌کنیم، این فایل‌ها یک سری مجوز ابتدایی یا پیش‌فرض دارند. ما می‌دانیم که به صورت پیش‌فرض، مالک فایل کسی است که فایل را ایجاد کرده و همچنین گروه فایل، گروه مالک فایل می‌باشد. اما مجوزهای پیش‌فرض از کجا می‌آیند؟

User Mask، مجوزهایی که لینوکس به صورت پیش‌فرض به فایل‌ها یا دایرکتوری‌ها می‌دهد را مشخص می‌کند. در واقع User Mask به لینوکس می‌گوید که هنگام ایجاد فایل‌ها، چه بیت‌های مجوزی را از ارقام ۶۶۶ و هنگام ایجاد دایرکتوری‌ها، چه بیت‌های مجوزی را از ارقام ۷۷۷، حذف کند. بیایید با استفاده از دستور umask، مجوزهای پیش‌فرض فایل‌ها و دایرکتوری‌ها در سیستم را ببینیم:

```
[root@localhost ~]# umask
0022
```

همانطور که می‌بینید، در خروجی این دستور، ٤ رقم به ما نمایش داده می‌شود. برای درک مفهوم این ارقام، باید مفاهیمی که در دو بخش قبل گفتیم را به خاطر آوریم:

- رقم اول، مشخص‌کننده‌ی مجوزهای ویژه (SUID، SGID و Sticky Bit) می‌باشد. صفر، در این قسمت، یعنی هیچکدام از مجوزهای ویژه روی فایل‌های قرار نگیرد.
- رقم دوم مشخص می‌کند که لایه‌ی مالک، چه مجوزهایی را نداشته باشد؛ یا به عبارت دیگر، مشخص می‌کند که چه بیت‌های مجوزی باید از مجوزهای لایه‌ی مالک، حذف شود. در اینجا، صفر یعنی از مجوزهای مالک، هیچ بیت مجوزی حذف نشود، یعنی مجوز پیش‌فرض لایه‌ی مالک هنگام ایجاد فایل، ۶ و هنگام ایجاد دایرکتوری، ۷ می‌باشد.
- رقم سوم مشخص می‌کند که لایه‌ی گروه، چه مجوزهایی را نداشته باشد؛ یا به عبارت دیگر، مشخص می‌کند که چه بیت‌های مجوزی باید از مجوزهای لایه‌ی گروه، حذف شود. در اینجا، رقم ۲ یعنی بیت مجوز ۲، باید از مجوزهای لایه‌ی گروه حذف شود. یعنی مجوز پیش‌فرض لایه‌ی گروه هنگام ایجاد فایل، ٤ و هنگام ایجاد گروه، ۵ خواهد بود. میدانیم که ۲ نشان دهنده‌ی مجوز خواندن می‌باشد، پس به عبارت دیگر می‌توان گفت که ۲، یعنی لایه‌ی گروه مجوز خواندن را نداشته باشد.

- رقم چهارم نیز مشخص کننده‌ی بیت‌هایی می‌باشد که باید از لایه‌ی سایرین حذف شود. در اینجا هم رقم ۲ را می‌بینیم که دقیقا همان مفهومی که در بخش قبلی گفتیم را دارد. پس به طور کلی، فایل‌هایی جدیدی که ایجاد می‌کنیم، دارای مجوز ۶۴۴ (یا ۰۶۴۴) و دایرکتوری‌های جدیدی که ایجاد می‌کنیم، دارای مجوز ۷۵۵ (یا ۰۷۵۵) خواهند بود.

برای تغییر مجوزهای پیش‌فرض نیز از دستور umask استفاده می‌کنیم. ما باید به دستور umask، مقدار octal مجوزهایی که می‌خواهیم حذف شوند را بدهیم. فراموش نکنید که هنگام استفاده از umask، مجوز فایل‌ها از عدد ۶۶۶ در نظر گرفته می‌شوند و مجوز دایرکتوری‌ها از عدد ۷۷۷ در نظر گرفته می‌شود.

نکته: شاید برایتان سوال باشد که چرا مجوز پیش‌فرض فایل‌ها از ۶۶۶ شروع شده، اما مجوز پیش‌فرض دایرکتوری‌ها از ۷۷۷ شروع می‌شود. دلیل این امر، این است که از نظر امنیتی، یک فایل ایجاد شده نباید به صورت پیش‌فرض مجوز اجرا داشته باشد؛ اما این امر در مورد دایرکتوری‌ها صادق نیست، چون اگر بخواهیم وارد یک دایرکتوری شویم، آن دایرکتوری باید مجوز اجرا داشته باشد.

فرض کنید می‌خواهیم کاری کنیم که به صورت پیش‌فرض، لایه‌ی مالک مجوز خواندن و نوشتن را داشته باشد، لایه‌ی گروه فقط مجوز خواندن را داشته باشد و لایه‌ی سایرین، فقط مجوز نوشتن روی فایل را داشته باشد (یعنی rw-r---w-).

ما می‌توانیم این کار را به دو روش انجام دهیم. در روش اول، می‌توانیم صرفا رقم Octal مجوزی که نمیخواهیم فایل‌های جدید داشته باشند را به umask بدهیم. اما ما می‌خواهیم روش دیگری که شاید درک آن راحت‌تر باشد را نیز توضیح دهیم. در این روش، ما ابتدا رقم Octal مربوط به مجوز دلخواه در هر لایه را پیدا می‌کنیم. گفتیم که:

- لایه‌ی مالک باید مجوز خواندن و نوشتن داشته باشد. پس رقم Octal مربوط به آن ۶ خواهد شد (۲+۴).
- لایه‌ی گروه باید فقط مجوز خواندن را داشته باشد، پس رقم Octal مربوط به آن، ۴ خواهد شد.
- لایه‌ی سایرین باید فقط مجوز نوشتن را داشته باشد، پس رقم Octal مربوط به آن ۲ خواهد شد.

پس ما می‌خواهیم فایل‌های جدید، مجوز پیش‌فرض ۶۴۲ داشته باشند. اما گفتیم که umask فقط بیت‌های مجوزی که باید حذف شوند را از ما میگیرد. برای پیدا کردن بیت‌های مجوزی که باید حذف شوند، عدد ۶۴۲ را از ۶۶۶ کسر میکنیم. یعنی:

۶۶۶-۶۲۴=۰۲۴

پس، بیت‌های مجوزی که باید حذف شوند، ۰ برای لایه‌ی مالک، ۲ برای لایه‌ی گروه و ۴ برای لایه‌ی سایرین می‌باشد. پس:

```
[root@localhost ~]# umask 024
```

حال هر فایلی که ایجاد کنیم، مجوزهای rw-r---w- را خواهند داشت:

```
[root@localhost ~]# touch new_file
[root@localhost ~]# ls -l
total 0
-rw-r---w-. 1 root root 0 Jul 30 12:00 new_file
```

این امر روی مجوز دایرکتوری‌های جدید نیز تاثیر می‌گذارد. یعنی اگر الان یک دایرکتوری جدید ایجاد کنیم، مجوزهای rwx-r-x-wx را خواهد داشت، چون همانطور که گفتیم، دایرکتوری‌ها به صورت پیش‌فرض، باید مجوز اجرا داشته باشند، وگرنه کسی نمی‌تواند وارد دایرکتوری شود. یعنی مجوز دایرکتوری‌های جدید، ۷۵۳ خواهد بود:

```
[root@localhost ~]# mkdir directory
[root@localhost ~]# ls -l
total 0
drwxr-x-wx. 2 root root 6 Jul 27 12:48 directory
```

اگر هنوز درک این قضیه برایتان دشوار است، در جدول ۶، برخی از umaskهای معروف و تاثیر آنها بر روی مجوزهای پیش‌فرض فایل و دایرکتوری را می‌بینید.

جدول ۶- برخی از umaskهای معروف و تاثیر آنها بر مجوزهای پیش‌فرض

umask	مجوز پیش‌فرض فایل	مجوز پیش‌فرض دایرکتوری
000	666 (rw-rw-rw-)	777 (rwxrwxrwx)
002	664 (rw-rw-r--)	775 (rwxrwxr-x)
022	644 (rw-r--r--)	755 (rwxr-xr-x)
027	640 (rw-r-----)	750 (rwxr-x---)
077	600 (rw-------)	700 (rwx------)
277	400 (r--------)	500 (r-x------)

نکته: مواردی که تا به اینجا در مورد مجوزها در لینوکس یاد گرفتیم، از ابتدای عمر لینوکس تا الان در این سیستم‌ها وجود داشته و عملکرد قابل قبولی نیز از خود ارائه داده‌اند. این مجوزها، از نوع Discretionary Access Control یا DAC می‌باشند. از نظر امنیتی، مجوزهای DAC برای امن‌سازی یک سیستم لینوکسی، کافی نیستند. شاید بتوان گفت Access Control List یا ACLها، انتخاب بهتری برای مدیریت مجوزها در لینوکس می‌باشند. ACL لیستی از کاربران یا گروه‌ها می‌باشد که مجوزهای مربوط به هر کاربر یا گروه در برخورد با یک رویداد، در آن مشخص شده است. برای اطلاعات بیشتر در مورد ACLها، به manpage دستور `setfacl` و `getfacl` مراجعه کنید.

شاید بتوان گفت بهترین رویکرد برای مدیریت مجوزها و امنیت سیستم، استفاده از مدل Mandatory Access Control یا MAC، مخصوصا زیرمجموعه‌ی آن، یعنی Role-based Access Control یا RBAC می‌باشد. این مدل‌ها، با استفاده از ابزار SELinux که روی اکثر سیستم‌های لینوکسی موجود می‌باشند، پیاده‌سازی می‌شوند. SELinux ابزار بسیاری پیچیده‌ای می‌باشد و صحبت در مورد آن از محدوده و حوصله‌ی درس ما خارج می‌باشد.