# Linux Professional Institute

## LPIC-1

## جلسه ششم: پارتیشن‌بندی، مدیریت پارتیشن‌ها و مدیریت فایل‌ها

### در این جلسه:

**ویدئو اول:**

**ویدئو دوم:**

- آشنایی با دستور `df` و `du` برای مدیریت حجم پارتیشن‌ها و دایرکتوری‌ها/فایل‌ها
- آشنایی با دستور `mount` و `umount` و فایل `fstab`
- آشنایی با ملاحظات موجود در نام‌گذاری فایل‌ها و مفهوم Wildcard Expansion
- آشنایی با دستورات `pwd`، `ls`، `cp` و `mv`
- آشنایی با دستور `rm` و آپشن‌های آن
- آشنایی با مفهوم آرشیوکردن و فشرده‌سازی با استفاده از ابزار `tar`

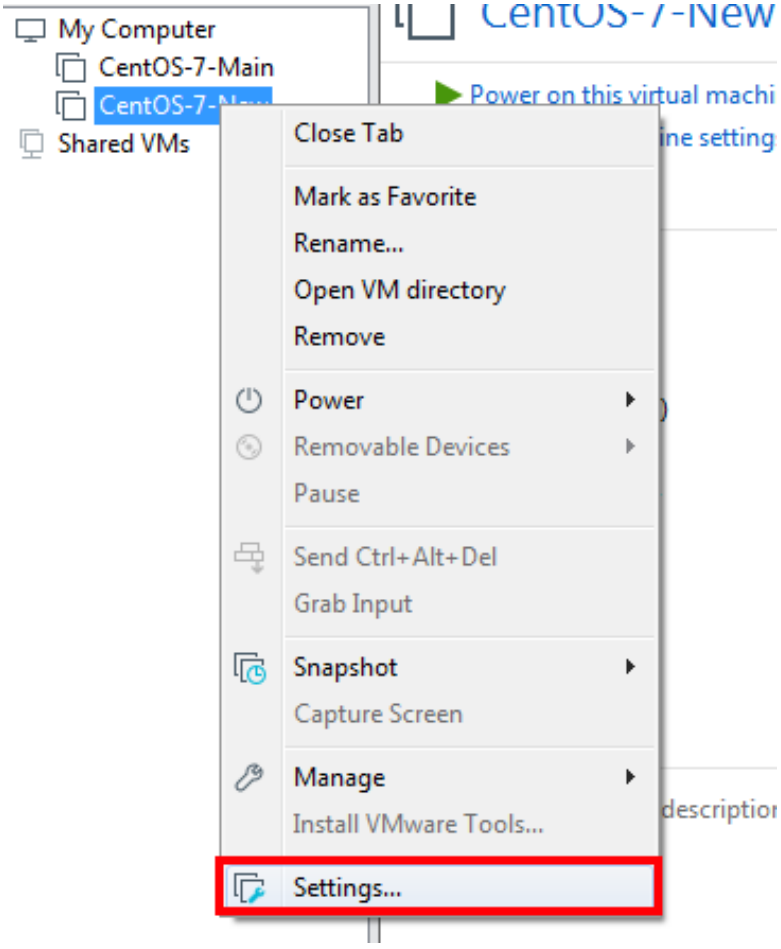
# فهرست مطالب

## مقدمه

جلسه‌ی قبل در مورد دستگاه‌های ذخیره‌سازی داده و چگونگی برخود لینوکس با آنها صحبت کردیم. سپس چگونگی پارتیش‌بندی هنگام نصب یک سیستم را یاد گرفتیم و با برخی از دستورهایی که وضعیت پارتیشن‌های یک سیستم را نشان می‌دادند، آشنا شدیم. در این جلسه، در مورد چگونگی پارتیشن‌بندی یک سیستم پس از نصب صحبت می‌کنیم و سپس به سراغ چگونگی مدیریت پارتیشن‌ها می‌رویم. پس از آن، در مورد مدیریت فایل‌ها در لینوکس صحبت می‌کنیم و با مواردی نظیر چگونگی ایجاد فایل و دایرکتوری، فشرده‌سازی فایل‌ها و مفهوم آرشیو، آشنا می‌شویم.

## اضافه کردن یک هارد دیسک به ماشین مجازی

جلسه‌ی قبل یک ماشین مجازی جدید ایجاد کردیم و سیستم‌عامل را هنگام نصب، پارتیشن‌بندی کردیم. قبل از ورود به مباحث مربوط به این جلسه، می‌خواهیم یک هارد دیسک جدید به ماشین مجازی اضافه کنیم. برای این کار، روی ماشین مجازی که جلسه قبل ایجاد کردیم (یا هر ماشین دیگر) راست‌کلیک کرده و روی گزینه‌ی Settings کلیک می‌کنیم (ابتدا از خاموش بودن ماشین مجازی اطمینان حاصل می‌کنیم):

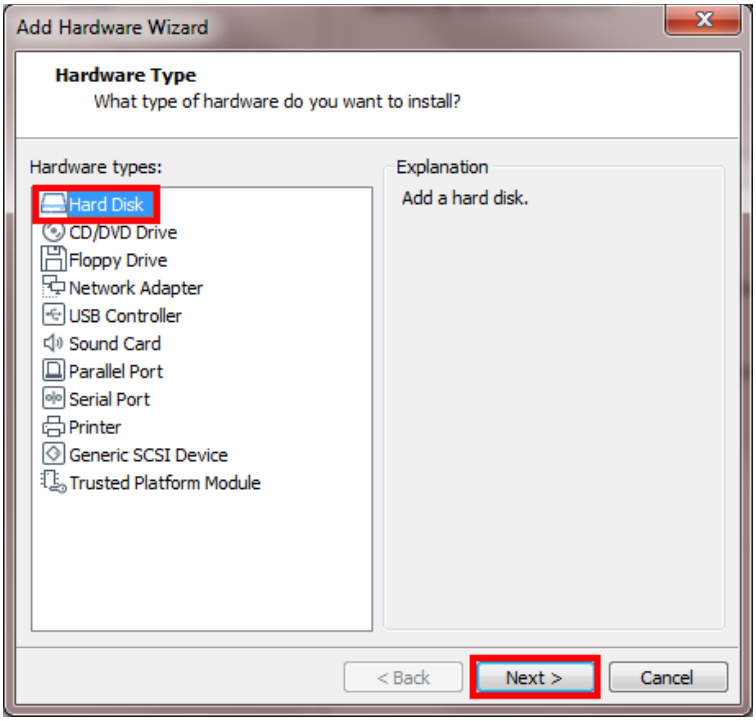



*تصویر ۱ - کلیک روی گزینه‌ی Settings جهت ورود به صفحه‌ی تنظیمات ماشین مجازی*

به محض کلیک روی گزینه‌ی Settings، با نمایی نظیر تصویر ۲ مواجه می‌شویم. برای اضافه کردن یک سخت‌افزار جدید، کافی است روی دکمه‌ی Add کلیک کنیم:

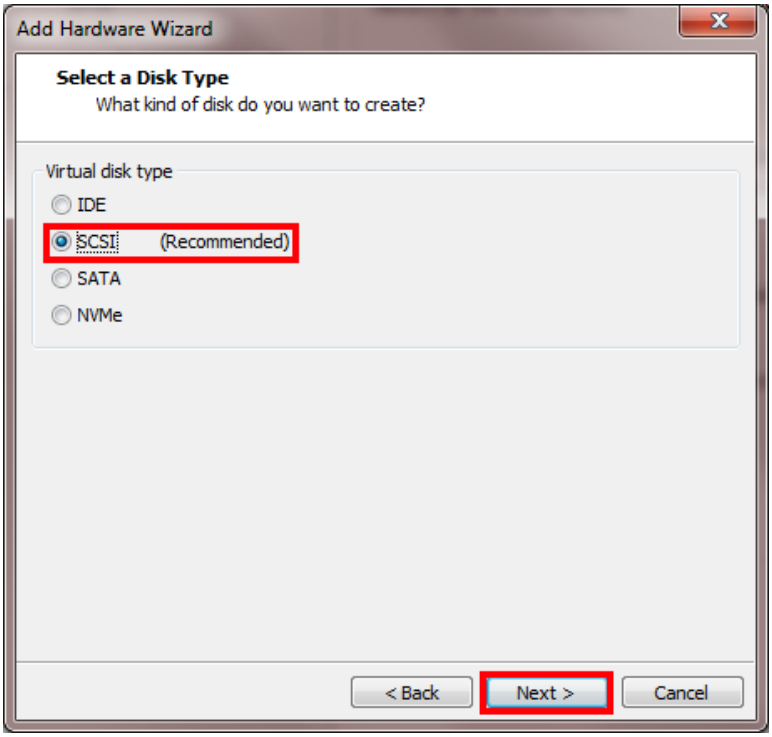

*تصویر ۲- کلیک روی دکمه‌ی Add جهت اضافه کردن یک سخت‌افزار جدید به سیستم*

پس از کلیک روی دکمه‌ی Add، با صفحه‌ی زیر مواجه می‌شویم. ما می‌خواهیم یک هارددیسک به سیستم اضافه کنیم؛ پس گزینه‌ی Hard Disk را انتخاب می‌کنیم و سپس روی دکمه‌ی Next کلیک می‌کنیم:



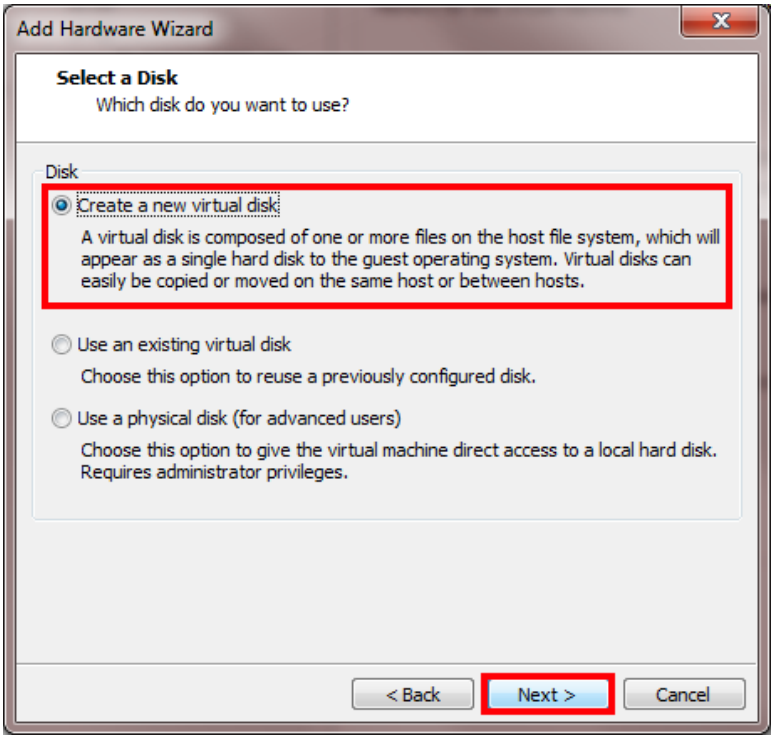

*تصویر ۳- انتخاب گزینه‌ی هارددیسک جهت اضافه کردن یک هارددیسک جدید به سیستم*

در صفحه‌ی بعد، باید چگونگی اتصال هارددیسک به ماشین مجازی را مشخص کنیم. ما گزینه‌ی SCSI را انتخاب می‌کنیم. انتخاب خود در این مرحله را به خاطر داشته باشید، چون طریقه‌ی اتصال هارددیسک به سیستم، بر روی نام دیوایس‌فایل آن دستگاه در لینوکس، تاثیر می‌گذارد. پس از انتخاب این گزینه، بر روی دکمه‌ی Next کلیک می‌کنیم:



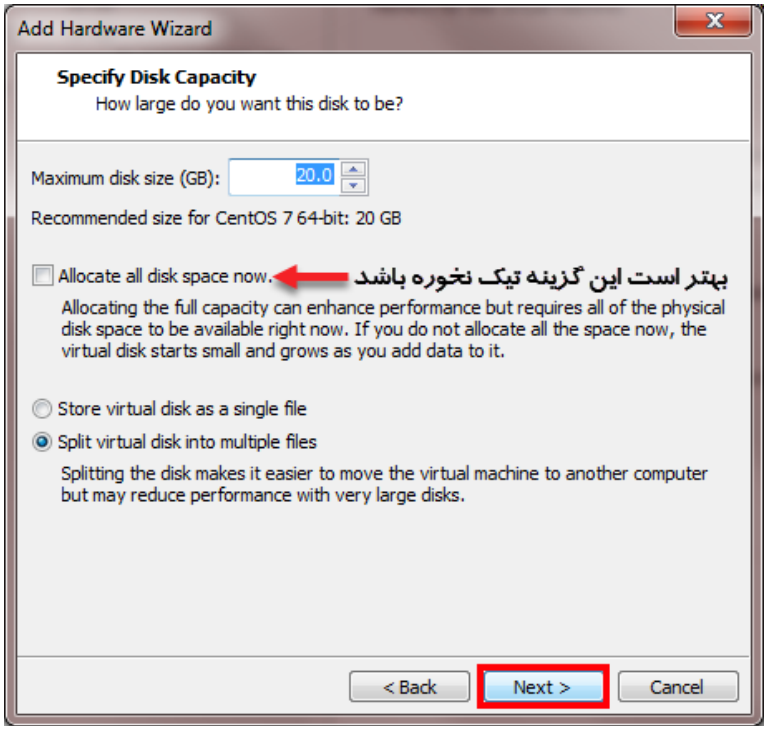

*تصویر ٤- انتخاب گزینه‌ی SCSI جهت اتصال هارددیسک به ماشین مجازی*

سپس در صفحه‌ی ظاهر شده، گزینه‌ی Create a new virtual disk را انتخاب کرده و سپس دکمه‌ی Next را می‌زنیم:



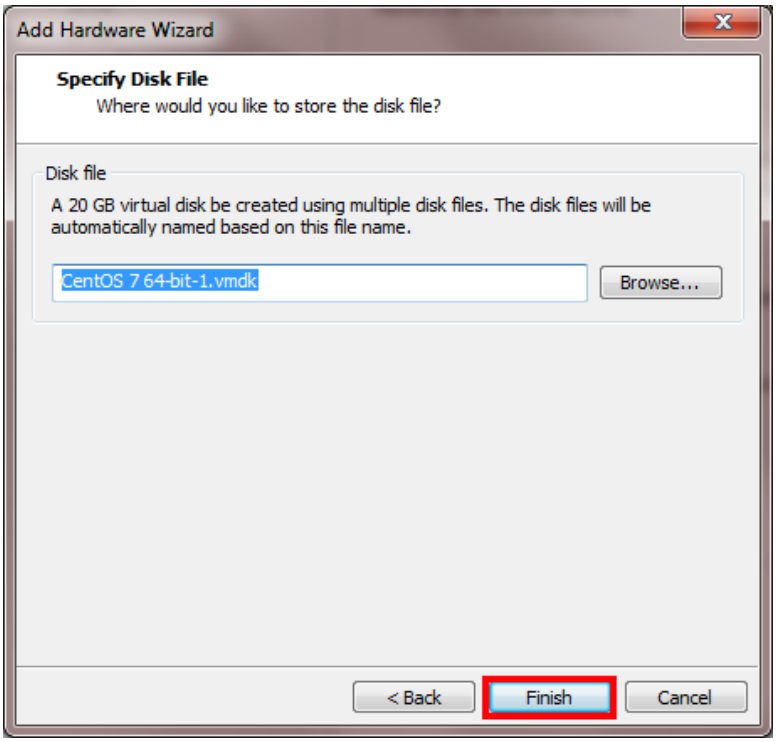

*تصویر ٥- انتخاب گزینه‌ی Create a new virtual disk و سپس انتخاب گزینه‌ی Next*

در مرحله‌ی بعد، می‌توانیم حجم این هارددیسک جدید را مشخص کنیم. ما به مقدار پیش‌فرض دستی نمی‌زنیم. در این صفحه، ابتدا از عدم تیک داشتن گزینه‌ی Allocate all disk space now مطمئن می‌شویم و سپس روی دکمه‌ی Next کلیک می‌کنیم:



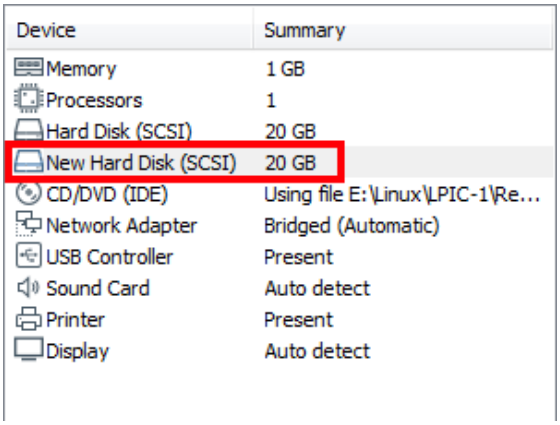

*تصویر ۶- انتخاب حجم هارددیسک جدید*

در مرحله‌ی بعد می‌توایم محل ذخیره و همچنین نام این هارددیسک جدید بر روی سیستم خودمان را مشخص کنیم. ما به این گزینه دستی نمی‌زنیم و آن را در حالت پیش‌فرض خود رها می‌کنیم و در نهایت روی گزینه‌ی Finish کلیک می‌کنیم:




*تصویر ۷- انتخاب محل ذخیره‌ی هارددیسک جدید*

در نهایت، اگر همه چیز به درستی پیش رفته باشد، باید یک هارددیسک جدید در بخش Settings ماشین مجازی اضافه شده باشد:




*تصویر ۸- هارددیسک جدیدی که به ماشین مجازی اضافه کردیم.*

حال کافی است ماشین مجازی را روشن کنیم و نگاهی به هارددیسک‌های موجود در سیستم بیاندازیم:

```
[root@localhost ~]# lsblk
NAME        MAJ:MIN     RM    SIZE  RO    TYPE  MOUNTPOINT
sda         8:0         0     20G   0     disk
├─sda1      8:1         0     953M  0     part  /boot
├─sda2      8:2         0     4.7G  0     part  /var
├─sda3      8:3         0     3.7G  0     part  /home
├─sda4      8:4         0     1K    0     part
├─sda5      8:5         0     1.9G  0     part  [SWAP]
└─sda6      8:6         0     8.8G  0     part  /
sdb         8:16        0     20G   0     disk
sr0         11:0        1     942M  0     rom
```

همانطور که می‌بینید، هارددیسک جدید با حجم ۲۰ گیبی‌بایت و با نام sdb، به سیستم ما متصل شده است. همانطور که گفتیم، به دلیل اتصال هارددیسک از طریق SCSI به ماشین مجازی، نام این هارددیسک با sd شروع می‌شود و از آنجایی که دومین هارددیسک SCSI متصل به سیستم می‌باشد، با حرف b تمام می‌شود.

حال بیایید نگاهی به خروجی دستور `fdisk -l` بیاندازیم و ببینیم وضعیت این هارددیسک چگونه است. از آنجایی که فقط می‌خواهیم اوضاع پارتیشن‌بندی این هارددیسک جدید را ببینیم، آدرس این هارددیسک جدید را نیز به `fdisk` می‌دهیم:

```
[root@localhost ~]# fdisk -l /dev/sdb
Disk /dev/sdb: 21.5 GB, 21474836480 bytes, 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

همانطور که می‌بینید این هارددیسک هنوز پارتیشن‌بندی نشده است و جدول پارتیشن، ندارد. پس ما باید از طریقی این هارددیسک جدید را پارتیشن‌بندی کنیم.

## پارتیشن‌بندی یک هارددیسک با استفاده از *fdisk*

پس از اتصال یک هارددیسک جدید به سیستم، باید آن را پارتیشن‌بندی کنیم (حتی اگر بخواهیم فقط یک پارتیشن داشته باشیم)؛ چرا که در صورت عدم وجود جدول پارتیشن، سیستم قابلیت تعامل با آن هارددیسک را نخواهد داشت همانطور که گفتیم، fdisk یکی از معروف‌ترین ابزارها برای پارتیشن‌بندی هارددیسک می‌باشد. این برنامه به ما امکان می‌دهد که هارددیسک‌های سیستم را پارتیشن‌بندی کنیم، پارتیشن‌های موجود را تغییر دهیم، پارتیشن‌ها را حذف کنیم و همچنین جدول پارتیشن را مشاهده کنیم (که با این قابلیت آن آشنا هستیم).

برای استفاده از fdisk جهت پارتیشن‌بندی، باید نام دیوایس‌فایل هارددیسک مورد نظر را به دستور fdisk بدهیم:

```
[root@localhost ~]# fdisk /dev/sdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table
Building a new DOS disklabel with disk identifier 0x6e1f55a9.

Command (m for help):
```

برنامه‌ی fdisk، دستورات منحصر به خود را دارد. ما در جلوی Command، می‌توانیم دستورات متفاوتی را بنویسیم. این دستورها، فقط یک حرف از حروف الفبا می‌باشند. همانطور که می‌بینید، با نوشتن حرف m و زدن دکمه‌ی Enter، می‌توانیم لیستی از دستورها و عملکرد آنها را ببینیم:

```
Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   g   create a new empty GPT partition table
   G   create an IRIX (SGI) partition table
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

Command (m for help):
```

برای مثال با وارد کردن حرف p و زدن دکمه‌ی Enter، می‌توانیم جدول پارتیشن هارددیسک /dev/sdb2 (که در حال حاضر خالی می‌باشد) را ببینیم:

```
Command (m for help): p

Disk /dev/sdb: 21.5 GB, 21474836480 bytes, 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x6e1f55a9

   Device Boot      Start         End      Blocks   Id  System

Command (m for help):
```

برای ایجاد یک پارتیشن جدید، حرف n را وارد کرده و دکمه‌ی Enter را می‌زنیم:

```
Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p):
```

در این قسمت، ما باید نوع پارتیشنی که می‌خواهیم ایجاد کنیم را مشخص کنیم. پارتیشن می‌تواند از نوع Primary یا Extended باشد. جلسه‌ی قبل به اندازه‌ی کافی در مورد مفهوم این عبارات در سیستم MBR صحبت کردیم. ما در اینجا می‌خواهیم یک پارتیشن Primay ایجاد کنیم، پس حرف p را وارد کرده و سپس دکمه‌ی Enter را می‌زنیم (البته مجبور نیستیم حرف p را وارد کنیم، اگر فقط Enter را بزنیم، به صورت پیش‌فرض پارتیشن به صورت Primary ایجاد می‌شود):

```
Select (default p): p
Partition number (1-4, default 1):
```

حال باید شماره‌ی این پارتیشن را انتخاب کنیم. همانطور که گفتیم در شماره‌گذاری پارتیشن‌های Primary می‌توانیم فاصله‌ی خالی داشته باشیم. پس در اینجا می‌توانیم هر عددی بین ۱ تا ٤ قرار دهیم. ما در این قسمت عدد ۱ را وارد می‌کنیم و دکمه‌ی Enter را می‌زنیم:

```
Partition number (1-4, default 1): 1
First sector (2048-41943039, default 2048):
```

در اینجا باید آدرس اولین سکتوری که این پارتیشن از آن شروع می‌شود را وارد کنیم. این عدد می‌تواند از ۲۰٤۸ شروع شود. دلیل این امر را در جلسه‌ی قبل توضیح دادیم. ما همان عدد ۲۰٤۸ را وارد می‌کنیم و دکمه‌ی Enter را می‌زنیم:

```
First sector (2048-41943039, default 2048): 2048
Last sector, +sectors or +size{K,M,G} (2048-41943039, default 41943039):
```

سپس باید سکتوری که پارتیشن در آن پایان می‌یابد را انتخاب کنیم. خوشبختانه در اینجا مجبور نیستیم شماره‌ی یک سکتور را بنویسیم و می‌توانیم فضایی که قصد داریم به این پارتیشن بدهیم را در واحد کیبی‌بایت، مبی‌بایت یا گیبی‌بایت بنویسیم، یا حتی می‌توانیم بگوییم که چند سکتور باید به این پارتیشن اختصاص یابد. برای این‌کار، کافی است علامت + را به همراه عدد مورد نظر وارد کرده و یکی از واحدهای K، M یا G را در آخر آن قرار دهیم. اگر واحدی به عدد وارد شده اختصاص ندهیم، به تعداد ذکر شده، سکتور به پارتیشن اضافه می‌شود. در ضمن اگر عددی در این قسمت وارد نکنیم، کل فضای موجود در هارددیسک به این پارتیشن اختصاص داده می‌شود.

ما به این پارتیشن، ۵ گیبی‌بایت فضا می‌دهیم:

```
Last sector, +sectors or +size{K,M,G} (2048-41943039, default 41943039): +5G
Partition 1 of type Linux and of size 5 GiB is set

Command (m for help):
```

همانطور که می‌بینید، برنامه‌ی fdisk یک پارتیشن پنج گیبی‌بایتی (معادل ۵,۳۶ گیگابایت) با شماره پارتیشن ۱، بر روی هارددیسک sdb ایجاد کرد.

حال بیایید با وارد کردن حرف p و زدن دکمه‌ی Enter، وضعیت جدول پارتیشن را بررسی کنیم:

```
Command (m for help): p
Disk /dev/sdb: 21.5 GB, 21474836480 bytes, 41943040 sectors
```

```
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xb4f019ca

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048    10487807     5242880   83  Linux
```

**Command (m for help):**

همانطور که می‌بینید، پارتیشن جدید ما با دیوایس‌فایلی به نام /dev/sdb1 ایجاد شده است. بیایید این بار یک پارتیشن Extended ایجاد کنیم:

```
Command (m for help): n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): e
Partition number (2-4, default 2): 2
First sector (10487808-41943039, default 10487808):
Using default value 10487808
Last sector, +sectors or +size{K,M,G} (10487808-41943039, default 41943039): +5G
Partition 2 of type Extended and of size 5 GiB is set

Command (m for help): p
Disk /dev/sdb: 21.5 GB, 21474836480 bytes, 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xb4f019ca

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048    10487807     5242880   83  Linux
/dev/sdb2        10487808    20973567     5242880    5  Extended
```

همانطور که می‌بینید، ما پارتیشن دوم را از نوع Extended ایجاد کردیم. حال می‌توانیم این پارتیشن Extended را تبدیل به بی‌نهایت پارتیشن منطقی (Logical) کنیم. بیایید این کار را امتحان کنیم:

```
Command (m for help): n
Partition type:
   p   primary (1 primary, 1 extended, 2 free)
   l   logical (numbered from 5)
```

پارتیشن‌های Logical از شماره‌ی ۵ شروع می‌شوند و ما نمی‌توانیم بین شماره‌ی آنها، فاصله‌ای قرار دهیم. بیایید حرف l را به منظور ایجاد پارتیشن Logical انتخاب کنیم:

```
Select (default p): l
Adding logical partition 5
```

**First sector (10489856-20973567, default 10489856):**

در اینجا ما باید شماره‌ی سکتور آغاز کننده‌ی اولین پارتیشن Logical را انتخاب کنیم. اما ابتدا به محدوده‌ی سکتورها نگاهی بیاندازید. همانطور که می‌بینید ما می‌توانیم سکتوری بین ۱۰۴۸۹۸۵۶ تا ۲۰۹۷۳۵۶۷ انتخاب کنیم. این تعداد سکتور، دقیقا معادل ۵ گیبی‌بایت می‌باشد. این بدین معناست که پارتیشن‌های Logical، به هیچ عنوان نمی‌توانند بزرگتر از پارتیشن Extended شوند.

ما به این پارتیشن Logical، ۲ گیبی‌بایت فضا می‌دهیم:

```
First sector (10489856-20973567, default 10489856):
Using default value 10489856
Last sector, +sectors or +size{K,M,G} (10489856-20973567, default 20973567): +2G
Partition 5 of type Linux and of size 2 GiB is set

Command (m for help): p

Disk /dev/sdb: 21.5 GB, 21474836480 bytes, 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xb4f019ca

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048    10487807     5242880   83  Linux
/dev/sdb2        10487808    20973567     5242880    5  Extended
/dev/sdb5        10489856    14684159     2097152   83  Linux

Command (m for help):
```

همانطور که می‌بینید پارتیشن‌های Extended و Logical ما به درستی ایجاد شده‌اند. یادآوری می‌کنیم که ما فقط می‌توانیم یک پارتیشن Extended داشته باشیم و سایر پارتیشن‌ها باید از نوع Primary (در اینجا فقط ۲ پارتیشن Primary دیگر می‌توانیم داشته باشیم) یا Logical باشند.

معمولا به خاطر منظم بودن، پارتیشن چهارم را تبدیل به پارتیشن Extended می‌کنند و پارتیشن‌های ۱، ۲ و ۳ را به پارتیشن‌های Primary اختصاص می‌دهند. پس بیایید این پارتیشن Extended که ایجاد کردیم را پاک کنیم.

برای این کار باید حرف d را وارد کرده و سپس دکمه‌ی Enter را بزنیم:

```
Command (m for help): d
Partition number (1,2,5, default 5):
```

سپس باید شماره‌ی پارتیشنی که می‌خواهیم حذف شود را وارد کنیم. اگر شماره‌ی پارتیشن Extended را وارد کنیم، کلیه‌ی پارتیشن‌های Logical نیز به همراه آن پاک می‌شوند. البته ما می‌توانیم پارتیشن‌های Logical را به صورت تکی نیز پاک کنیم، اما از آنجا که ما در اینجا می‌خواهیم پارتیشن Extended را پاک کنیم، عدد ۲ را وارد کرده و دکمه‌ی Enter را می‌زنیم:

```
Partition number (1,2,5, default 5): 2
Partition 2 is deleted
```

حال بیایید نگاهی به جدول پارتیشن بیاندازیم:

```
Command (m for help): p
Disk /dev/sdb: 21.5 GB, 21474836480 bytes, 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xb4f019ca

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048    10487807     5242880   83  Linux
```

همانطور که می‌بینید، پارتیشن Extended و همچنین پارتیشن‌های Logical زیرمجموعه‌ی آن به صورت کامل پاک شدند.

نکته‌ی قابل توجه این است که تا به اینجا هیچ‌کدام از پارتیشن‌هایی که ایجاد کردیم بر روی هارددیسک ذخیره نشده‌اند و اگر در اینجا با نوشتن دستور exit از fdisk خارج شویم، کلیه‌ی پارتیشن‌هایی که تا به اینجا ایجاد کرده‌ایم، پاک خواهند شد (یا به صورت دقیق‌تر، پارتیشن‌ها ذخیره نخواهند شد).

برای ذخیره‌ی پارتیشن‌بندی، کافی است حرف w را وارد کرده و دکمه‌ی Enter را بزنیم:

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

همانطور که می‌بینید با انجام این کار، اطلاعات پارتیشن‌بندی ما ذخیره و جدول پارتیشن ایجاد شد. برای اطمینان از این امر، می‌توانیم دستور fdisk -l یا lsblk را اجرا کنیم:

```
[root@localhost ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   20G  0 disk
├─sda1   8:1    0  953M  0 part /boot
├─sda2   8:2    0  4.7G  0 part /var
├─sda3   8:3    0  3.7G  0 part /home
├─sda4   8:4    0    1K  0 part
├─sda5   8:5    0  1.9G  0 part [SWAP]
└─sda6   8:6    0  8.8G  0 part /
sdb      8:16   0   20G  0 disk
└─sdb1   8:17   0    5G  0 part
sr0     11:0    1  942M  0 rom
```

همانطور که می‌بینید، پارتیشن sdb1 هیچ Mount Point ندارد، یا به عبارت دیگر بر روی هیچ دایرکتوری Mount نشده و در نتیجه ما نمی‌توانیم در حال حاضر از آن استفاده کنیم. در جلسه‌ی قبل گفتیم که برای این که بتوانیم از یک پارتیشن استفاده کنیم، باید دیوایس‌فایل آن پارتیشن را در یک دایرکتوری، مانت کنیم. بیایید این پارتیشن را در موقعیت /mnt مانت کنیم و کمی از آن استفاده کنیم. ما می‌توانیم با استفاده از دستور mount، این کار را انجام دهیم. در بخش‌های بعدی به طور کامل در مورد این دستور صحبت می‌کنیم، اما فعلا این دستور را به علاوه‌ی دیوایس‌فایل پارتیشن مورد نظر و سپس موقعیت Mount Point، وارد ترمینال می‌کنیم. به صورت زیر:

```
[root@localhost ~]# mount /dev/sdb1 /mnt
mount: /dev/sdb1 is write-protected, mounting read-only
mount: unknown filesystem type '(null)'
[root@localhost ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
...
sdb      8:16   0   20G  0 disk
└─sdb1   8:17   0    5G  0 part
sr0     11:0    1  942M  0 rom
```

همانطور که می‌بینید، mount نتوانست این پارتیشن را در /mnt مانت کند، و ظاهرا دلیل این امر، این است که پارتیشن sdb1 هیچ فایل‌سیستمی ندارد. پس بیایید کمی با فایل‌سیستم‌ها آشنا شویم.

## مفهوم فایل‌سیستم‌ها و انواع آنها

همانطور که قرار دادن یک سری کتاب در کتابخانه احتیاج به یک سیستم چیدمان برای مرتب ماندن دارد (قرار دادن کتاب‌ها بر اساس حروف الفبا، سال چاپ، رنگ جلد 😐)، قرار دادن فایل‌ها بر روی هارددیسک نیز احتیاج به یک سیستم برای مرتب ماندن دارد. لینوکس با استفاده از فایل‌سیستم‌ها، فایل‌های موجود برروی هارددیسک را مدیریت می‌کند. برای درک بهتر، می‌توانید فرض کنید که فایل‌سیستم یک نقشه دارد که لینوکس با نگاه کردن به آن، می‌تواند موقعیت هر فایل بر روی سیستم را پیدا کند. همانطور که در بخش قبل گفتیم، برای این که بتوانیم از یک پارتیشن استفاده کنیم یا به عبارت دیگر آن را روی سیستم مانت کنیم، باید حتما به آن یک فایل‌سیستم بدهیم. به همین دلیل، در این بخش با فایل‌سیستم‌های متفاوت آشنا می‌شویم.

### فایل‌سیستم‌های لینوکسی

بسیاری از سیستم‌عامل‌ها، فایل‌سیستم‌های منحصر به خود را دارند و لینوکس نیز از این قاعده مستثنی نیست. اگر قرار باشد از یک پارتیشن یا یک دستگاه ذخیره‌سازی فقط روی سیستم‌های لینوکسی استفاده کنیم، می‌توانیم یکی از فایل‌سیستم‌های زیر که منحصر به لینوکس هستند را انتخاب کنیم:

- **btrfs**

یک فایل‌سیستم جدید و قدرتمند می‌باشد که فایل‌هایی تا حجم ۱۶ اگزبی‌بایت (معادل ۱۸,۴۵ اگزابایت) را می‌تواند درون خود جا دهد. پارتیشن‌هایی که از این فایل‌سیستم استفاده می‌کنند، می‌توانند به صورت حداکثر ۱۶ اگزبی‌بایت حجم داشته باشد. این فایل‌سیستم قابلیت‌های پیشرفته‌ای نظیر امکان snapshot گرفتن از اطلاعات، fault tolerance و همچنین فشرده‌سازی داده را دارد.

- **encryptfs**

این فایل‌سیستم، با استفاده از یکی از پروتکل‌های رمزنگاری سازگار با استاندارد POSIX، فایل‌ها را قبل از قرار دادن روی هارد، رمزگذاری می‌کند. این فایل‌سیستم، فقط توسط سیستم‌عاملی که آن را ایجاد کرده قابل خوانده شدن می‌باشد.

- **ext2**

این فایل‌سیستم که به آن ext2fs نیز می‌گویند، یکی از قدیمی‌ترین فایل‌سیستم‌های لینوکس می‌باشد. این فایل‌سیستم فقط برای استفاده در لینوکس ایجاد شده بود و در دهه‌ی ۹۰ میلادی، معمول‌ترین فایل‌سیستم برای هارددیسک‌ها در لینوکس بود. امروزه زیاد از این فایل‌سیستم استفاده نمی‌شود. این فایل‌سیستم فایل‌هایی تا حجم ۲ تبی‌بایت را پشتیبانی می‌کند و در کل می‌تواند روی پارتیشن‌هایی تا حجم ۳۲ تبی‌بایت اعمال شود.

- **ext3**

این فایل‌سیستم که به آن ext3fs نیز می‌گویند، دقیقا مانند ext2 می‌باشد، با این تفاوت که از قابلیت Journaling برخوردار می‌باشد و همچنین سریع‌تر از ext2 می‌باشد.

**نکته:** فایل‌سیستمی که از قابلیت Journaling برخوردار باشد، یک ژورنال (یا log) از کلیه‌ی تغییراتی در حال اعمال شدن بر روی فایل‌های موجود در پارتیشن می‌باشد، نگهداری می‌کند. این باعث می‌شود که هنگام قطع شدن برق یا crash کردن سیستم، بتوانیم هر گونه خرابی احتمالی به وجود آمده روی فایل‌ها را رفع کنیم.

- **ext4**

این فایل‌سیستم که آن ext4fs نیز می‌گویند، جدیدترین نسخه‌ی فایل‌سیستم‌های ext می‌باشد. این فایل‌سیستم، فایل‌هایی تا حجم ۱۶ تبی‌بایت را پشتیبانی می‌کند و می‌تواند روی پارتیشن‌هایی تا حجم ۱ اگزبی‌بایت اعمال شود. این فایل‌سیستم نیز از ژورنالینگ پشتیبانی می‌کند و در کل عملکرد بهبود یافته‌ای نسبت به سایر فایل‌سیستم‌های ext دارد.

- **ReiserFS**

این فایل‌سیستم قبل از به وجود آمدن ext3 و ext4 ایجاد شده بود و قابلیت‌هایی که اکنون در ext3 و ext4 وجود دارد، نظیر ژورنالینگ را درون خود داشت. امروزه لینوکس دیگر از این فایل‌سیستم پشتیبانی نمی‌کند و در عوض، نسخه‌ی جدیدتر این فایل‌سیستم، Reiser4 را پشتیبانی می‌کند.

- **swap**

این فایل‌سیستم که در جلسه‌ی قبل به آن اشاره کردیم، به ما امکان می‌دهد با استفاده از فضای موجود بر روی هارددیسک، یک حافظه‌ی مجازی (virtual memory) برای سیستم ایجاد کنیم. با این کار، سیستم می‌تواند داده‌های موجود در RAM را داخل فضای swap قرار دهد. لازم به ذکر است که سرعت حافظه‌ی RAM بسیار بالاتر از سرعت هارددیسک می‌باشد و در نتیجه نمی‌توان پارتیشن‌های swap را جایگزین RAM کرد.

فایل‌سیستم پیش‌فرضی که در اکثر سیستم‌های لینوکسی از آن استفاده می‌شود، فایل‌سیستم ext4 می‌باشد؛ چرا که این فایل‌سیستم سریع و قدرتمند است و همچنین قابلیت ژورنالینگ دارد. فایل‌سیستم btrfs نیز امروزه از محبوبیت بالایی برخوردار شده است، چرا که این فایل‌سیستم قابلیت‌های پیشرفته‌ای نظیر قابلیت فشرده‌سازی اتوماتیک داده‌ها و... را دارد.

## فایل‌سیستم‌های غیرلینوکسی

یکی از قابلیت‌های عالی لینوکس، پشتیبانی از فایل‌سیستم‌هایی که مختص یک سیستم‌عامل دیگر هستند، می‌باشد. این امر باعث می‌شود که بتوان اطلاعات را به راحتی بین سیستم‌عامل‌ها به اشتراک گذاشت. برخی از فایل‌سیستم‌های غیرلینوکسی که لینوکس از آنها پشتیبانی می‌کند، به شرح زیر می‌باشند:

- **CIFS**

این فایل‌سیستم که مخفف Common Internet Filesystem می‌باشد، توسط شرکت مایکروسافت جهت خواندن و نوشتن اطلاعات بر روی دستگاه‌های ذخیره‌سازی تحت شبکه، ایجاد شده است. مایکروسافت این فایل‌سیستم را در اختیار عموم قرار داده است و در نتیجه، در همه‌ی سیستم‌عامل‌ها می‌توان از آن استفاده کرد.

- **FAT و exFAT**

این فایل‌سیستم که مخفف File Allocation Table می‌باشد، یکی از ابتدایی‌ترین فایل‌سیستم‌های موجود در دنیای کامپیوتر می‌باشد؛ با این حال، تقریبا توسط همه‌ی سیستم‌عامل‌ها پشتیبانی می‌شود و عملکرد قابل قبولی نیز دارد. این فایل‌سیستم، تنها فایل‌سیستمی بود که توسط سیستم‌عامل DOS و برخی از نسخه‌های اولیه‌ی Windows پشتیبانی می‌شد و به همین دلیل، همه‌ی سیستم‌عامل‌ها می‌توانند این فایل‌سیستم را بخوانند و از آن پشتیبانی کنند. این امر باعث می‌شود که فایل‌سیستم

FAT، بهترین انتخاب برای دستگاه‌های ذخیره‌سازی قابل حمل نظیر فلش مموری و... باشد. البته این فایل‌سیستم محدودیت‌های خود را نیز دارد و شاید بزرگترین محدودیت آن، عدم امکان ذخیره‌ی فایل‌های دارای سایز بیش از ٤ گیبی‌بایت می‌باشد. با این حال، این فایل‌سیستم همچنین از محبوبیت نسبی برخوردار می‌باشد.

- **HFS و HFS+**

این فایل‌سیستم که مخفف Hierarchical Filesystem می‌باشد، توسط شرکت اپل برای استفاده در کامپیوترهای Mac ایجاد شده است. لینوکس می‌تواند پارتیشن‌هایی که از این فایل‌سیستم استفاده می‌کنند را بخواند و همچنین بر روی آنها بنویسد. نسخه‌ی جدیدتر این فایل‌سیستم که HFS+ نام دارد نیز به صورت محدود توسط لینوکس پشتیبانی می‌شود.

- **NFS**

این فایل‌سیستم که مخفف Network Filesystem می‌باشد، یک استاندارد متن‌باز برای خواندن و نوشتن اطلاعات تحت شبکه بر روی دستگاه‌های ذخیره‌سازی شبکه‌ای می‌باشد.

- **NTFS**

این فایل‌سیستم که مخفف New Technology Filesystem می‌باشد، فایل‌سیستمی است که اکثر سیستم‌های ویندوزی از آن استفاده می‌کنند. در سال‌های اخیر و از نسخه‌ی کرنل ۲,۶ به بعد، لینوکس نیز از این فایل‌سیستم به صورت قابل قبولی پشتیبانی می‌کند.

- **SMB**

این فایل‌سیستم که مخفف Server Message Block می‌باشد، یک فایل‌سیستم Proprietary می‌باشد که توسط Microsoft ایجاد شده است. این فایل‌سیستم، برای ارتباط با دستگاه‌های ذخیره‌سازی تحت شبکه و همچنین سایر دستگاه‌های تحت شبکه نظیر پرینترها و... به کار می‌رود. پشتیبانی لینوکس از SMB به این معناست که کامپیوترهای لینوکسی می‌توانند با دستگاه‌ها و سرورهای ویندوزی ارتباط برقرار کنند.

- **UDF**

این فایل‌سیستم که مخفف Universal Disk Format می‌باشد، توسط دستگاه‌های DVD-ROM جهت ذخیره‌ی اطلاعات به کار می‌رود. لینوکس با استفاده از این فایل‌سیستم، هم می‌تواند اطلاعات را از روی DVD بخواند و هم روی آن بنویسد.

- **XFS**

این فایل‌سیستم که مخفف X Filesystem می‌باشد، توسط شرکت Silicon Graphics برای دستگاه‌های Graphical Workstation این شرکت ایجاد شده بود. قابلیت‌های پیشرفته و عملکرد بسیار عالی این فایل‌سیستم، باعث شده که این فایل‌سیستم در دنیای لینوکس، از محبوبیت بالایی برخوردار باشد.

- **ZFS**

این فایل‌سیستم که مخفف Zettabyte Filesystem می‌باشد، توسط شرکت Sun (که اکنون بخشی از Oracle می‌باشد) برای استفاده بر روی سرورها و Workstationهای یونیکسی ایجاد شده بود. این

فایل‌سیستم از نظر عملکرد بسیار شبیه به فایل‌سیستم btrfs می‌باشد.

## ایجاد فایل‌سیستم با استفاده از دستور *mkfs*

برنامه‌ی mkfs، اصلی‌ترین برنامه‌ی لینوکس برای اعمال فایل‌سیستم بر روی یک پارتیشن می‌باشد. کار با این برنامه بسیار ساده می‌باشد و فقط کافی است آپشن t-، نام فایل‌سیستم و آدرس دیوایس‌فایل پارتیشن مورد نظر را به این برنامه بدهیم. بیایید فایل‌سیستم ext4 را به پارتیشن /dev/sdb1 که در بخش قبل ایجاد کردیم، اعمال کنیم:

```
[root@localhost ~]# mkfs -t ext4 /dev/sdb1
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
327680 inodes, 1310720 blocks
65536 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=1342177280
40 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

همانطور که می‌بینید، با استفاده از دستور mkfs، توانستیم به پارتیشن /dev/sdb1، یک فایل‌سیستم اختصاص دهیم. بیایید بار دیگر عملکرد سیستم در مانت کردن این پارتیشن را بررسی کنیم:

```
[root@localhost ~]# mount /dev/sdb1 /mnt/
[root@localhost ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   20G  0 disk
├─sda1   8:1    0  953M  0 part /boot
├─sda2   8:2    0  4.7G  0 part /var
├─sda3   8:3    0  3.7G  0 part /home
├─sda4   8:4    0    1K  0 part
├─sda5   8:5    0  1.9G  0 part [SWAP]
└─sda6   8:6    0  8.8G  0 part /
sdb      8:16   0   20G  0 disk
└─sdb1   8:17   0    5G  0 part /mnt
sr0     11:0    1  942M  0 rom
```

همانطور که می‌بینید، این بار با اجرای دستور mount، سیستم‌عامل هیچ پیغامی به ما نداد. این یعنی سیستم موفق به مانت کردن پارتیشن شده است و پس از مشاهده‌ی خروجی دستور lsblk، می‌بینیم که پارتیشن /sdb1 در مانت‌پوینت یا دایرکتوری /mnt، مانت شده است.

**نکته:** دستور mkfs کلیه‌ی اطلاعات موجود بر روی پارتیشن را پاک می‌کند، پس هنگام ارائه‌ی نام دیوایس‌فایل پارتیشن مورد نظر به این دستور، حواستان را جمع کنید.

# مانت کردن پارتیشن‌ها

همانطور که قبلا هم دیدیم، پس از انتخاب و اعمال فایل‌سیستم به پارتیشن، باید آن را در یک دایرکتوری، مانت کنیم. ما می‌توانیم این کار را به صورت دستی انجام دهیم، یا از لینوکس بخواهیم که هنگام بوت شدن، به صورت اتوماتیک پارتیشن را در موقعیت مورد نظر مانت کند. ما در این بخش در مورد این دو روش مانت صحبت می‌کنیم.

## مانت کردن دستی پارتیشن‌ها

ما قبلا در مورد چگونگی مانت کردن یک پارتیشن صحبت کردیم. همانطور که میدانید، برای استفاده از یک پارتیشن، باید آن پارتیشن را در یک دایرکتوری (که به آن مانت پوینت می‌گویند)، مانت کنیم. ما این کار را با استفاده از دستور `mount` انجام می‌دهیم. در بخش قبل، پارتیشن `/dev/sdb1` را بر روی دایرکتوری `/mnt` مانت کردیم. جالب است بدانید که معمولا از این دایرکتوری برای مانت کردن هارددیسک‌ها و دستگاه‌هایی که قرار است دائما به سیستم متصل باشند استفاده نمی‌شود، اما فعلا ما از این دایرکتوری برای این کار استفاده می‌کنیم. پس برای این کار، ما دستور `mount` را به صورت زیر اجرا کردیم:

```
[root@localhost ~]# mount /dev/sdb1 /mnt
```

نتیجه‌ی این دستور نباید برای ما چیز عجیبی باشد و می‌دانیم که این دستور پارتیشن `/dev/sdb1` را در موقعیت `/mnt` مانت می‌کند. چیزی که در بخش قبل یاد نگرفتیم، این است که لینوکس کلیه‌ی دیوایس‌فایل‌های مانت شده بر روی سیستم و همچنین یک سری اطلاعات جانبی نظیر مانت‌پوینت و... را بر روی فایل `/etc/mtab` ذخیره می‌کند. بیایید نگاهی به محتویات این فایل بیاندازیم:

```
[root@localhost ~]# cat /etc/mtab
...
/dev/sda3 /home xfs rw,seclabel,relatime,attr2,inode64,noquota 0 0
/dev/sda1 /boot xfs rw,seclabel,relatime,attr2,inode64,noquota 0 0
/dev/sda2 /var xfs rw,seclabel,relatime,attr2,inode64,noquota 0 0
tmpfs /run/user/0 tmpfs rw,seclabel,nosuid,nodev,relatime,size=99576k,mode=700 0 0
/dev/sdb1 /mnt ext4 rw,seclabel,relatime,data=ordered 0 0
```

همانطور که می‌بینید، دیوایس‌فایل پارتیشن `/dev/sdb1` را که همین الان مانت کردیم، به علاوه موقعیت مانت شدن آن و یک سری اطلاعات دیگر نظیر permissionهای آن در انتهای این فایل نوشته شده است. ما بعدا در مورد مفهوم تک‌تک بخش‌های نوشته شده در این فایل صحبت خواهیم کرد.

برای مشاهده‌ی دیوایس‌های مانت شده برروی سیستم، می‌توانیم خود دستور `mount` را نیز به تنهایی اجرا کنیم:

```
[root@localhost ~]# mount
...
/dev/sda2 on /var type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/sda1 on /boot type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,seclabel,size=99576k,mode=700)
/dev/sdb1 on /mnt type ext4 (rw,relatime,seclabel,data=ordered)
```

همانطور که می‌بینید، خروجی این دستور بسیار طولانی می‌باشد. دلیل این امر این است که لینوکس همیشه تعداد زیادی دستگاه مجازی را در قسمت‌های مختلف سیستم مانت می‌کند؛ پس اگر خواستید به سراغ استفاده از دستور `mount` برای مشاهده‌ی دیوایس‌های مانت شده بروید، بهتر است خروجی آن را با استفاده از `grep`، فیلتر کنید. در اینجا، چون به تازگی پارتیشن `/dev/sdb1` را مانت کرده‌ایم، این پارتیشن در انتهای لیست

نمایش داده شده توسط mount قرار دارد و نیازی به استفاده از grep و... نداریم؛ ولی اکثر اوقات مجبوریم برای پیدا کردن یک دیوایس خاص از بین این خروجی‌ها، به سراغ استفاده از grep برویم.

دستور mount آپشن‌های زیادی دارد. مثلا با آپشن r-، می‌توانیم کاری کنیم که پارتیشن به صورت Read Only مانت شود. همانطور که دیدید mount در حالت عادی خروجی به ما نشان نمی‌دهد و فقط اجرا می‌شود. با استفاده از آپشن v-، می‌توانیم کاری کنیم که این دستور، گزارشی از عملکرد خود را نیز در خروجی به ما نشان دهد. ما بررسی عملکرد این آپشن‌ها را به خودتان می‌سپاریم. اگر می‌خواهید با این آپشن‌ها و آپشن‌های دیگر این دستور بیشتر آشنا شوید، به manpage دستور mount مراجعه کنید.

حال اگر بخواهیم یک پارتیشن را از حالت مانت بودن در آوریم باید چه کنیم؟ برای این کار باید از دستور umount استفاده کنیم. بیایید با استفاده از این دستور، پارتیشن /dev/sdb1 را اصطلاحا unmount کنیم:

```
[root@localhost ~]# umount /dev/sdb1
```

حال بیایید نگاهی به فایل mtab بیاندازیم:

```
[root@localhost ~]# cat /etc/mtab
...
/dev/sda3 /home xfs rw,seclabel,relatime,attr2,inode64,noquota 0 0
/dev/sda1 /boot xfs rw,seclabel,relatime,attr2,inode64,noquota 0 0
/dev/sda2 /var xfs rw,seclabel,relatime,attr2,inode64,noquota 0 0
tmpfs /run/user/0 tmpfs rw,seclabel,nosuid,nodev,relatime,size=99576k,mode=700 0 0
```

همانطور که می‌بینید پس از آنمانت کردن این پارتیشن، دیوایس‌فایل مربوط به این پارتیشن، دیگر در فایل mstab وجود ندارد.

**نکته:** توجه داشته باشید که می‌توانیم به جای ارائه‌ی نام دیوایس‌فایل به دستور umount، موقعیت مانت‌پوینت یک پارتیشن را به این دستور بدهیم. این یعنی در صورت نداستن نام یک دیوایس‌فایل، می‌توانیم تنها با ارائه‌ی مانت‌پوینت آن پارتیشن، عملیات unmounting را انجام دهیم.

## مانت کردن اتوماتیک پارتیشن‌ها

مشکل مانت دستی با استفاده از دستور mount، این است که اگر سیستم خود را ری‌استارت کنیم، پارتیشن مورد نظر را باید دوباره مانت کنیم. این امر می‌تواند برای ما مشکل‌ساز شود؛ پس باید به سراغ روشی برویم که هنگام بوت‌شدن لینوکس، پارتیشن‌های ما را نیز در موقعیت مورد نظر، مانت کند.

لینوکس برای این که بداند چه دستگاه‌هایی باید هنگام روشن شدن در کجا مانت شوند، از فایلی به نام /etc/fstab استفاده می‌کند. فایل /etc/fstab در واقع یک جدول است که با مشخص‌کردن یک دیوایس‌فایل و همچنین موقعیت Mount Point آن، به لینوکس می‌گوید که هنگام روشن شدن، هر دیوایس را در کدام قسمت از سیستم و با چه آپشن‌هایی مانت کند. بیایید نگاهی به محتویات فایل /etc/fstab بیاندازیم:

```
[root@localhost ~]# cat /etc/fstab
#
# /etc/fstab
# Created by anaconda on Wed Jun  3 05:51:18 2020
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=b8babd8c-a32f-4216-9185-a4e7a558036a /                       xfs     defaults        0 0
UUID=b1fbcb19-d24b-4ad1-9447-7d6db41afd9d /boot                   xfs     defaults        0 0
UUID=df59f5b5-13fe-4743-b480-dae7f4ef11cc /home                   xfs     defaults        0 0
```

```
UUID=009e54b5-bc19-4811-864e-7fe913354d08 /var                   xfs     defaults        0 0
UUID=87aa0b13-48c5-45b6-9eca-8ecdbed23199 swap                   swap    defaults        0 0
```

در ستون اول این فایل، UUID هر دیوایس یا در اینجا، هر پارتیشن نوشته شده است. همانطور که قبلا گفتیم، نام دیوایس‌فایل‌ها می‌تواند با توجه به زمان وصل شدن به سیستم و... دچار تغییر شود، پس استفاده از UUID عمل هوشمندانه‌تری می‌باشد.

در ستون دوم این فایل، دایرکتوری که هرپارتیشن باید روی آن مانت شود نوشته شده است. در واقع این ستون، موقعیت مانت‌پوینت هر پارتیشن را مشخص می‌کند.

در ستون سوم، فایل‌سیستم هر پارتیشن عنوان شده است. به جای نوشتن نام فایل‌سیستم، می‌توانیم از عبارت `auto` در این ستون استفاده کنیم. این عبارت باعث می‌شود کرنل به صورت اتوماتیک سعی به تشخیص فایل‌سیستم یک پارتیشن کند.

ستون چهارم، آپشن‌های مخصوص مانت را مشخص می‌کند، یا به طور صحیح‌تر، چگونگی رفتار کرنل با فایل‌سیستم را مشخص می‌کند. در اینجا می‌توانیم چندین آپشن متفاوت قرار دهیم؛ مثلا می‌توانیم کاری کنیم که پارتیشن به صورت Read Only روی سیستم مانت شود. ما در اکثر اوقات، در این ستون مقدار `defaults` را می‌نویسیم تا دستگاه با تنظیمات پیش‌فرض مانت شود. توجه کنید که اگر پارتیشن‌هایی که از فایل‌سیستم‌های ویندوزی نظیر NTFS استفاده می‌کنند داشته باشیم، باید در این قسمت، آپشن‌هایی که به کاربران اجازه‌ی نوشتن روی پارتیشن و... می‌دهند را قرار دهیم. اطلاعات بیشتر در مورد این آپشن‌ها را می‌توانید در manpage دستور `mount` بخوانید.

ستون پنجم، بک‌آپ‌گیری یا عدم بک‌آپ‌گیری از پارتیشن توسط ابزار `dump` را مشخص می‌کند. قدیم‌ها، `dump` ابزاری بود که بسیاری از آن برای بک‌آپ‌گیری استفاده می‌کردند، اما امروزه خیلی کمتر از آن استفاده می‌شود. اگر از این ابزار برای بک‌آپ‌گیری استفاده نمی‌کنید، باید در این ستون عدد `0` را قرار دهید، اما اگر از آن استفاده می‌کنید، باید از عدد `1` در این ستون استفاده کنید.

ستون ششم، بررسی یا عدم بررسی یکپارچگی فایل‌سیستم توسط برنامه‌ی `fsck` را مشخص می‌کند. اگر بخواهیم فایل‌سیستم بررسی نشود، در این ستون عدد `0` را قرار می‌دهیم. اگر بخواهیم فایل‌سیستم بررسی شود، یک عدد بزرگتر از `0` در این بخش قرار می‌دهیم. عدد انتخابی، ترتیب بررسی را مشخص می‌کند. اگر بخواهیم برنامه‌ی `fsck` یکپارچگی فایل‌سیستم را بررسی کند، پارتیشن `root` (`/`) باید عدد `1` را در این ستون داشته باشد و سایر پارتیشن‌ها باید عدد `2` را داشته باشند.

بیایید کاری کنیم که پارتیشن `/dev/sdb1` به صورت دائم روی سیستم مانت شود. برای این کار، فایل `/etc/fstab` را با استفاده از `vi` باز کرده و اطلاعات مربوط به آن را درون این فایل اضافه می‌کنیم. اما همانطور که گفتیم، ما برای این کار نیاز به UUID این پارتیشن داریم. برای به دست آوردن UUID باید چه کنیم؟

برای به دست آوردن UUID، می‌توانیم از دستور `blkid` استفاده کنیم:

```
[root@localhost ~]# blkid
/dev/sda1: UUID="b1fbcb19-d24b-4ad1-9447-7d6db41afd9d" TYPE="xfs"
/dev/sda2: UUID="009e54b5-bc19-4811-864e-7fe913354d08" TYPE="xfs"
/dev/sda3: UUID="df59f5b5-13fe-4743-b480-dae7f4ef11cc" TYPE="xfs"
/dev/sda5: UUID="87aa0b13-48c5-45b6-9eca-8ecdbed23199" TYPE="swap"
/dev/sda6: UUID="b8babd8c-a32f-4216-9185-a4e7a558036a" TYPE="xfs"
/dev/sdb1: UUID="2bf8a60d-3c5f-4971-832d-7aef88d1702b" TYPE="ext4"
/dev/sr0: UUID="2019-09-11-19-02-53-00" LABEL="CentOS 7 x86_64" TYPE="iso9660" PTTYPE="dos"
```

همانطور که می‌بینید با استفاده از این دستور، نام هر دیوایس‌فایل و UUID مربوط به آنها و همچنین اطلاعات دیگر نظیر فایل‌سیستم آنها و... را به دست می‌آوریم.

حال که UUID پارتیشن /dev/sdb1 را داریم، می‌توانیم به سراغ اضافه کردن آن به فایل /etc/fstab برویم:

```
[root@localhost ~]# vi /etc/fstab
#
# /etc/fstab
# Created by anaconda on Wed Jun  3 05:51:18 2020
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=b8babd8c-a32f-4216-9185-a4e7a558036a /                       xfs     defaults        0 0
UUID=b1fbcb19-d24b-4ad1-9447-7d6db41afd9d /boot                   xfs     defaults        0 0
UUID=df59f5b5-13fe-4743-b480-dae7f4ef11cc /home                   xfs     defaults        0 0
UUID=009e54b5-bc19-4811-864e-7fe913354d08 /var                    xfs     defaults        0 0
UUID=87aa0b13-48c5-45b6-9eca-8ecdbed23199 swap                    swap    defaults        0 0
UUID=2bf8a60d-3c5f-4971-832d-7aef88d1702b /mnt                    ext4    defaults        0 0
```

همانطور که می‌بینید، ما با ارائه‌ی UUID پارتیشن /dev/sdb1، مانت‌پوینت آن (/mnt)، فایل‌سیستم آن (ext4) و سایر آپشن‌ها، به لینوکس گفتیم که هنگام روشن شدن، این پارتیشن را در موقعیت /mnt مانت کند.

بیایید سیستم را ریبوت کنیم و از مانت شدن این پارتیشن هنگام روشن شدن مطمئن شویم. ما با استفاده از دستور init 6 می‌توانیم سیستم را ریبوت کنیم. در جلسات بعد با این دستور بیشتر آشنا می‌شویم:

```
[root@localhost ~]# init 6
[root@localhost ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   20G  0 disk
├─sda1   8:1    0  953M  0 part /boot
├─sda2   8:2    0  4.7G  0 part /var
├─sda3   8:3    0  3.7G  0 part /home
├─sda4   8:4    0    1K  0 part
├─sda5   8:5    0  1.9G  0 part [SWAP]
└─sda6   8:6    0  8.8G  0 part /
sdb      8:16   0   20G  0 disk
└─sdb1   8:17   0    5G  0 part /mnt
sr0     11:0    1  942M  0 rom
```

همانطور که می‌بینید پس از ریبوت سیستم، پارتیشن /dev/sdb1 در دایرکتوری /mnt مانت شده است.

**نکته:** اگر به مواردی که در فایل fstab نوشته شده بیشتر دقت کنید، می‌بینید که برخی از خطوط با علامت # شروع شده‌اند. این خطوط، مانند commentها در برنامه‌نویسی هستند و توسط لینوکس خوانده نمی‌شوند. اگر بیشتر به این خطوط در فایل fstab نگاه کنید، می‌بینید که به ما پیشنهاد شده تا manpage مربوط به fstab(5) و چند دستور دیگر را مطالعه کنیم. منظور از عدد 5 در پرانتز چیست؟

manpageها، از چندین بخش متفاوت تشکیل شده‌اند. مثلا بخش ۱ برای دستورهای شل می‌باشد، بخش ۵ برای فایل‌های سیستمی می‌باشد، یا مثلا بخش ۸ برای دستورات مدیریت سیستم می‌باشد. برای دیدن همه‌ی اعداد و بخش‌های مربوط به آن، به manpage دستور man مراجعه کنید.

زمانی که نام یک دستور یا فایل را می‌بینیم و پس از آن یک عدد درون پرانتز می‌بینیم، بدین معنی است که برای دریافت اطلاعات مربوط به کارمان، باید به بخش مشخص شده در manpage آن دستور مراجعه کنیم.

مثلا در اینجا برای رفتن به بخش ۵ صفحه‌ی manpage فایل fstab، از دستور man 5 fstab استفاده می‌کنیم. اگر هیچ شماره‌ی بخشی را به دستور man ندهیم، این دستور اولین بخش موجود در manpage دستور یا فایل درخواستی را به ما نشان می‌دهد. پس نکته‌ی دیگری که می‌توانیم یاد بگیریم این است که علاوه بر دستورها، برخی از فایل‌های مربوط به تنظیمات سیستم یا سرویس‌ها نیز دارای manpage می‌باشند.

به طور کلی، لازم نیست زیاد نگران بخش‌ها و شماره‌ی آنها باشید و اکثر اوقات، فقط اجرای دستور man به علاوه‌ی دستور یا فایل مورد نظر، کار را راه می‌اندازد. مفهوم بخش‌ها در man زمانی به کار می‌آیند که ما دو manpage با نام‌های مشابه داشته باشیم، که در زمینه‌ی fstab، همچین موردی وجود ندارد. یکی دیگر از دلایل وجود این شماره بخش‌ها، به ایام قدیم باز می‌گردد که کاربران، manpageها را به صورت فیزیکی داخل یک سری کلاسور داشتند و هر کلاسور، شماره‌ای داشت که با شماره‌ی بخش‌ها در manpage همخوانی داشت. مثلا کلاسور شماره‌ی ۱، manpageهای مربوط به دستورهای شل را درون خود داشت و... .

## مدیریت پارتیشن‌ها

پس از ایجاد یک پارتیشن و مانت کردن آن، باید پارتیشن را از نظر میزان حجم باقی مانده و... مدیریت کنیم. در جلسات قبل با دستورهایی نظیر lsblk و... برای مدیریت سخت‌افزارها و پارتیشن‌ها آشنا شدیم. در این بخش با برخی دیگر از ابزارهای مخصوص این کار آشنا می‌شویم.

### مشاهده‌ی میزان مصرف از یک پارتیشن با *df*

با استفاده از دستور df، می‌توانیم اطلاعاتی نظیر نام دیوایس‌فایل پارتیشن، میزان فضای اشغال شده از پارتیشن، میزان فضای باقی‌مانده از پارتیشن، موقعیت مانت‌پوینت پارتیشن و.... را به دست آوریم. اگر فقط دستور df را وارد کنیم، خلاصه‌ای از کل فضای استفاده شده و باقی‌مانده بر روی همه‌ی پارتیشن‌ها به ما نشان داده می‌شود:

```
[root@localhost ~]# df
Filesystem     1K-blocks    Used Available Use% Mounted on
devtmpfs          487136       0    487136   0% /dev
tmpfs             497872       0    497872   0% /dev/shm
tmpfs             497872    7844    490028   2% /run
tmpfs             497872       0    497872   0% /sys/fs/cgroup
/dev/sda6        9194496 1130316   8064180  13% /
/dev/sda3        3895296   32992   3862304   1% /home
/dev/sdb1        5029504   20472   4730504   1% /mnt
/dev/sda1         972452  133244    839208  14% /boot
/dev/sda2        4872192   94268   4777924   2% /var
tmpfs              99576       0     99576   0% /run/user/0
```

همانطور که می‌بینید، در ستون Filesystem، موقعیت دیوایس‌فایل هر پارتیشن قرار گرفته است. در ستون 1-K-blocks، تعداد کل بلوک‌های موجود بر روی این پارتیشن مشخص شده است. در ستون Used، تعداد بلوک‌های مصرف شده از کل بلوک‌ها نشان داده شده است. در ستون Available، تعداد بلوک‌های خالی از تعداد کل بلوک‌ها نشان داده شده است. در ستون Use%، میزان فضای مصرف شده از این پارتیشن نشان داده شده است و در ستون Mount Point، مانت‌پوینت هر پارتیشن مشخص شده است.

**نکته:** معمولا زمانی که میزان فضای مصرف شده از یک پارتیشن به بالای ۸۰ درصد برسد، باید به سراغ پاک‌سازی پارتیشن برویم.

همانطور که می‌بینید، خروجی این دستور در حال حاضر زیاد برای ما خوانا نیست. مثلا دانستن تعداد بلوک‌های موجود برروی یک پارتیشن و تعداد بلوک‌های مصرف شده از آن، دردی از ما دوا نمی‌کند. با استفاده از آپشن h-، خروجی این دستور در واحد مبی‌بایت، گیبی‌بایت و... نشان داده می‌شود:

```
[root@localhost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        476M     0  476M   0% /dev
tmpfs           487M     0  487M   0% /dev/shm
tmpfs           487M  7.7M  479M   2% /run
tmpfs           487M     0  487M   0% /sys/fs/cgroup
/dev/sda6       8.8G  1.1G  7.7G  13% /
/dev/sda3       3.8G   33M  3.7G   1% /home
/dev/sdb1       4.8G   20M  4.6G   1% /mnt
/dev/sda1       950M  131M  820M  14% /boot
/dev/sda2       4.7G   93M  4.6G   2% /var
tmpfs            98M     0   98M   0% /run/user/0
```

همانطور که می‌بینید اکنون خروجی این دستور برای ما خواناتر شد. اگر طرفدار واحدهای پایه‌ی ۲ (مبی‌بایت، گیبی‌بایت و...) نیستید، می‌توانید از آپشن H- استفاده کنید. این آپشن، واحدها را در پایه‌ی ۱۰ (مگابایت، گیگابایت و...) به شما نشان می‌دهد:

```
[root@localhost ~]# df -H
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        499M     0  499M   0% /dev
tmpfs           510M     0  510M   0% /dev/shm
tmpfs           510M  8.1M  502M   2% /run
tmpfs           510M     0  510M   0% /sys/fs/cgroup
/dev/sda6       9.5G  1.2G  8.3G  13% /
/dev/sda3       4.0G   34M  4.0G   1% /home
/dev/sdb1       5.2G   21M  4.9G   1% /mnt
/dev/sda1       996M  137M  860M  14% /boot
/dev/sda2       5.0G   97M  4.9G   2% /var
tmpfs           102M     0  102M   0% /run/user/0
```

ما می‌توانیم از df بخواهیم که فقط فضای موجود در یک پارتیشن خاص را به ما گزارش دهد. برای این کار، می‌توانیم نام دیوایس‌فایل یا مانت‌پوینت آن پارتیشن را به df بدهیم. به علاوه می‌توانیم صرفا نام یک دایرکتوری را به df بدهیم تا به صورت اتوماتیک، فضای موجود در پارتیشنی که آن دایرکتوری در آن قرار دارد را به ما بدهد:

```
[root@localhost ~]# df -h /dev/sda1
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       950M  131M  820M  14% /boot
[root@localhost ~]# df -h /home/
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3       3.8G   33M  3.7G   1% /home
[root@localhost ~]# df -h /etc/networks
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda6       8.8G  1.1G  7.7G  13% /
```

همانطور که می‌بینید، با ارائه‌ی نام دیوایس‌فایل و یا مانت‌پوینت یک پارتیشن، می‌توانیم اطلاعات مربوط به آن پارتیشن را به دست آوریم. در دستور سوم، ما با ارائه‌ی نام یک دایرکتوری نیز، اطلاعات مربوط به پارتیشنی که این دایرکتوری برروی آن قرار دارد را به دست آوردیم. این دستور، زمانی که ندانیم یک دایرکتوری بر روی کدام پارتیشن قرار دارد به کار می‌آید.

## مشاهده‌ی فضای اشغال شده توسط هر فایل و دایرکتوری با *du*

همانطور که دیدیم، یکی از کاربردهای ابزار df، پیدا کردن پارتیشن‌هایی که در حال پر شدن هستند می‌باشد. اما وقتی فهمیدیم که کدام پارتیشن در حال پر شدن می‌باشد، لازم است ببینیم که دقیقا کدام فایل‌ها یا دایرکتوری‌ها بیشترین فضا را پر کرده‌اند. این کار را می‌توانیم با ابزار du انجام دهیم. استفاده از du بسیار شبیه به ابزار df می‌باشد. برای مثال:

```
[root@localhost ~]# du /etc/firewalld/
0     /etc/firewalld/helpers
0     /etc/firewalld/icmptypes
0     /etc/firewalld/ipsets
0     /etc/firewalld/services
8     /etc/firewalld/zones
16    /etc/firewalld/
```

همانطور که می‌بینید، این دستور به سراغ تک‌تک دایرکتوری‌های موجود در دایرکتوری /etc/firewalld می‌رود و تعداد بلوک‌های مصرفی هر کدام را در اولین ستون خروجی نشان می‌دهد. در آخرین خط، تعداد بلوک اشغالی توسط دایرکتوری اصلی درخواستی به ما نشان داده می‌شود.

این دستور به صورت پیش‌فرض فقط به سراغ دایرکتوری‌ها می‌رود و فضای اشغالی توسط آنها را گزارش می‌دهد. برای این که du به سراغ فایل‌ها نیز برود و حجم آنها را نیز گزارش دهد، از آپشن a- استفاده می‌کنیم:

```
[root@localhost ~]# du -a /etc/firewalld/
4     /etc/firewalld/firewalld.conf
0     /etc/firewalld/helpers
0     /etc/firewalld/icmptypes
0     /etc/firewalld/ipsets
4     /etc/firewalld/lockdown-whitelist.xml
0     /etc/firewalld/services
4     /etc/firewalld/zones/public.xml
4     /etc/firewalld/zones/public.xml.old
8     /etc/firewalld/zones
16    /etc/firewalld/
```

همانطور که می‌بینید، با استفاده از آپشن a-، این دستور به سراغ فایل‌های موجود در دایرکتوری /etc/firewalld نیز رفت. نکته‌ی قابل توجه این است که df -a (و به طور کلی، خود دستور df)، به صورت recursive عمل می‌کند؛ یعنی به سراغ فایل‌های موجود درون دایرکتوری‌های داخل /etc/firewalld نیز می‌رود.

بسیاری از آپشن‌های دستور du، شبیه به دستور df می‌باشند. برای مثال با استفاده از آپشن h-، میزان فضای اشغالی توسط هر فایل یا دایرکتوری در واحدهای پایه‌ی ۲ (کیبی‌بایت، مبی‌بایت و...) به ما نشان داده می‌شود و در صورت استفاده از آپشن H-، میزان فضای اشغالی در واحدهای پایه ۱۰ به ما نشان داده می‌شود (کیلوبایت، مگابایت و...).

برای مثال:

```
[root@localhost ~]# du -h /etc/sysconfig/
8.0K  /etc/sysconfig/cbq
0     /etc/sysconfig/console
0     /etc/sysconfig/modules
236K  /etc/sysconfig/network-scripts
332K  /etc/sysconfig/
```

اگر بخواهیم du فقط حجم اشغالی توسط دایرکتوری مشخص شده را به ما گزارش دهد، کافی است از آپشن s- استفاده کنیم:

```
[root@localhost ~]# du -hs /etc/sysconfig/
332K  /etc/sysconfig/
```

اگر به du هیچ دایرکتوری را ندهیم، فضای اشغالی توسط دایرکتوری کنونی را به ما نشان می‌دهد.

### چک کردن فایل‌سیستم‌ها با استفاده از *fsck*

چک کردن و بررسی یک پارتیشن، کاری است که مجبوریم در بسیاری از اوقات انجام دهیم. باگ‌ها، مشکلات برق و حتی مشکلات مکانیکی می‌توانند سلامت ساختمان داده‌های موجود بر روی هارد دیسک را تحت تاثیر قرار دهند. خوشبختانه لینوکس ابزارهایی جهت بررسی یکپارچگی و تعمیر فایل‌سیستم‌ها در اختیار ما قرار می‌دهد. یکی از این ابزارها، ابزار fsck می‌باشد. استفاده از این ابزار بسیار ساده می‌باشد. کافی است آدرس دیوایس‌فایل پارتیشن مورد نظر را به این دستور بدهیم تا عملیات خود را انجام دهد. توجه کنید که این برنامه روی پارتیشن‌هایی که مانت نیستند قابل اجرا می‌باشد:

```
[root@localhost ~]# fsck /dev/sdb1
fsck from util-linux 2.23.2
e2fsck 1.42.9 (28-Dec-2013)
/dev/sdb1: clean, 11/327680 files, 58462/1310720 blocks
```

اگر بخواهیم پارتیشن‌های اصلی سیستم نظیر / و... را چک کنیم، باید این کار را هنگام بوت انجام دهیم. برای این کار، کافی است در فایل fstab که قبلا در مورد آن صحبت کردیم، چک کردن پارتیشن هنگام روشن شدن سیستم را فعال کنیم.

## مدیریت فایل‌ها در لینوکس

همانطور که در جلسه‌ی قبل گفتیم، کلیه‌ی فایل‌ها در یک سیستم لینوکسی، تحت یک ساختار واحد که به آن virtual directory یا دایرکتوری مجازی می‌گویند، ذخیره می‌شوند. این ساختار، یک دایرکتوری پایه به نام root (/) دارد که دسترسی به سایر دایرکتوری‌ها از طریق آن امکان‌پذیر می‌باشد. این بدین معناست که برای دسترسی به دایرکتوری‌ها و فایل‌های موجود بر روی یک پارتیشن یا حتی یک دستگاه ذخیره‌سازی دیگر، ما حرکت خود را از دایرکتوری روت شروع می‌کنیم. این دقیقا بر خلاف عملکرد سیستم‌عاملی مانند ویندوز می‌باشد که در آن برای دسترسی به هر پارتیشن یا دستگاه ذخیره‌سازی جانبی، حرکت خود را از drive letter اختصاص یافته به آن پارتیشن یا دستگاه شروع می‌کنیم.

تا به اینجا به درک مناسبی در مورد ساختار دایرکتوری‌ها در لینوکس رسیده‌ایم و با رفتار لینوکس با پارتیشن‌ها و دستگاه‌های ذخیره‌سازی آشنا شده‌ایم. در این بخش می‌خواهیم در مورد ابزارهایی که از آن برای مشاهده، ایجاد، کپی و جابه‌جایی فایل‌ها در ساختار دایرکتوری مجازی لینوکس استفاده می‌کنیم، صحبت کنیم. تا به اینجا با

برخی از این دستورها به صورت ابتدایی کار کرده‌ایم، اما اکنون سعی می‌کنیم برخی از آنها را یادآوری کرده و برخی دیگر را با عمق بیشتری یاد بگیریم.

## مشاهده‌ی موقعیت کنونی در دایرکتوری مجازی با *pwd*

هنگام استفاده از کامندلاین لینوکس، ما همیشه داخل یک دایرکتوری هستیم و برای انجام کارهای متفاوت، معمولا باید وارد سایر دایرکتوری‌ها شویم. خیلی از اوقات، حین حرکت بین دایرکتوری‌های متفاوت، ممکن است گم شویم و ندانیم که دقیقا در کجای سیستم هستیم. دستور pwd، می‌تواند موقعیت کنونی ما درون سیستم، یا به عبارت دیگر، absolute path موقعیت کنونی را به ما نشان دهد. استفاده از این دستور بسیار ساده می‌باشد:

```
[root@localhost ~]# pwd
/root
```

همانطور که می‌بینید، در حال حاضر موقعیت کنونی ما دایرکتوری /root می‌باشد. این دایرکتوری، home directory (نشان داده شده با علامت ~) کاربر root می‌باشد.

**نکته:** در جزوه‌ی جلسه‌ی اول، در مورد مفهوم absolute path و relative path صحبت کردیم. اگر مفهوم آنها را فراموش کرده‌اید، به صفحه‌ی ۲٤ جزوه‌ی جلسه‌ی اول مراجعه کنید.

## مشاهده‌ی محتویات یک دایرکتوری با استفاده از دستور *ls*

ساده‌ترین دستور برای مشاهده‌ی محتویات یک دایرکتوری و همچنین به دست‌آوردن برخی اطلاعات جانبی در مورد فایل‌ها، دستور ls می‌باشد. ما تا به اینجا چندین بار با دستور ls کار کرده‌ایم و تقریبا با آن آشنایی داریم. دستور ls، در ساده‌ترین حالت خود، محتویات دایرکتوری کنونی را به ما نشان می‌دهد:

```
[root@localhost ~]# ls
albatross        nohup.out   squid-4.10.tar.gz
anaconda-ks.cfg  squid-4.10  squid-4.10.tar.gz.asc
[root@localhost ~]# pwd
/root
```

همانطور که می‌بینید، این دستور محتویات موجود در دایرکتوری /root را به ما نشان داد.

دستور ls، آپشن‌های بسیار زیادی دارد که در اینجا با برخی از آنها آشنا می‌شویم. اولین آپشنی که در مورد آن صحبت می‌کنیم، آپشن l- می‌باشد. با استفاده از این آپشن، می‌توانیم خروجی ls را در قالب یک لیست مشاهده کنیم:

```
[root@localhost ~]# ls -l
total 5204
-rw-r--r--.  1 root root      47 Apr 19 11:58 albatross
-rw-------.  1 root root    1257 Mar 20 10:48 anaconda-ks.cfg
-rw-------.  1 root root   53083 May  6 11:01 nohup.out
drwxr-xr-x. 16 1000 1000    4096 Apr 16 11:24 squid-4.10
-rw-r--r--.  1 root root 5256312 Jan 20 08:25 squid-4.10.tar.gz
-rw-r--r--.  1 root root    1194 Jan 20 08:25 squid-4.10.tar.gz.asc
```

همانطور که می‌بینید، l- فایل‌ها و دایرکتوری‌های موجود در این موقعیت را به صورت یک لیست به ما نشان داد. البته اطلاعات دیگری نیز در خروجی به ما داده می‌شود. برای مثال در خط اول، تعداد کل بلوک‌های اشغال شده توسط این دایرکتوری (۵۲۰٤) به ما نشان داده می‌شود.

در ستون اول، permissionهای فایل یا دایرکتوری به ما نشان داده می‌شود (-rw-r--r--.). در جلسات بعد به صورت کامل در مورد مفهوم محتویات این ستون صحبت می‌کنیم. در ستون دوم، تعداد لینک‌هایی که به این فایل زده شده است نشان داده می‌شود. در جلسات آینده با لینک‌های symbolic و hard آشنا می‌شویم. در ستون سوم، Username صاحب فایل یا فولدر نوشته شده است. در ستون چهارم، نام گروهی که عضوهای آن به فایل دسترسی خواهند داشت نشان داده می‌شود (در ستون سوم و چهارم، ممکن است به جای نام User و Group، User ID و Group ID نشان داده شود). در ستون پنجم، تعداد بلوک‌های اشغالی توسط این فایل یا فولدر به ما نمایش داده می‌شود و ستون‌های بعدی، تاریخ ایجاد یا تغییر و همچنین نام فایل یا دایرکتوری را نمایش می‌دهند.

همانطور که می‌بینید، در حال حاضر، فضای اشغالی (حجم) توسط هر دایرکتوری یا فایل، زیاد برای ما قابل درک نیست. برای رفع این مشکل، دستور ls را با آپشن h- اجرا می‌کنیم:

```
[root@localhost ~]# ls -lh
total 5.1M
-rw-r--r--.  1 root root   47 Apr 19 11:58 albatross
-rw-------.  1 root root 1.3K Mar 20 10:48 anaconda-ks.cfg
-rw-------.  1 root root  52K May  6 11:01 nohup.out
drwxr-xr-x. 16 1000 1000 4.0K Apr 16 11:24 squid-4.10
-rw-r--r--.  1 root root 5.1M Jan 20 08:25 squid-4.10.tar.gz
-rw-r--r--.  1 root root 1.2K Jan 20 08:25 squid-4.10.tar.gz.asc
```

همانطور که می‌بینید، با استفاده از این آپشن، فضای اشغالی توسط هر فایل یا دایرکتوری در واحد مبی‌بایت و کیبی‌بایت به ما نمایش داده می‌شود (سایزها در پایه‌ی ۲). اگر بخواهیم فضای اشغالی در واحد کیلوبایت و... به ما نشان داده شوند (سایزهای پایه ۱۰)، از آپشن si-- استفاده می‌کنیم:

```
[root@localhost ~]# ls -l --si
total 5.4M
-rw-r--r--.  1 root root   47 Apr 19 11:58 albatross
-rw-------.  1 root root 1.3k Mar 20 10:48 anaconda-ks.cfg
-rw-------.  1 root root  54k May  6 11:01 nohup.out
drwxr-xr-x. 16 1000 1000 4.1k Apr 16 11:24 squid-4.10
-rw-r--r--.  1 root root 5.3M Jan 20 08:25 squid-4.10.tar.gz
-rw-r--r--.  1 root root 1.2k Jan 20 08:25 squid-4.10.tar.gz.asc
```

توجه کنید که برای این که h- و si-- به ما خروجی دهند، باید حتما l- نیز به دستور ls اعمال شده باشد.

روشی دیگر برای مشاهده‌ی فایل‌های موجود در یک دایرکتوری به صورت لیست، استفاده از آپشن 1- (عدد ۱) می‌باشد. این آپشن بر خلاف l-، اطلاعات جانبی به ما نشان نمی‌دهد و فقط در هر خط، نام یک فایل یا یک دایرکتوری را نشان می‌دهد:

```
[root@localhost ~]# ls -1
albatross
anaconda-ks.cfg
nohup.out
squid-4.10
squid-4.10.tar.gz
squid-4.10.tar.gz.asc
```

با استفاده از آپشن a-، می‌توانیم کلیه‌ی فایل‌ها و دایرکتوری‌های موجود در موقعیت کنونی، اعم از فایل‌های پنهان (hidden) را مشاهده کنیم:

در لینوکس فایل‌هایی که در ابتدای اسمشان یک نقطه (.) داشته باشند، hidden در نظر گرفته می‌شوند.

```
[root@localhost ~]# ls -a
.                .bash_logout   .emacs.d       .pki                    .w3m
..               .bash_profile  .gnupg         squid-4.10
albatross        .bashrc        .im_hidden_yo  squid-4.10.tar.gz
anaconda-ks.cfg  .cache         .lesshst       squid-4.10.tar.gz.asc
.bash_history    .cshrc         nohup.out      .tcshrc
```

یا برای مشاهده‌ی بهتر خروجی:

```
[root@localhost ~]# ls -lha
total 5.2M
dr-xr-x---.  8 root root 4.0K Jun 23 13:05 .
dr-xr-xr-x. 17 root root  224 Mar 20 10:47 ..
-rw-r--r--.  1 root root   47 Apr 19 11:58 albatross
-rw-------.  1 root root 1.3K Mar 20 10:48 anaconda-ks.cfg
-rw-------.  1 root root  16K Jun  5 02:42 .bash_history
-rw-r--r--.  1 root root   18 Dec 29  2013 .bash_logout
-rw-r--r--.  1 root root  176 Dec 29  2013 .bash_profile
-rw-r--r--.  1 root root  176 Dec 29  2013 .bashrc
drwx------.  3 root root   17 May  2 10:58 .cache
-rw-r--r--.  1 root root  100 Dec 29  2013 .cshrc
drwx------.  2 root root    6 Mar 24 17:44 .emacs.d
drwx------.  2 root root   60 Apr 18 13:16 .gnupg
-rw-r--r--.  1 root root    0 Jun 23 10:55 .im_hidden_yo
-rw-------.  1 root root  375 Jun 23 13:07 .lesshst
-rw-------.  1 root root  52K May  6 11:01 nohup.out
drwxr-----.  3 root root   19 Apr 13 14:38 .pki
drwxr-xr-x. 16 1000 1000 4.0K Apr 16 11:24 squid-4.10
-rw-r--r--.  1 root root 5.1M Jan 20 08:25 squid-4.10.tar.gz
-rw-r--r--.  1 root root 1.2K Jan 20 08:25 squid-4.10.tar.gz.asc
-rw-r--r--.  1 root root  129 Dec 29  2013 .tcshrc
drwx------.  2 root root   35 Apr 15 10:31 .w3m
```

با استفاده از آپشن t-، می‌توانیم خروجی را برحسب تاریخ مرتب کنیم (در حالت معمولی، خروجی بر حسب نام فایل مرتب می‌شود):

```
[root@localhost ~]# ls -tl
total 5204
-rw-r--r--.  1 root root      52 Jun 24 11:22 albatross
-rw-------.  1 root root   53083 May  6 11:01 nohup.out
drwxr-xr-x. 16 1000 1000    4096 Apr 16 11:24 squid-4.10
-rw-------.  1 root root    1257 Mar 20 10:48 anaconda-ks.cfg
-rw-r--r--.  1 root root 5256312 Jan 20 08:25 squid-4.10.tar.gz
-rw-r--r--.  1 root root    1194 Jan 20 08:25 squid-4.10.tar.gz.asc
```

با استفاده از آپشن R-، می‌توانیم از ls بخواهیم که به صورت recursive عمل کند؛ این یعنی ls به محتویات دایرکتوری کنونی نگاه می‌کند و در صورت وجود دایرکتوری دیگری داخل دایرکتوری کنونی، محتویات آن دایرکتوری را نیز در خروجی به ما می‌دهد و در صورت وجود دایرکتوری در آن دایرکتوری، محتویات آن را نیز به ما نشان می‌دهد و بدین شکل، کلیه‌ی محتویات کلیه‌ی دایرکتوری‌های تو در تو را به ما گزارش می‌دهد. مثلا:

```
[root@localhost ~]# ls -lR
.:
total 5200
drwxr-xr-x. 3 root root      44 Jun 24 11:33 aDirectory
-rw-r--r--. 1 root root      52 Jun 24 11:22 albatross
-rw-------. 1 root root    1257 Mar 20 10:48 anaconda-ks.cfg
```

```
-rw-------. 1 root root   53083 May  6 11:01 nohup.out
-rw-r--r--. 1 root root 5256312 Jan 20 08:25 squid-4.10.tar.gz
-rw-r--r--. 1 root root    1194 Jan 20 08:25 squid-4.10.tar.gz.asc

./aDirectory:
total 0
drwxr-xr-x. 2 root root 20 Jun 24 11:34 andAnotherDirectory
-rw-r--r--. 1 root root  0 Jun 24 11:33 thatHasAFile

./aDirectory/andAnotherDirectory:
total 0
-rw-r--r--. 1 root root 0 Jun 24 11:34 killMe
```

همانطور که می‌بینید با اعمال آپشن R-، این دستور ابتدا محتویات دایرکتوری کنونی را به ما نشان داد. از آنجایی که در دایرکتوری کنونی یک دایرکتوری دیگر به نام aDirectory داشتیم، محتویات آن را پس از قرار دادن یک خط خالی، برای ما لیست کرد. از آنجا که در دایرکتوری aDirectory یک دایرکتوری دیگر به نام andAnotherDirectory داشتیم، محتویات آن را نیز پس از قرار دادن یک خط خالی در خروجی به ما نشان داد. سپس از آنجا که هیچ دایرکتوری دیگر باقی نمانده بود، کار این دستور به پایان رسید.

توجه کنید که ما می‌توانیم به دستور ls، آدرس یک دایرکتوری را بدهیم و از آن بخواهیم که محتویات آن دایرکتوری خاص را به ما نشان دهد. برای مثال:

```
[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 27176 Oct 30  2018 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
```

دستور ls آپشن‌های دیگری نیز دارد. پیشنهاد می‌شود manpage این دستور را مطالعه کنید و آپشن‌هایی که فکر می‌کنید ممکن است به دردتان بخورد را یاد بگیرید.

**نکته:** اگر وارد کردن دستور ls با آپشن l- برایتان دشوار است، می‌توانید، از دستور ll استفاده کنید. دستور ll در اکثر توزیع‌ها موجود است و در واقع یک alias برای دستور ls -l می‌باشد و به محض اجرا، دقیقا خروجی ls -l را به شما می‌دهد. در آینده با aliasها و عملکرد آنها بیشتر آشنا می‌شویم.

## ایجاد فایل‌ها با استفاده از *touch*

با استفاده از دستور touch می‌توانیم فایل‌های خالی روی سیستم ایجاد کنیم. در جلسه‌ی دوم با این قابلیت دستور touch آشنا شدیم، اما جالب است بدانید که هدف اصلی از این دستور، ایجاد تغییرات در timestamp یک فایل می‌باشد. فعلا بیایید چگونگی ایجاد فایل‌های خالی با دستور touch را به یاد آوریم:

```
[root@localhost ~]# touch file1.txt
[root@localhost ~]# ls -l
total 0
-rw-r--r--. 1 root root 0 Jun 25 10:16 file1.txt
```

همانطور که می‌بینید با استفاده از دستور touch و ارائه‌ی یک نام، موفق به ایجاد یک فایل خالی شدیم. اگر به دستور touch، بیش از یک نام فایل دهیم، این دستور به تعداد نام‌های ارائه شده، اقدام به ایجاد فایل خالی می‌کند:

```
[root@localhost ~]# touch file2.txt file3.txt
[root@localhost ~]# ls -l
total 0
-rw-r--r--. 1 root root 0 Jun 25 10:16 file1.txt
-rw-r--r--. 1 root root 0 Jun 25 10:16 file2.txt
-rw-r--r--. 1 root root 0 Jun 25 10:16 file3.txt
```

همانطور که می‌بینید ما با ارائه‌ی نام‌های file2.txt و file3.txt در یک خط، دو فایل با این نام‌ها ایجاد کردیم.

## نام‌گذاری فایل‌ها و دایرکتوری‌ها در لینوکس

حال که داریم در مورد فایل‌ها صحبت می‌کنیم، بهتر است در مورد کاراکترهای مجاز در ایجاد نام یک فایل (و دایرکتوری) نیز صحبت کنیم. برای لینوکس بزرگ یا کوچک بودن حروف نام فایل‌ها و دایرکتوری‌ها مهم می‌باشد. یعنی از نظر لینوکس، فایل killme.txt، KillMe.txt و فایل KILLME.TXT، فایل‌هایی کاملا مجزا می‌باشند. به طور کلی، لینوکس محدودیت‌های خاصی در نام‌گذاری فایل‌ها و دایرکتوری‌ها در سر راه ما قرار نمی‌دهد، اما بهتر است برای جلوگیری از به وجود آمدن مشکلات و سردرگمی، از کاراکترهای زیر در نام‌گذاری فایل‌ها و دایرکتوری‌ها استفاده نکنیم:

```
* ? [ ] ‘ “ \ / $ ; & ( ) | ^ < >
```

لازم است بار دیگر تاکید کنیم که شما می‌توانید در نام دایرکتوری یا فایل خود اکثر این کاراکترها را داشته باشید (به جز / که در آدرس‌دهی از آن استفاده می‌شود)، اما پیشنهاد می‌شود این کار را انجام ندهید، چرا که این کاراکترها، کاراکترهای ویژه هستند در کارهای نظیر regex و globbing از آنها استفاده می‌شود.

طول نام‌هایی که می‌توانیم به یک فایل یا دایرکتوری اختصاص دهیم، بستگی به فایل‌سیستم مورد استفاده دارد. بر روی فایل‌سیستم‌های ext2، ext3، ext4، XFS، btrfs و خیلی دیگر از فایل‌سیستم‌ها، نام فایل‌ها محدودیت ۲۵۵ کاراکتری دارند.

در قرار دادن پسوند برای فایل‌ها نیز، باید مراقب کاراکترهایی که در بالا نام بردیم باشید. پسوند در لینوکس، مانند بسیاری دیگر از سیستم‌عامل‌ها، پس از یک علامت . قرار می‌گیرد. البته ما می‌توانیم در هر کجای نام فایل (حتی در ابتدای نام فایل)، علامت . را داشته باشیم. آیا به خاطر دارید که قرار دادن علامت . در ابتدای نام یک فایل، سبب چه چیزی می‌شد؟ 😉

## مشاهده‌ی نوع فایل با استفاده از *file*

همانطور که قبلا گفتیم، در لینوکس، همه چیز یک فایل است. اما یک فایل متنی، با یک دیوایس‌فایل خیلی فرق دارد. اگر بخواهیم نوع یک فایل را بدانیم، باید از دستور file استفاده کنیم. برای مثال:

```
[root@localhost ~]# file file1.txt
file1.txt: empty
```

همانطور که می‌بینید، فایل file1.txt که با دستور touch ایجاد کردیم، یک فایل خالی می‌باشد و نوع خاصی ندارد.

بیایید یک فایل دیگر را بررسی کنیم:

```
[root@localhost ~]# file /etc/postfix/main.cf
/etc/postfix/main.cf: ASCII text
```

همانطور که می‌بینید، فایل تنظیمات نرم‌افزار postfix، یک فایل متنی ASCII می‌باشد. بیایید این کار را روی یک دیوایس‌فایل نیز امتحان کنیم:

```
[root@localhost ~]# file /dev/sda2
/dev/sda2: block special
```

همانطور که می‌بینید، این دستور به ما می‌گوید که /dev/sda2، یک دیوایس‌فایل بلوکی می‌باشد.

## استفاده از Wildcard Expansionها

هنگام استفاده از دستورهایی نظیر ls و...، ما نمی‌توانیم نام‌فایل‌ها یا مسیرهای داده شده به دستور را توسط regexها فیلتر کنیم. دلیل این امر، عدم توانایی bash در تفسیر (یا درک) regexها می‌باشد. در واقع عدم توانایی bash در درک regexها، دلیل استفاده‌ی ما از ابزارهایی نظیر grep و sed بود. شاید از خود بپرسید که چرا صرفا خروجی دستوری مانند ls را درون نرم‌افزاری مانند grep پایپ نمی‌کنیم؟ با این که همچین کاری هنگام استفاده از دستور ls ممکن است، نکته‌ی کلیدی که باید به آن توجه کنیم، کلمه‌ی «خروجی» می‌باشد؛ بله، ما می‌توانیم خروجی یک دستور را توسط grep فیلتر کنیم و نام‌فایل‌های مورد نظر را به دست آوریم، اما اگر بخواهیم فایل‌هایی که یک سری نام خاص دارند را به ورودی یک نرم‌افزار مانند rm بدهیم چه؟ در جزوه‌ی جلسه دوم دیدیم که دستور rm، نمی‌تواند خروجی سایر دستورها را به عنوان ورودی خود قبول کند. در چنین جایی grep به کار ما نمی‌آید، و یا صرفا کار ما را سخت‌تر می‌کند.

پس تا اینجا فهمیدیم که bash به تنهایی نمی‌تونند regexها را درک کند. اما bash، برخی از کاراکترهای ویژه که به آنها Wildcard می‌گویند را درک می‌کند و زمانی که به دستوری مانند ls، یک Wildcard می‌دهیم، آن Wildcard توسط bash ترجمه شده و سپس به دستور ls داده می‌شود. به این عمل، Filename Expansion، Wildcard Expansion، File Globbing یا Globbing می‌گویند 😢.

پس به طور خلاصه، ما با استفاده از برخی کاراکترهای ویژه یا وایلدکاردها، می‌توانیم نام فایل‌ها را فیلتر کنیم. بیایید با برخی از این وایلدکارها آشنا شویم:

- کاراکتر ? هنگامی که به عنوان یک وایلدکارد به کار رود، نمایانگر یک کاراکتر می‌باشد. برای مثال، دستوری مانند ls c?t، کلیه‌ی فایل‌های ۳ حرفی که نامشان با c شروع شود و با t تمام شود را به ما نشان می‌دهد. مثلا در اینجا کلماتی نظیر cut و cat در خروجی به ما نشان داده می‌شوند، اما کلمه‌ای نظیر count به ما نشان داده نمی‌شود، چون بیش از ۳ حرف دارد.

- کاراکتر * هنگامی که به عنوان یک وایلدکارد به کار رود، نمایانگر صفر یا هر تعداد کاراکتر می‌باشد. برای مثال دستوری نظیر ls b*k، هر فایلی که نام آن با b شروع شود و با k تمام شود را به ما نشان می‌دهد. یعنی در اینجا، ls b*k، فایل‌هایی که نام‌هایی نظیر book، black، buck، barack، bk و... داشته باشند را به ما نشان می‌دهد. توجه کنید که اگر از این وایلدکارد به صورت تنها استفاده کنید، bash فایل‌ها یا دایرکتوری‌های Hidden (که با علامت . شروع می‌شوند) را به شما باز نخواهد گرداند. اگر می‌خواهید bash فایل‌ها و دایرکتوری‌های hidden را نیز به شما بازگرداند، باید از .* استفاده کنید.

- اگر برخی از کاراکترها را درون براکت ([]) قرار دهیم، به سیستم می‌گوییم که فقط یکی از کاراکترهای موجود درون براکت می‌توانند در این مکان در نام فایل، قرار داشته باشند. برای مثال دستوری مانند `ls b[ae]t`، هر فایلی که نامی ۳ حرفی داشته باشد، به طوری که اولین حرف آن b، دومین حرف آن کاراکتر a **یا** کاراکتر e، و سومین حرف آن t باشد را در خروجی به ما نشان می‌دهد. یعنی این دستور فایل‌هایی با نام bat و bet را درخروجی به ما نشان می‌دهد، اما فایل‌هایی با اسم baet، beat یا bit، به ما بازگردانده نخواهد شد. ما می‌توانیم در جستجوی خود از ۲ یا چند براکت نیز استفاده کنیم.

ما می‌توانیم درون براکت، range یا محدوده‌ای کاراکترها را قرار دهیم. برای مثال `ls b[a-z]t`، به سراغ فایل‌هایی که نامی ۳ حرفی دارند، به طوری که حرف اول آنها b، حرف دوم آنها هر یک از حروف کوچک الفبا (یعنی a کوچک تا z کوچک) و حرف سوم آنها t می‌باشد می‌رود و آنها را به ما باز می‌گرداند. یعنی `ls b[a-z]t`، نام‌هایی نظیر bat، bit، bot، bet، but و... را در خروجی به ما نشان می‌دهد.

> شاید از خود بپرسید که تفاوت b[a-z]t با b?t در چیست. در b?t، حرف دوم، می‌تواند هر کاراکتری، اعم از حروف بزرگ، کوچک، اعداد و برخی دیگر از کاراکتر باشد. اما در b[a-z]t، حرف دوم، فقط می‌تواند حروف کوچک الفبا باشد. در جزوه‌ی جلسه‌ی دوم، در مورد محدوده‌های کاراکتری صحبت کردیم. در صورت تمایل برای کسب اطلاعات بیشتر، به آن جزوه مراجعه کنید.

- با قرار دادن یک علامت ^ درون براکت، می‌توانیم به bash بگوییم که چه حروفی را در جستجوی نام فایل‌ها، اعمال نکند. برای مثال دستور `ls b[^eio]t`، کلیه‌ی فایل‌هایی که نامی ۳ حرفی داشته باشند، به طوری که حرف اول آنها b، حرف دوم آنها هر کاراکتری به جز کاراکترهای e، i و o و در نهایت حرف سوم آنها t باشد را به ما باز می‌گرداند. یعنی `ls b[^eio]t`، فایل‌هایی با نام bat و but یا حتی bEt را به ما باز می‌گرداند، اما فایل‌های دارای نام bet، bit و bot را به ما باز نمی‌گرداند.

توجه کنید که کلیه‌ی مفاهیمی که در مورد globbing گفتیم، در مورد نام دایرکتوری‌ها نیز صادق است؛ یعنی globbing هم برای فیلتر نام فایل‌ها و هم برای فیلتر نام دایرکتوری‌ها به کار می‌رود.

## ایجاد دایرکتوری‌ها با `mkdir`

با استفاده از دستور `mkdir`، می‌توانیم به راحتی دایرکتوری‌های مورد نظر را ایجاد کنیم. استفاده از این دستور بسیار ساده می‌باشد. برای مثال:

```
[root@localhost ~]# mkdir TestDirectory
```

همانطور که می‌بینید، برای ایجاد دایرکتوری، کافی است نام دلخواه خود را به دستور `mkdir` بدهیم. حال بیایید مطمئن شویم که دایرکتوری‌مان ایجاد شده است:

```
[root@localhost ~]# ls -F
bad  born  sign  TestDirectory/  under
```

همانطور که می‌بینید دایرکتوری ما با نام `TestDirectory` ایجاد شده است.

> با اعمال آپشن `F-` به دستور `ls`، از این دستور می‌خواهیم که نوع فایل‌ها را نیز به ما نشان دهد. (/ نشان دهنده‌ی دایرکتوری می‌باشد.)

ما می‌توانیم با استفاده از دستور mkdir، در موقعیت‌های دیگر نیز یک دایرکتوری ایجاد کنیم. برای مثال، بیایید یک دایرکتوری در مسیر /tmp ایجاد کنیم:

```
[root@localhost ~]# mkdir /tmp/AnotherDirectory
[root@localhost ~]# ls -F /tmp/
AnotherDirectory/
```

همانطور که می‌بینید، برای ایجاد یک دایرکتوری در هر موقعیتی، کافی است آدرس موقعیت مورد نظر و سپس نام دایرکتوری که میخواهیم ایجاد کنیم را به دستور mkdir بدهیم. در اینجا /tmp موقعیت مورد نظر و AnotherDirectory، نام دایرکتوری که می‌خواهیم ایجاد کنیم می‌باشد.

اگر بخواهیم چند دایرکتوری تو در تو ایجاد کنیم، باید از آپشن p- استفاده کنیم؛ در غیر این صورت دستور mkdir اجازه‌ی ایجاد دایرکتوری‌های تو در تو را به ما نمی‌دهد. بیایید این قضیه را با یک مثال ببینیم:

```
[root@localhost ~]# ls -F
bad  born  sign  TestDirectory/  under
[root@localhost ~]# mkdir This/Is/Weird
mkdir: cannot create directory ‘This/Is/Weird’: No such file or directory
```

همانطور که می‌بینید، هنگام درخواست ایجاد چند دایرکتوری تو در تو، mkdir به ما یک پیغام خطا می‌دهد. پس باید به سراغ استفاده از آپشن p- برویم:

```
[root@localhost ~]# mkdir -p This/Is/Weird
[root@localhost ~]# ls -RF
.:
bad  born  sign  TestDirectory/  This/  under

./TestDirectory:

./This:
Is/

./This/Is:
Weird/

./This/Is/Weird:
```

همانطور که می‌بینید، استفاده از آپشن p- سبب شد که دستور mkdir بتواند دایرکتوری‌های تو در تو This/Is/Weird را در موقعیت کنونی ایجاد کند.

> اگر به دستورهای وارد شده جهت ایجاد دایرکتوری تو در تو دقت کنید، می‌بینید که ما در ابتدای نام دایرکتوری‌های مورد نظر، علامت / را قرار ندادیم (یعنی نوشتیم This/Is/Weird، یا به عبارت دیگر، در ابتدای This، یک علامت / قرار ندادیم). دلیل این کار، این است که ما می‌خواستیم این دایرکتوری‌های تو در تو، در موقعیت کنونی ایجاد شوند. اگر دایرکتوری‌های مورد نظر را به صورت /This/Is/Weird به mkdir داده بودیم، این دستور دایرکتوری This را به جای موقعیت کنونی، در دایرکتوری روت (/) ایجاد می‌کرد.

## کپی کردن فایل‌ها و دایرکتوری‌ها با استفاده از *cp*

برای کپی یک فایل یا یک دایرکتوری، از دستور cp استفاده می‌کنیم. به طور کلی، این دستور از syntax زیر پیروی می‌کند:

```
cp [options] SOURCE DEST
```

پس ما در صورت نیاز می‌توانیم به این دستور یک سری آپشن بدهیم و سپس باید حتما آدرس فایلی که می‌خواهیم کپی کنیم را وارد کرده و سپس باید حتما آدرس مکانی که می‌خواهیم فایل درون آن کپی شود را وارد کنیم:

```
[root@localhost ~]# ls
ADirectory  opeth.txt  rush.txt
[root@localhost ~]# cp rush.txt zz-top.txt
[root@localhost ~]# ls
ADirectory  opeth.txt  rush.txt  zz-top.txt
```

همانطور که می‌بینید، با استفاده از دستور cp، یک کپی از فایل rush.txt ایجاد کردیم و آن را با نام zz-top.txt در فولدر کنونی ذخیره کردیم. از آنجایی که فایل را در فولدر کنونی کپی کردیم، مجبور بودیم یک نام جدید به این فایل بدهیم. اگر فایل را به مکانی دیگر کپی می‌کردیم، می‌توانستیم فقط آدرس دایرکتوری یا مکان مورد نظر را وارد کرده تا فایل با همین نام rush.txt در آن موقعیت کپی شود.

ما می‌توانیم با دستور cp، فایل‌های موجود در سایر موقعیت‌های سیستم را نیز کپی کنیم. برای مثال، بیایید فایل /etc/postfix/main.cf را با نام copy-main.cf در موقعیت /tmp کپی کنیم:

```
[root@localhost ~]# ls -1 /etc/postfix/*.cf
/etc/postfix/main.cf
/etc/postfix/master.cf
[root@localhost ~]# cp /etc/postfix/main.cf /tmp/copy-main.cf
[root@localhost ~]# ls /tmp/
copy-main.cf
```

همانطور که می‌بینید، ابتدا در دایرکتوری /etc/postfix دنبال فایل‌هایی که با پسوند cf تمام می‌شوند گشتیم. سپس آدرس دقیق (absolute path) فایل main.cf را به cp دادیم و سپس موقعیتی که می‌خواهیم فایل را در آن کپی کنیم (/tmp)، به علاوه‌ی نامی که می‌خواهیم فایل کپی شده داشته باشد (copy-main.cf) را وارد کردیم. در اینجا، اگر نام خاصی را وارد نمی‌کردیم، فایل کپی شده همان نام فایل اصلی را به خود می‌گرفت.

همانطور که می‌بینید، در حال حاضر دستور cp پس از انجام عملیات کپی، هیچ چیزی در خروجی به ما نشان نمی‌دهد. برای این که کاری کنیم که این دستور گزارشی از عملیات کپی خود به ما نشان دهد، از آپشن v- استفاده می‌کنیم. بیایید این بار کلیه‌ی فایل‌هایی که در موقعیت /etc با پسوند conf. وجود دارند را در دایرکتوری /tmp کپی کنیم:

```
[root@localhost ~]# cp -v /etc/*.conf /tmp/
‘/etc/asound.conf’ -> ‘/tmp/asound.conf’
‘/etc/dracut.conf’ -> ‘/tmp/dracut.conf’
‘/etc/e2fsck.conf’ -> ‘/tmp/e2fsck.conf’
‘/etc/host.conf’ -> ‘/tmp/host.conf’
‘/etc/kdump.conf’ -> ‘/tmp/kdump.conf’
‘/etc/krb5.conf’ -> ‘/tmp/krb5.conf’
‘/etc/ld.so.conf’ -> ‘/tmp/ld.so.conf’
‘/etc/libaudit.conf’ -> ‘/tmp/libaudit.conf’
‘/etc/libuser.conf’ -> ‘/tmp/libuser.conf’
‘/etc/locale.conf’ -> ‘/tmp/locale.conf’
‘/etc/logrotate.conf’ -> ‘/tmp/logrotate.conf’
‘/etc/man_db.conf’ -> ‘/tmp/man_db.conf’
‘/etc/mke2fs.conf’ -> ‘/tmp/mke2fs.conf’
‘/etc/nsswitch.conf’ -> ‘/tmp/nsswitch.conf’
‘/etc/resolv.conf’ -> ‘/tmp/resolv.conf’
```

```
‘/etc/rsyncd.conf’ -> ‘/tmp/rsyncd.conf’
‘/etc/rsyslog.conf’ -> ‘/tmp/rsyslog.conf’
‘/etc/sestatus.conf’ -> ‘/tmp/sestatus.conf’
‘/etc/sudo.conf’ -> ‘/tmp/sudo.conf’
‘/etc/sudo-ldap.conf’ -> ‘/tmp/sudo-ldap.conf’
‘/etc/sysctl.conf’ -> ‘/tmp/sysctl.conf’
‘/etc/tcsd.conf’ -> ‘/tmp/tcsd.conf’
‘/etc/vconsole.conf’ -> ‘/tmp/vconsole.conf’
‘/etc/yum.conf’ -> ‘/tmp/yum.conf’
```

همانطور که می‌بینید، این بار دستور cp، عملیات کپی خود را به صورت کامل گزارش کرد.

برای این که با استفاده از دستور cp بتوانیم یک دایرکتوری و کلیه‌ی محتویات آن را کپی کنیم، باید از آپشن r- استفاده کنیم. بیایید دایرکتوری /etc/postfix و کلیه‌ی محتویات آن را در یک دایرکتوری در موقعیت کنونی (در حال حاضر ~) کپی کنیم:

```
[root@localhost ~]# cp -vr /etc/postfix/ copy_postfix
‘/etc/postfix/’ -> ‘copy_postfix’
‘/etc/postfix/access’ -> ‘copy_postfix/access’
‘/etc/postfix/canonical’ -> ‘copy_postfix/canonical’
‘/etc/postfix/generic’ -> ‘copy_postfix/generic’
‘/etc/postfix/header_checks’ -> ‘copy_postfix/header_checks’
‘/etc/postfix/main.cf’ -> ‘copy_postfix/main.cf’
‘/etc/postfix/master.cf’ -> ‘copy_postfix/master.cf’
‘/etc/postfix/relocated’ -> ‘copy_postfix/relocated’
‘/etc/postfix/transport’ -> ‘copy_postfix/transport’
‘/etc/postfix/virtual’ -> ‘copy_postfix/virtual’
```

همانطور که می‌بینید با استفاده از آپشن r-، به cp گفتیم که کلیه‌ی محتویات /etc/postfix را درون یک دایرکتوری جدید به نام copy_postfix کپی کند. اگر فولدر مشخص شده موجود نباشد، خود cp آن را ایجاد می‌کند. بیایید نگاهی به دایرکتوری‌های موجود در موقعیت کنونی بیاندازیم:

```
[root@localhost ~]# ls -1
ADirectory
copy_postfix
opeth.txt
rush.txt
woo.txt
zz-top.txt
[root@localhost ~]# ls copy_postfix/
access     generic        main.cf    relocated  virtual
canonical  header_checks  master.cf  transport
```

همانطور که می‌بینید همه‌ی فایل‌ها، در این دایرکتوری کپی شده‌اند.

در حالت عادی، اگر فایلی که به عنوان DEST به دستور cp می‌دهید در موقعیت مشخص شده وجود داشته باشد، cp فایل جدید را روی فایل قبلی می‌نویسد یا به عبارت دیگر آن را overwrite می‌کند. برای جلوگیری از چنین مشکلاتی، بهتر است هنگام استفاده از دستور cp، از آپشن i- نیز استفاده کنید. اضافه کردن این آپشن، باعث می‌شود که cp در صورت وجود فایل یا دایرکتوری که نامی یکسان با نام مورد نظر شما دارد (DEST)، به شما هشدار داده و درخواست تایید عملیات دهد. برای مثال بیایید سعی کنیم بار دیگر محتویات دایرکتوری /etc/postfix را درون دایرکتوری copy_postfix که قبلا ایجاد کردیم کپی کنیم. این بار از آپشن i- نیز استفاده می‌کنیم:

```
[root@localhost ~]# cp -ir /etc/postfix/ copy_postfix
```

همانطور که می‌بینید، اتفاق خاصی نیافتاد و دستور cp از ما درخواست تایید برای overwrite کردن فایل‌ها نکرد. این بدین معنی است که cp در انجام عملیات کپی خود، موفق بوده است. پس بیایید نگاهی به محتویات دایرکتوری copy_postfix بیاندازیم:

```
[root@localhost ~]# ls -l copy_postfix/
total 148
-rw-r--r--. 1 root root 20876 Jul  3 21:27 access
-rw-r--r--. 1 root root 11883 Jul  3 21:27 canonical
-rw-r--r--. 1 root root 10106 Jul  3 21:27 generic
-rw-r--r--. 1 root root 21545 Jul  3 21:27 header_checks
-rw-r--r--. 1 root root 27176 Jul  3 21:27 main.cf
-rw-r--r--. 1 root root  6105 Jul  3 21:27 master.cf
drwxr-xr-x. 2 root root   154 Jul  3 21:27 postfix
-rw-r--r--. 1 root root  6816 Jul  3 21:27 relocated
-rw-r--r--. 1 root root 12549 Jul  3 21:27 transport
-rw-r--r--. 1 root root 12696 Jul  3 21:27 virtual
```

همانطور که می‌بینید، یک دایرکتوری جدید به نام postfix داخل دایرکتوری copy_postfix ایجاد شده است که حاوی کلیه‌ی فایل‌های موجود در /etc/postfix می‌باشد. برای درک دلیل این اتفاق، باید به یاد آوریم که دستور cp، در صورت عدم ارائه‌ی یک نام جدید برای دایرکتوری یا فایل DEST، اقدام به استفاده از نام خود دایرکتوری اصلی یا SOURCE می‌کند. در اینجا، ما به cp نام فایل یا دایرکتوری جدیدی را ندادیم، بلکه به cp گفتیم که محتویات را درون دایرکتوری copy_postfix که از قبل در این مکان وجود داشت، کپی کند. این امر سبب شد که cp، محتویات دایرکتوری SOURCE را با همان نام SOURCE، داخل دایرکتوری copy_postfix ذخیره کند.

روش‌های متفاوتی برای رفع این مشکل وجود دارد. در این مثال، ما صرفا می‌توانیم با استفاده از globbing، به جای کل دایرکتوری postfix، کلیه‌ی محتویات آن را داخل دایرکتوری copy_postfix کپی کنیم. یعنی:

```
[root@localhost ~]# cp -ir /etc/postfix/* copy_postfix/
cp: overwrite ‘copy_postfix/access’? n
cp: overwrite ‘copy_postfix/canonical’? n
cp: overwrite ‘copy_postfix/generic’? n
cp: overwrite ‘copy_postfix/header_checks’? n
cp: overwrite ‘copy_postfix/main.cf’? n
cp: overwrite ‘copy_postfix/master.cf’? n
cp: overwrite ‘copy_postfix/relocated’? n
cp: overwrite ‘copy_postfix/transport’? n
cp: overwrite ‘copy_postfix/virtual’? n
```

همانطور که می‌بینید، چون کلیه‌ی فایل‌های موجود در /etc/postfix در دایرکتوری copy_postfix وجود داشتند، آپشن i- باعث شد که این دستور از ما در خواست اجازه برای overwrite کردن آنها کند؛ که ما چنین اجازه‌ای را به این دستور ندادیم.

روش دیگر، و شاید بهتر برای overwrite کردن یک دایرکتوری، استفاده از آپشن T- می‌باشد. این آپشن باعث می‌شود که دستور cp، به دایرکتوری copy_postfix به عنوان یک دایرکتوری جدید DEST نگاه کند. یعنی:

```
[root@localhost ~]# cp -Tir /etc/postfix/ copy_postfix/
...
cp: overwrite ‘copy_postfix/transport’? n
cp: overwrite ‘copy_postfix/virtual’? n
```

همانطور که می‌بینید، این بار بدون استفاده از globbing، موفق به انجام هدف خود شدیم.
نکته‌ی از این بخش می‌توانیم بفهمیم، این است که ما مفهومی به نام overwrite کردن یک دایرکتوری نداریم، بلکه فقط فایل‌های داخل دایرکتوری DEST، در صورت داشتن نامی یکسان با فایل‌های موجود در SOURCE، overwrite می‌شوند. برای مثال اگر در دایرکتوری copy_postfix، فایلی به نام apustaja.txt داشتیم، این فایل در طی هیچکدام از دستورات وارد شده، overwrite نمیشد. این امر در کارهایی نظیر بک‌آپ‌گیری و... برای ما مهم است، چون ممکن است باعث شود چند کپی از یک فایل که صرفا دچار تغییر نام شده داشته باشیم.

**نکته:** در اکثر توزیع‌ها، دستور cp به صورت اتوماتیک با آپشن i- اجرا می‌شود؛ با این حال، همیشه از صحت این امر با نگاه به خروجی دستور alias اطمینان حاصل کنید.

یکی دیگر از آپشن‌های جالب دستور cp، آپشن u- می‌باشد. این آپشن باعث می‌شود که cp تنها در صورتی فایل‌های هم‌نام در DEST را overwrite کند که فایل SOURCE، جدیدتر از فایل DEST باشد. مثلا در مثال بالا، اگر بخواهیم کلیه‌ی فایل‌های موجود در /etc/postfix را باردیگر درون copy_postfix کپی کنیم، تنها فایل‌هایی overwrite می‌شوند که به تازگی در /etc/postfix دچار تغییر شده باشند (مثلا متنی به آنها اضافه شده باشد). ما تست عملکرد این آپشن را به خودتان میسپاریم و پیشنهاد می‌کنیم manpage دستور cp را مطالعه کنید.

## جابه‌جایی (Cut کردن) و تغییر نام فایل‌ها و دایرکتوری‌ها با *mv*

برای جابه‌جایی یا تغییر نام یک فایل یا دایرکتوری، از دستور mv استفاده می‌کنیم. syntax این دستور بسیار شبیه به دستور cp می‌باشد؛ یعنی به طور کلی:

```
mv [OPTION] SOURCE DEST
```

بسیاری از آپشن‌های این دستور، شبیه به آپشن‌های دستور cp می‌باشند. یعنی آپشن‌های v-، i- و u- دقیقا همان کاری که در cp شرح دادیم را انجام می‌دهند. استفاده از دستور mv بسیار ساده می‌باشد. برای مثال:

```
[root@localhost ~]# ls -1
ADirectory
copy_postfix
opeth.txt
rush.txt
woo.txt
zz-top.txt
[root@localhost ~]# mv copy_postfix/ ADirectory/
[root@localhost ~]# ls
ADirectory
opeth.txt
rush.txt
woo.txt
zz-top.txt
[root@localhost ~]# ls -1 ADirectory/
AnotherDir
copy_postfix
DreamTheater.txt
```

همانطور که می‌بینید، ما با استفاده از دستور ls -1 نگاهی به محتویات دایرکتوری ~ انداختیم و سپس با استفاده از دستور mv، دایرکتوری copy_postfix را به داخل دایرکتوری ADirectory جابه‌جا کردیم. همانطور که می‌بینید دایرکتوری copy_postfix کاملا از دایرکتوری ~ محو شده و اکنون داخل دایرکتوری ADirectory قرار گرفته است. همچنین بر خلاف دستور cp، ما نیازی به استفاده از آپشن r- برای کار با دایرکتوری‌ها نداریم. پس می‌توان گفت که عملکرد دستور mv، بسیار شبیه به عملکرد Cut در سیستم‌عامل ویندوز می‌باشد.

ما می‌توانیم از دستور mv برای تغییر نام یا rename کردن یک فایل یا دایرکتوری نیز استفاده کنیم. برای مثال:

```
[root@localhost ~]# ls -1
ADirectory
opeth.txt
rush.txt
woo.txt
zz-top.txt
[root@localhost ~]# mv rush.txt 2112.txt
[root@localhost ~]# ls -1
2112.txt
ADirectory
opeth.txt
woo.txt
zz-top.txt
[root@localhost ~]# mv ADirectory/ BDirectory
[root@localhost ~]# ls -1
2112.txt
BDirectory
opeth.txt
woo.txt
zz-top.txt
[root@localhost ~]# mv woo.txt opeth.txt
mv: overwrite ‘opeth.txt’? n
```

همانطور که می‌بینید ما ابتدا با استفاده از دستور mv، نام فایل rush.txt را به 2112.txt تغییر دادیم. سپس با استفاده از همین دستور، نام دایرکتوری ADirectory را به BDirectory تغییر دادیم و در نهایت سعی کردیم نام فایل woo.txt را به opeth.txt تغییر دهیم، اما از آنجایی که در این مکان یک فایل به نام opeth.txt وجود داشت، mv به ما هشدار داد و از ما خواست که در صورت تمایل به overwrite کردن این فایل، عملیات را تایید کنیم.

توجه کنید که در حالت عادی، چنین هشداری فقط در صورت ارائه‌ی آپشن i- به ما داده میشد؛ اما همانطور که در مورد cp نیز گفتیم، در اکثر توزیع‌ها، دستور mv به صورت پیش‌فرض با آپشن i- اجرا می‌شود.

همچنین ما می‌توانیم در یک حرکت، هم یک فایل را جابه‌جا کرده و هم نام آن را تغییر دهیم:

```
[root@localhost ~]# ls
2112.txt  BDirectory  opeth.txt  woo.txt  zz-top.txt
[root@localhost ~]# mv woo.txt BDirectory/kek.txt
[root@localhost ~]# ls -1 BDirectory/
AnotherDir
copy_postfix
DreamTheater.txt
kek.txt
```

همانطور که می‌بینید ما با اجرای یک دستور، فایل woo.txt را به داخل دایرکتوری BDirectory منتقل کرده و حین انجام این حرکت، نام آن را نیز به kek.txt تغییر دادیم.

## پاک‌کردن فایل‌ها و دایرکتوری‌ها با *rm*

برای پاک‌کردن فایل‌ها و دایرکتوری‌ها، از ابزار انعطاف‌پذیر و کارآمد rm استفاده می‌کنیم. بیایید بدون هیچ‌گونه اتلاف وقت، به سراغ استفاده از این دستور برویم:

```
[root@localhost ~]# ls -1
2112.txt
BDirectory
opeth.txt
zz-top.txt
[root@localhost ~]# rm 2112.txt
rm: remove regular empty file ‘2112.txt’? y
```

همانطور که می‌بینید، با ارائه‌ی نام فایل مورد نظر به دستور rm، این دستور از ما درخواست تایید پاک‌کردن فایل مشخص شده را داد و ما با وارد کردن حرف y و زدن دکمه‌ی Enter، عملیات پاک‌کردن این فایل را تایید کردیم. بیایید از پاک‌شدن این فایل اطمینان حاصل کنیم:

```
[root@localhost ~]# ls -1
BDirectory
opeth.txt
zz-top.tx
```

**نکته:** دلیل این که rm هنگام پاک‌کردن فایل از ما اجازه می‌خواهد، این است که این دستور در اکثر توزیع‌ها به صورت پیش‌فرض با آپشن -i اجرا می‌شود. می‌توانید این امر را با وارد کردن دستور alias و مشاهده‌ی خروجی آن، مشاهده کنید.

حال بیایید یک دایرکتوری ایجاد کنیم و سپس آن را پاک کنیم:

```
[root@localhost ~]# mkdir deleteMe
[root@localhost ~]# ls -1
BDirectory
deleteMe
opeth.txt
zz-top.txt
[root@localhost ~]# rm deleteMe
rm: cannot remove ‘deleteMe’: Is a directory
```

همانطور که می‌بینید rm به ما می‌گوید که توانایی پاک‌کردن دایرکتوری‌ها را ندارد. برای این که rm بتواند دایرکتوری را پاک کند، باید از آپشن -r استفاده کنیم:

```
[root@localhost ~]# rm -r deleteMe
rm: remove directory ‘deleteMe’? y
[root@localhost ~]# ls -1
BDirectory
opeth.txt
zz-top.txt
```

همانطور که می‌بینید با استفاده از آپشن -r، دستور rm پس از درخواست تایید برای حذف، دایرکتوری deleteMe را کاملا پاک کرد.

دقت داشته باشید که آپشن -r، به صورت recursive عمل می‌کند، یعنی هم دایرکتوری، هم فایل‌های داخل

دایرکتوری و هم دایرکتوری‌های داخل دایرکتوری و فایل‌های داخل آنها را پاک می‌کند. اگر صرفا هدفمان پاک کردن دایرکتوری‌های خالی روی سیستم باشد، بهتر است به جای آپشن r-، از آپشن d- استفاده کنیم. این آپشن فقط دایرکتوری‌های خالی را پاک می‌کند و کاری با دایرکتوری‌هایی که فایل یا دایرکتوری درون خود دارند، ندارد:

```
[root@localhost ~]# ls BDirectory/
AnotherDir  copy_postfix  DreamTheater.txt  kek.txt
[root@localhost ~]# rm -d BDirectory/
rm: cannot remove ‘BDirectory/’: Directory not empty
```

همانطور که می‌بینید، از آنجایی که دایرکتوری BDirectory خالی نبود، rm با آپشن d- آن را پاک نکرد. این آپشن در کارهایی نظیر اسکریپتینگ خیلی به کار می‌آید. مثلا ما می‌توانیم با استفاده از globbing، کاری کنیم که rm در مسیرهای مشخص شده، فقط دایرکتوری‌های خالی را پاک کند.

حال که داریم در مورد دایرکتوری‌های خالی صحبت می‌کنیم، بد نیست که با دستور rmdir نیز آشنا شویم. دستور rmdir، دستوری است که فقط دایرکتوری‌های خالی را پاک می‌کند. برای مثال:

```
[root@localhost ~]# mkdir IAmEmpty
[root@localhost ~]# ls -1
BDirectory
IAmEmpty
opeth.txt
[root@localhost ~]# rmdir IAmEmpty/
[root@localhost ~]# ls -1
BDirectory
opeth.txt
```

همانطور که می‌بینید، ما یک دایرکتوری خالی به نام IAmEmpty ایجاد کردیم و سپس با استفاده از rmdir آن را پاک کردیم. اگر از آپشن v- استفاده کنیم، این دستور گزارشی از عملکرد خود نیز در خروجی به ما می‌دهد. ما تست این امر را به خودتان می‌سپاریم.

اگر به این دستور یک دایرکتوری که خالی نیست بدهیم، یک پیغام خطا در خروجی دریافت می‌کنیم:

```
[root@localhost ~]# rmdir BDirectory/
rmdir: failed to remove ‘BDirectory/’: Directory not empty
```

اگر بخواهیم دستور rm عملیاتی که انجام می‌دهد را در خروجی به ما گزارش دهد، از آپشن v- استفاده می‌کنیم. برای مثال:

```
[root@localhost ~]# ls -1
BDirectory
opeth.txt
zz-top.txt
[root@localhost ~]# rm -v zz-top.txt
rm: remove regular empty file ‘zz-top.txt’? y
removed ‘zz-top.txt’
[root@localhost ~]# ls -1
BDirectory
opeth.txt
```

همانطور که می‌بینید با اعمال آپشن v-، دستور rm پس از دریافت تاییدیه برای حذف فایل zz-top.txt، به ما گزارش داد که فایل zz-top.txt را پاک کرده است.

اگر بخواهیم تعداد زیادی فایل را پاک کنیم، ممکن است ارائه‌ی تایید برای هر کدام از آنها، کار اعصاب‌خردکنی شود. با استفاده از آپشن f-، می‌توانیم از rm بخواهیم که برای پاک‌کردن فایل‌ها و دایرکتوری‌ها از ما درخواست تایید نکند. برای مثال:

```
[root@localhost ~]# ls -1
BDirectory
opeth.txt
newfile1.txt
newfile2.txt
woo
[root@localhost ~]# rm -f *
rm: cannot remove ‘BDirectory’: Is a directory
rm: cannot remove ‘woo’: Is a directory
[root@localhost ~]# ls -1
BDirectory
woo
```

همانطور که می‌بینید، با استفاده از آپشن f و وایلدکارد *، دستور rm بدون درخواست برای تایید، کلیه‌ی فایل‌های موجود در ~ را پاک کرد، اما به سراغ دایرکتوری‌ها نرفت. برای این که دایرکتوری‌ها و کلیه‌ی فایل‌ها و فولدرهای درون آن را بدون ارائه‌ی تایید پاک کنیم، از آپشن r- و f- استفاده می‌کنیم. این بار آپشن v- را نیز به آن می‌دهیم تا گزارشی از عملکرد را در خروجی مشاهده کنیم:

```
[root@localhost ~]# ls
BDirectory  woo
[root@localhost ~]# rm -vrf *
...
removed directory: ‘BDirectory/copy_posfix’
removed directory: ‘BDirectory/AnotherDir’
removed ‘BDirectory/DreamTheater.txt’
removed ‘BDirectory/kek.txt’
removed directory: ‘BDirectory’
removed directory: ‘woo’
```

همانطور که می‌بینید، با استفاده از آپشن f- و r-، دستور rm بدون درخواست تایید، کلیه‌ی فایل‌ها و دایرکتوری‌های موجود در ~ را پاک کرد. به دلیل اعمال آپشن v-، این دستور گزارشی از عملکرد خود در خروجی به ما نشان داد. توجه کنید که در globbing، وایلدکارد *، فایل‌های hidden را به ما باز نمی‌گرداند، پس در اینجا فایل‌های hidden ما پاک نشده‌اند.

**نکته:** اگر به rm آپشن I- را بدهیم، فقط هنگام پاک‌کردن بیش از ۳ فایل یا هنگام پاک‌کردن یک دایرکتوری، از ما درخواست تایید می‌کند. ممکن است در برخی شرایط استفاده از این آپشن، معقول‌تر از استفاده از آپشن f- باشد.

## فشرده‌سازی فایل‌ها

خیلی اوقات ممکن است فایل‌هایی نظیر فایل‌های بک‌آپ داشته باشیم که حجم زیادی دارند و در نتیجه فضای زیادی از دیسک را اشغال می‌کنند. ما می‌توانیم حجم اشغال شده توسط این فایل‌ها را با فشرده‌سازی آنها، کمتر کنیم. در این بخش، می‌خواهیم با برخی از ابزارهای فشرده‌سازی موجود در لینوکس آشنا شویم.

اما قبل از آن، بیایید خیلی سریع یک فایل ۲ گیگ‌بایتی روی سیستم خود ایجاد کنیم. فعلا نگران مفهوم دستور زیر نباشید و فقط آن را درون سیستم خود وارد کنید:

```
[root@localhost ~]# dd if=/dev/zero of=testfile bs=2G count=1
0+1 records in
0+1 records out
2147479552 bytes (2.1 GB) copied, 145.473 s, 14.8 MB/s
```

توجه کنید که ممکن است اجرای این دستور مدت زمان زیادی طول بکشد. این دستور، یک فایل دو گیبی‌بایتی که در آن فقط تعدادی زیادی عدد صفر وجود دارد را با نام testfile ایجاد می‌کند. ما بعدا با دستور dd بیشتر آشنا می‌شویم، پس فعلا نگران درک آن نباشید. بیایید از صحت ایجاد این فایل اطمینان حاصل کنیم:

```
[root@localhost ~]# ls -lh
total 2.0G
drwxr-xr-x. 3 root root   64 Jun 26 23:12 BDirectory
-rw-r--r--. 1 root root 2.0G Jun 28 11:40 testfile
drwxr-xr-x. 2 root root    6 Jun 26 23:10 woo
```

همانطور که می‌بینید، فایل testfile با حجم ۲ گیبی‌بایت ایجاد شده است. حال در مورد روش‌های متفاوت فشرده‌سازی صحبت می‌کنیم.

## فشرده‌سازی با *gzip*

ابزار gzip در سال ۱۹۹۲ توسعه داده شده و تا به امروز نیز یک ابزار مناسب برای فشرده‌سازی فایل‌ها می‌باشد. این ابزار که از الگوریتم LZ77 استفاده می‌کند، می‌تواند فایل‌های متنی را تا حدود ۶۰ الی ۷۰ درصد، فشرده‌سازی کند. برای فشرده‌سازی یک فایل با gzip، کافی است دستور gzip به علاوه‌ی نام فایلی که می‌خواهیم فشرده‌سازی کنیم را وارد ترمینال کنیم. یعنی:

```
[root@localhost ~]# gzip -v testfile
testfile:    99.9% -- replaced with testfile.gz
```

همانطور که می‌بینید، ما دستور gzip را با آپشن -v وارد کردیم و سپس نام فایلی که می‌خواستیم فشرده‌سازی کنیم را به این دستور دادیم. دلیل استفاده از آپشن -v، دریافت گزارشی از چگونگی انجام کار و درصد فشرده‌سازی در خروجی بود. همانطور که می‌بینید، به ما گزارش شده که دستور gzip توانسته این فایل را تا ۹۹,۹ درصد، فشرده‌سازی کند. بیایید این امر را بررسی کنیم:

```
[root@localhost ~]# ls -lh
total 2.0M
drwxr-xr-x. 3 root root   64 Jun 26 23:12 BDirectory
-rw-r--r--. 1 root root 2.0M Jun 28 11:40 testfile.gz
drwxr-xr-x. 2 root root    6 Jun 26 23:10 woo
```

همانطور که می‌بینید، gzip توانسته حجم فایل testfile را از ۲ گیبی‌بایت، به ۲ مبی‌بایت تقلیل دهد. نکته‌ی قابل توجه دیگر این است که gzip، فایل فشرده‌سازی شده را جایگزین فایل اصلی کرده است، یا به عبارتی دیگر، فایل اصلی testfile را حذف کرده و به جای آن، فایل فشرده‌سازی شده‌ی testfile.gz را قرار داده است.

برای این که یک فایل فشرده‌سازی شده توسط gzip را از حالت فشرده‌سازی شده خارج کنیم (decompress کنیم)، از دستور gunzip استفاده می‌کنیم:

```
[root@localhost ~]# gunzip -v testfile.gz
testfile.gz:       99.9% -- replaced with testfile
[root@localhost ~]# ls -lh
total 2.0G
drwxr-xr-x. 3 root root   64 Jun 26 23:12 BDirectory
-rw-r--r--. 1 root root 2.0G Jun 28 11:40 testfile
```

```
drwxr-xr-x. 2 root root    6 Jun 26 23:10 woo
```

همانطور که می‌بینید، این دستور فایل testfile.gz را از حالت فشرده در آورد و فایل اکنون همان حجم ۲ گیبی‌بایتی اولیه را دارد. این دستور آپشن‌های بیشتری نیز دارد، اما ما به توضیح آنها نمی‌پردازیم و شما را تشویق به مطالعه‌ی manpage این دستور می‌کنیم.

## فشرده‌سازی با *bzip2*

ابزار bzip2 در سال ۱۹۹۶ توسعه داده شده و نرخ فشرده‌سازی بهتری نسبت به برنامه‌ی gzip دارد؛ البته این نرخ فشره‌سازی بهتر، باعث کندتر بودن این برنامه نسبت به gzip شده است. bzip2 از روش‌ها و الگوریتم‌های متفاوتی جهت فشرده‌سازی اطلاعات استفاده می‌کند. جالب است بدانید که تا سال ۲۰۱۳، کرنل لینوکس توسط این برنامه فشرده‌سازی و در اختیار عموم قرار می‌گرفت.

برای فشرده‌سازی یک فایل توسط این برنامه، کافی است دستور bzip2 به علاوه‌ی نام فایلی که می‌خواهیم فشرده‌سازی کنیم را درون ترمینال وارد کنیم. فقط به خاطر داشته باشید که این دستور نیز فایل فشرده شده را جایگزین فایل اصلی می‌کند:

```
[root@localhost ~]# bzip2 -v testfile
  testfile: 1410958.970:1,  0.000 bits/byte,
  100.00% saved, 2147479552 in, 1522 out.
```

همانطور که می‌بینید، استفاده از این دستور بسیار شبیه به استفاده از دستور gzip می‌باشد پس به توضیح بیشتر آن نمی‌پردازیم. بیایید حجم فایل فشرده شده را بررسی کنیم:

```
[root@localhost ~]# ls -lh
total 4.0K
drwxr-xr-x. 3 root root   64 Jun 26 23:12 BDirectory
-rw-r--r--. 1 root root 1.5K Jun 28 11:40 testfile.bz2
drwxr-xr-x. 2 root root    6 Jun 26 23:10 woo
```

همانطور که می‌بینید، فایل testfile پس از فشرده‌سازی، فقط ۱٫۵ کیبی‌بایت حجم دارد. اگر حجم این فایل را با حجم فایل فشرده‌سازی شده توسط gzip مقایسه کنید، می‌بینید که bzip2 توانسته این فایل را بهتر از gzip، فشرده‌سازی کند.

برای این که یک فایل فشرده‌سازی شده توسط bzip2 را از حالت فشرده در آوریم (decompress کنیم)، از دستور bunzip2 به علاوه‌ی نام فایل فشرده شده توسط bzip2، استفاده می‌کنیم:

```
[root@localhost ~]# bunzip2 -v testfile.bz2
  testfile.bz2: done
[root@localhost ~]# ls -lh
total 2.0G
drwxr-xr-x. 3 root root   64 Jun 26 23:12 BDirectory
-rw-r--r--. 1 root root 2.0G Jun 28 11:40 testfile
drwxr-xr-x. 2 root root    6 Jun 26 23:10 woo
```

این دستور آپشن‌های زیادی دارد و پیشنهاد می‌کنیم که در صورت تمایل، به مطالعه‌ی manpage آن بپردازید.

## فشرده‌سازی با *xz*

ابزار xz در سال ۲۰۰۹ توسعه داده شده و تبدیل به یکی از ابزارهای محبوب جهت فشرده‌سازی فایل‌ها شده است. به دلیل استفاده از الگوریتم LZMA2، میزان فشرده‌سازی این ابزار، بالاتر از bzip2 و gzip می‌باشد.

جالب است بدانید که از سال ۲۰۱۳، کرنل لینوکس توسط xz فشرده‌سازی شده و توزیع می‌شود. برای فشرده‌سازی یک فایل توسط xz، کافی است مانند قبل، نام فایل را به دستور xz بدهیم. فقط به خاطر داشته باشید که این دستور نیز، فایل فشرده شده را جایگزین فایل اصلی می‌کند:

```
[root@localhost ~]# xz -v testfile
testfile (1/1)
  100 %     305.2 KiB / 2,048.0 MiB = 0.000    35 MiB/s       0:58
```

همانطور که می‌بینید استفاده از این دستور نیز مانند دستورهای قبلی می‌باشد، با این فرق که آپشن v- در این دستور، نمای زیباتری دارد و اطلاعات بیشتری به ما می‌دهد. بیایید حجم فایل فشرده شده را بررسی کنیم:

```
[root@localhost ~]# ls -lh
total 308K
drwxr-xr-x. 3 root root   64 Jun 26 23:12 BDirectory
-rw-r--r--. 1 root root 306K Jun 28 11:40 testfile.xz
drwxr-xr-x. 2 root root    6 Jun 26 23:10 woo
```

همانطور که می‌بینید حجم این فایل پس از فشرده‌سازی، ۳۰۶ کیبی‌بایت شده است. مگر ما نگفتیم که xz بهتر از bzip2 فشرده‌سازی می‌کند؟ پس چرا اکنون حجم فایل فشرده‌سازی شده با xz بالاتر از حجم فایل فشرده‌سازی شده با bzip2 است؟

نکته‌ای که باید به آن توجه کنید این است که محتویات فایلی که می‌خواهیم فشرده‌سازی کنیم، بسیار بر روی عملکرد یک روش فشرده‌سازی تاثیر دارد. برخی از برنامه‌ها و الگوریتم‌ها عملکرد بهتری هنگام فشرده‌سازی یک نوع فایل خاص دارند (مثل bzip2 که فایلی ساختگی ما را بهتر از سایر برنامه‌ها فشرده‌سازی می‌کند). پس وقتی می‌گوییم xz از نظر فشرده‌سازی، عملکرد بهتری نسبت به bzip2 و gzip دارد، داریم به صورت کلی صحبت می‌کنیم؛ در واقع منظور ما این است که در اکثر مواقع، xz عملکرد بهتری دارد. به طور کلی، دلیل این که ما در اینجا توانسته‌ایم فایلی ۲ گیبی‌بایتی را تبدیل به یک فایل چند کیبی‌بایتی کنیم، این است که فایلی که ما ایجاد کرده‌ایم فقط یک سری عدد صفر درون خود دارد و در نتیجه ابزارهای فشرده‌سازی به راحتی می‌توانند چنین فایلی را فشرده کنند. پس به عبارت دیگر، این فایل ساختگی نمی‌تواند معیار خیلی مناسبی برای سنجش عملکرد روش‌های فشرده‌سازی باشد.

برای این که فایل فشرده‌سازی شده توسط xz را از حالت فشرده در آوریم (decompress کنیم)، از دستور unxz استفاده می‌کنیم:

```
[root@localhost ~]# unxz -v testfile.xz
testfile.xz (1/1)
  100 %     305.2 KiB / 2,048.0 MiB = 0.000   283 MiB/s       0:07
[root@localhost ~]# ls -lh
total 308K
drwxr-xr-x. 3 root root   64 Jun 26 23:12 BDirectory
-rw-r--r--. 1 0 0 2.0G Jun 29 12:49 testfile
drwxr-xr-x. 2 root root    6 Jun 26 23:10 woo
```

در نهایت، مطالعه‌ی manpage این دستور نیز پیشنهاد می‌شود.

## فشرده‌سازی با zip

ابزار zip یکی از ابزارهای فشرده‌سازی می‌باشد که بر خلاف سایر ابزارهایی که تا به اینجا معرفی کردیم، می‌تواند چندین فایل را در قالب یک فایل قرار داده و فشرده کند. اکثر ما با این فرمت آشنا هستیم، چرا که به احتمال زیاد در ویندوز نیز از آن استفاده کرده‌ایم. این برنامه چندین فایل را تبدیل به یک فایل، که به آن آرشیو می‌گویند، کرده و سپس آن فایل را فشرده‌سازی می‌کند. همچنین می‌تواند یک دایرکتوری را به صورت کامل فشرده‌سازی کرده و در قالب یک فایل به ما تحویل دهد. تفاوت دیگر این ابزار با سایر ابزارهای فشرده‌سازی در این است که zip، فایل‌های فشرده‌سازی شده را جایگزین فایل‌های اصلی نمی‌کند.

برای استفاده از این ابزار، کافی است دستور zip را وارد کرده، سپس نام فایلی که میخواهیم ایجاد کنیم را مشخص کرده و پس از آن، کلیه‌ی فایل‌هایی که می‌خواهیم فشرده کنیم و در قالب یک فایل قرار دهیم را وارد می‌کنیم. بیایید این دستور را امتحان کنیم تا عملکرد آن را بهتر درک کنیم. ما می‌خواهیم کل دایرکتوری را در قالب یک فایل فشرده شده‌ی zip به دست آوریم، پس:

```
[root@localhost ~]# zip etc.zip /etc/
  adding: etc/ (stored 0%)
```

همانطور که می‌بینید، ما دستور zip را وارد کرده، سپس نام فایلی که می‌خواهیم ایجاد کنیم را مشخص کرده (etc.zip) و در نهایت فایل یا دایرکتوری‌هایی که می‌خواهیم فشرده کنیم را وارد کردیم. در اینجا ما دایرکتوری /etc را در قالب فایل etc.zip ذخیره کردیم. بیایید نگاهی به حجم دایرکتوری /etc و نگاهی به حجم فایل etc.zip بیاندازیم:

```
[root@localhost ~]# du -hs /etc/
31M   /etc/
[root@localhost ~]# ls -lh
total 308K
drwxr-xr-x. 3 root root   64 Jun 26 23:12 BDirectory
-rw-r--r--. 1 0 0  158 Jun 29 16:17 etc.zip
drwxr-xr-x. 2 root root    6 Jun 26 23:10 woo
```

همانطور که می‌بینید، دایرکتوری /etc در ابتدا ۳۸ مبی‌بایت حجم داشته و پس از فشرده‌سازی توسط zip، حجم آن به ۱۵۸ بایت تقلیل یافته است.

ما گفتیم که با این دستور، می‌توانیم چندین فایل و دایرکتوری را نیز فشرده‌سازی کرده و در قالب یک فایل قرار دهیم. بیایید فایل testfile و دایرکتوری /etc و /boot را با استفاده از این دستور فشرده‌سازی کنیم و در قالب یک فایل قرار دهیم:

```
[root@localhost ~]# zip -v cocktailfile.zip testfile /etc/ /boot/
adding:
testfile.....................................................................
.............................................................................
.........................   (in=2147479552) (out=2084083) (deflated 100%)
  adding: etc/ (in=0) (out=0) (stored 0%)
  adding: boot/       (in=0) (out=0) (stored 0%)
total bytes=2147479552, compressed=2084083 -> 100% savings
```

همانطور که می‌بینید، ما مانند قبل دستور zip را وارد کرده و پس از آن نام فایلی که می‌خواهیم ایجاد کنیم را مشخص کردیم (cocktailfile.zip) و سپس فایل testfile، دایرکتوری /etc و /boot را با قرار دادن یک فاصله در میانشان، به این دستور دادیم. این بار آپشن v- را نیز اضافه کردیم تا گزارشی از عملکرد را در خروجی مشاهده کنیم. بیایید ببینیم حجم فایل zip چقدر می‌باشد:

```
[root@localhost ~]# du -h cocktailfile.zip
2.0M  cocktailfile.zip
```

همانطور که می‌بینید، ما موفق شدیم فایل testfile و دو دایرکتوری /etc و /boot را در یک فایل قرار داده و حجم آن را به ۲ مبی‌بایت تقلیل دهیم.

ما می‌توانیم محتویات فایل zip را بدون این که آن را از حالت فشرده خارج کنیم، مشاهده کنیم. برای این کار، از دستور unzip و آپشن l-، استفاده می‌کنیم:

```
[root@localhost ~]# unzip -l cocktailfile.zip
Archive:  cocktailfile.zip
  Length           Date       Time        Name
-----------       ----------  -----       ----
2147479552        06-29-2020  12:49       testfile
        0         06-29-2020  16:29       etc/
        0         03-20-2020  10:48       boot/
-----------                                 -------
2147479552                                  3 files
```

همانطور که می‌بینید، با استفاده از unzip و آپشن l-، محتویات این فایل zip را بدون اکسترکت کردن مشاهده کردیم.

برای این که فایل zip را از حالت فشرده در آوریم، از دستور unzip و نام فایل استفاده می‌کنیم:

```
[root@localhost ~]# unzip cocktailfile.zip
Archive:  cocktailfile.zip
replace testfile? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: testfile
   creating: etc/
   creating: boot/
```

همانطور که می‌بینید، این دستور در حالت عادی، محتویات خود را در موقعیت کنونی از حالت فشرده در می‌آورد. از آنجایی که در موقعیت کنونی، ما فایلی با نام testfile داشتیم، برنامه از ما پرسید که آیا مایل به جایگزینی فایل موجود درون zip با فایل موجود در این موقعیت هستیم یا نه. ما با وارد کردن حرف y، این عمل را تایید کردیم، اما شما می‌توانید هر کاری میخواهید بکنید 😐.

بیایید از صحت اکسترکت شدن فایل‌ها در موقعیت کنونی اطمینان حاصل کنیم:

```
[root@localhost ~]# ls -lh
total 2.1G
dr-xr-xr-x. 2 root root    6 Mar 20 10:48 boot
-rw-r--r--. 1 root root 2.0M Jun 29 16:30 cocktailfile.zip
drwxr-xr-x. 2 root root    6 Jun 29 16:29 etc
-rw-r--r--. 1 root root  158 Jun 29 16:17 etc.zip
-rw-r--r--. 1 root root 2.0G Jun 29 12:49 testfile
```

همانطور که می‌بینید، ما یک دایرکتوری boot و یک دایرکتوری etc در موقعیت کنونی داریم و فایل testfile هم در اینجا موجود می‌باشد.

**نکته:** اگر سناریو را دنبال کرده‌اید و اکنون قصد پارک کردن فولدر etc و boot موجود در ~ را دارید، هنگام استفاده از دستور rm، مراقب باشید که نام دایرکتوری‌ها را به صورت etc و boot وارد کنید، نه /etc و /boot؛ چرا که در آن صورت، بدبخت می‌شوید.

معمولا ترجیح می‌دهیم که محتویات یک فایل zip را در داخل یک دایرکتوری دیگر اکسترکت کنیم. برای این که به دستور unzip بگوییم که فایل zip را در یک دایرکتوری دیگر extract کند، از آپشن d- و موقعیت دایرکتوری مورد نظر استفاده می‌کنیم:

```
[root@localhost ~]# unzip cocktailfile.zip -d cocktail_directory
Archive:  cocktailfile.zip
  inflating: cocktail_directory/testfile
   creating: cocktail_directory/etc/
   creating: cocktail_directory/boot/
```

همانطور که می‌بینید، ما با استفاده از دستور unzip و وارد کردن نام فایل zip و سپس اعمال آپشن d- و وارد کردن مسیر دایرکتوری مورد نظر، از unzip خواستیم که محتویات فایل zip را در داخل دایرکتوری به نام cocktail_directory اکسترکت کند. اگر دایرکتوری مشخص شده وجود نداشته باشد، unzip اقدام به ایجاد این دایرکتوری می‌کند. بیایید از صحت انجام عملیات اطمینان حاصل کنیم:

```
[root@localhost ~]# ls -lh
total 2.1G
drwxr-xr-x. 4 root root   45 Jun 29 16:59 cocktail_directory
-rw-r--r--. 1 root root 2.0M Jun 29 16:30 cocktailfile.zip
-rw-r--r--. 1 root root  158 Jun 29 16:17 etc.zip
-rw-r--r--. 1 root root 2.0G Jun 29 12:49 testfile
[root@localhost ~]# ls cocktail_directory/
boot  etc  testfile
```

همانطور که می‌بینید، همه‌چیز در سر جای خود قرار دارد.

در سیستم‌های لینوکسی، زیاد از ابزار zip استفاده نمی‌شود و معمولا برای انجام کارهایی که zip می‌تواند انجام دهد، به سراغ استفاده از ابزار tar به علاوه‌ی سایر ابزارهای فشرده‌سازی می‌روند. دلیل این امر این است که فرمت zip، بسیاری از فراداده‌ها (metadataها) را حین فشرده‌سازی و ادغام درون یک فایل، از بین می‌برد.

## آرشیو کردن فایل‌ها با استفاده از tar

آرشیو کردن فایل‌ها، به معنی قرار دادن دسته‌ای از فایل‌ها و دایرکتوری درون یک فایل واحد می‌باشد. در لینوکس به این فایل واحد، آرشیو می‌گویند. جابه‌جایی و مدیریت آرشیوها کار بسیار ساده‌تری نسبت به جابه‌جایی و مدیریت تعداد زیادی از فایل‌ها می‌باشد. لینوکس از ابزارهای آرشیوکننده‌ی متعددی پشتیبانی می‌کند. ما در این جلسه، با ابزار tar آشنا می‌شویم.

ابزار tar، یکی از معروف‌ترین و پراستفاده‌ترین ابزارها برای آرشیوکردن فایل‌ها و دایرکتوری‌ها می‌باشد. در دنیای لینوکس، از فایل‌های tar فشرده‌سازی شده برای جابه‌جایی فایل‌ها و همچنین توزیع سورس‌کدها استفاده می‌کنند. به فایل‌های tar (یا آرشیوهای tar)، اصطلاحا tarball می‌گویند.

دستور tar، دستور پیچیده‌ای می‌باشد و آپشن‌های بسیار زیادی دارد، اما انجام کارهای معمول با این دستور، نسبتا ساده می‌باشد. ابتدا بیایید چند فایل ساختگی دیگر ایجاد کنیم. این بار می‌خواهیم دو فایل ۱ گیبی‌بایتی ایجاد کنیم. پس دو دستور زیر را در سیستم وارد می‌کنیم:

```
[root@localhost ~]# dd if=/dev/zero of=testfile2 bs=1G count=1
1+0 records in
1+0 records out
1073741824 bytes (1.1 GB) copied, 54.3406 s, 19.8 MB/s
```

```
[root@localhost ~]# dd if=/dev/zero of=testfile3 bs=1G count=1
1+0 records in
1+0 records out
1073741824 bytes (1.1 GB) copied, 66.7393 s, 16.1 MB/s
[root@localhost ~]# ls -lh
total 4.0G
-rw-r--r--. 1 root root 2.0G Jun 28 11:40 testfile
-rw-r--r--. 1 root root 1.0G Jun 29 10:27 testfile2
-rw-r--r--. 1 root root 1.0G Jun 29 10:29 testfile3
```

همانطور که می‌بینید، ما با استفاده از dd، دو فایل ساختگی دیگر به حجم ۱ گیبی‌بایت ایجاد کردیم و با احتساب فایلی که در بخش قبل ایجاد کردیم، اکنون ۳ فایل ساختگی داریم.

حال بیایید با دستور tar، این ۳ فایل را تبدیل به یک فایل کنیم؛ یا به عبارت دیگر، این ۳ فایل را آرشیو کنیم:

```
[root@localhost ~]# tar -cvf ar-testfile.tar testfile testfile2 testfile3
testfile
testfile2
testfile3
```

همانطور که می‌بینید، برای ایجاد یک فایل آرشیو، یا یک تاربال، از آپشن‌های cvf- استفاده کردیم. آپشن c، به دستور tar می‌گوید که باید یک فایل آرشیو ایجاد کند. اهمیت این آپشن را وقتی درک می‌کنید که بدانید دستور tar هم می‌تواند فایل‌ها را از حالت آرشیو در آورد، و هم می‌تواند فایل‌های درون یک تاربال را آپدیت کند. پس مشخص کردن این که می‌خواهیم tar چه کاری کند، بسیار مهم می‌باشد. آپشن v، به دستور tar می‌گوید که به صورت verbose عمل کند و نام فایل‌هایی که درون آرشیو یا تاربال قرار می‌دهد را در خروجی به ما گزارش دهد و با آپشن f، می‌توانیم نام فایل آرشیوی که tar قرار است ایجاد کند را مشخص کنیم.

پس از مشخص کردن آپشن cvf-، یک نام برای فایل آرشیوی که قرار است ایجاد شود مشخص می‌کنیم. در اینجا، ما نام ar-testfile.tar را انتخاب کردیم. پس از مشخص کردن نام فایل آرشیو، نام کلیه‌ی فایل‌هایی که می‌خواهیم آرشیو شوند را وارد می‌کنیم، که ما نام سه فایلی که ساخته بودیم را وارد کردیم.

> پس وقتی آپشن‌های tar را مشخص کردیم، ابتدا نام فایلی که tar قرار است ایجاد کند را مشخص می‌کنیم و پس از آن، نام فایل‌هایی که می‌خواهیم در آرشیو قرار گیرند را به tar می‌دهیم.

بیایید از صحت ایجاد فایل آرشیو، مطمئن شویم:

```
[root@localhost ~]# ls -lh
total 8.1G
-rw-r--r--. 1 root root 4.1G Jun 29 10:38 ar-tesfile.tar
-rw-r--r--. 1 root root 2.0G Jun 28 11:40 testfile
-rw-r--r--. 1 root root 1.0G Jun 29 10:27 testfile2
-rw-r--r--. 1 root root 1.0G Jun 29 10:29 testfile3
```

همانطور که می‌بینید، فایل ar-testfile.tar به درستی ایجاد شده است. حال از کجا می‌توانیم مطمئن شویم که سه فایل مورد نظر ما درون این فایل قرار دارند؟

ما در اینجا، به دنبال راهی هستیم که بتوانیم محتویات فایل آرشیو را بدون این که آن را از حالت آرشیو در آوریم، ببینیم. برای این کار، از آپشن tf- استفاده می‌کنیم:

```
[root@localhost ~]# tar -tf ar-tesfile.tar
testfile
testfile2
testfile3
```

همانطور که می‌بینید، با استفاده از آپشن tf، توانستیم محتویات این تاربال را مشاهده کنیم. آپشن t- به دستور tar می‌گوید که محتویات فایل آرشیو را لیست کند و آپشن f-، نام فایلی باید توسط tar باز شود را مشخص می‌کند. اگر بخواهیم اطلاعات بیشتری در مورد فایل‌های موجود درون تاربال پیدا کنیم، کافی است آپشن v- را نیز به آپشن tf اضافه کنیم. یعنی:

```
[root@localhost ~]# tar -tvf ar-tesfile.tar
-rw-r--r-- root/root 2147479552 2020-06-28 11:40 testfile
-rw-r--r-- root/root 1073741824 2020-06-29 10:27 testfile2
-rw-r--r-- root/root 1073741824 2020-06-29 10:29 testfile3
```

همانطور که می‌بینید، این بار اطلاعاتی نظیر permissionها، صاحب فایل‌ها، حجم فایل به بایت و تاریخ و ساعت ایجاد آن نیز به ما داده می‌شود. حال بیایید یک فایل دیگر بسازیم و سعی کنیم آن را به آرشیو ar-testfile اضافه کنیم. برای این کار ما یک فایل ۱۰۰ مبی‌بایتی با نام testfile4 ایجاد می‌کنیم:

```
[root@localhost ~]# dd if=/dev/zero of=testfile4 bs=100M count=1
1+0 records in
1+0 records out
104857600 bytes (105 MB) copied, 0.958513 s, 109 MB/s
[root@localhost ~]# ls -lh
total 4.1G
-rw-r--r--. 1 root root 4.1G Jun 29 10:38 ar-tesfile.tar
-rw-r--r--. 1 root root 2.0G Jun 28 11:40 testfile
-rw-r--r--. 1 root root 1.0G Jun 29 10:27 testfile2
-rw-r--r--. 1 root root 1.0G Jun 29 10:29 testfile3
-rw-r--r--. 1 root root 100M Jun 29 12:44 testfile4
```

حال برای اضافه کردن فایل testfile4 به آرشیو ar-testfile.tar، به صورت زیر از دستور tar استفاده می‌کنیم:

```
[root@localhost ~]# tar -rvf ar-tesfile.tar testfile4
testfile4
```

همانطور که می‌بینید، با استفاده از آپشن r-، به دستور tar گفتیم که فایل testfile4 را به آرشیو ar-testfile.tar اضافه کند. آپشن v و f همان عملکرد گذشته را دارند پس به توضیح آنها نمی‌پردازیم. بیایید از صحت قرار گرفتن فایل testfile4 درون آرشیو خود مطمئن شویم:

```
[root@localhost ~]# tar -tvf ar-tesfile.tar
-rw-r--r-- root/root 2147479552 2020-06-28 11:40 testfile
-rw-r--r-- root/root 1073741824 2020-06-29 10:27 testfile2
-rw-r--r-- root/root 1073741824 2020-06-29 10:29 testfile3
-rw-r--r-- root/root  104857600 2020-06-29 12:44 testfile4
```

همانطور که می‌بینید، این فایل درون آرشیو قرار گرفته است.

اگر بخواهیم فایل‌ها را از حالت آرشیو در آوریم یا آنها را اکسترکت کنیم، به صورت زیر عمل می‌کنیم. اما ابتدا لازم است بدانیم که اگر فایل‌هایی هم‌نام با فایل‌های موجود درون تاربال در موقعیت کنونی وجود داشته باشند، tar فایل‌های درون خود را جایگزین فایل‌های موجود در موقعیت کنونی می‌کند. یعنی در واقع اگر در موقعیت کنونی یک فایل به نام A داشته باشیم و درون تاربال نیز یک فایل به نام A داشته باشیم، اگر اقدام به اکسترکت کردن تاربال کنیم، فایل A که در موقعیت کنونی بوده حذف شده و فایل A موجود درون تاربال، جایگزین آن می‌شود. پس بهتر است ابتدا یک دایرکتوری جداگانه برای اکسترکت کردن فایل‌ها ایجاد کنیم. پس:

```
[root@localhost ~]# mkdir Extract
[root@localhost ~]# ls -lh
```

```
total 8.2G
-rw-r--r--. 1 root root 4.1G Jun 29 12:55 ar-testfile.tar
drwxr-xr-x. 2 root root    6 Jun 29 13:02 Extract
-rw-r--r--. 1 root root 2.0G Jun 29 12:49 testfile
-rw-r--r--. 1 root root 1.0G Jun 29 10:27 testfile2
-rw-r--r--. 1 root root 1.0G Jun 29 10:29 testfile3
-rw-r--r--. 1 root root 100M Jun 29 12:44 testfile4
```

سپس، از دستور tar به صورت زیر استفاده می‌کنیم:

```
[root@localhost ~]# tar -xvf ar-testfile.tar -C Extract/
testfile
testfile2
testfile3
testfile4
```

ما با استفاده از آپشن x، به tar می‌گوییم که باید عملیات اکسترکت را انجام دهد. آپشن v و f همان عملکرد قبلی را دارند، پس آنها را توضیح نمی‌دهیم. tar در حالت پیش‌فرض، فایل‌ها را فقط در دایرکتوری کنونی اکسترکت می‌کند. برای این که به tar بگوییم که فایل‌ها را در دایرکتوری دیگری اکسترکت کند، از آپشن C- (یا directory--) استفاده می‌کنیم و سپس موقعیت دایرکتوری را به آن می‌دهیم.

به نحوه‌ی استفاده از این آپشن‌ها نیز دقت کنید. همانطور که می‌بینید ما ابتدا xvf- را به کار بردیم، سپس نام فایل تاربال را به آن دادیم و سپس آپشن C- را اعمال کرده و آدرس دایرکتوری مورد نظر را مشخص کردیم. ما می‌توانستیم آپشن C را ادامه‌ی xvf نیز بنویسیم، اما در آن حالت، درک این دستور سخت‌تر می‌شد. اگر این کار را می‌کردیم، دستور نمایی نظیر زیر به خود می‌گرفت:

```
[root@localhost ~]# tar -xvfC ar-testfile.tar Extract/
...
```

در اینجا، از آنجایی که آپشن f قبل از آپشن C آمده، باید نام فایل تاربال را اول و پس از آن آدرس دایرکتوری را وارد کنیم. اگر C قبل از f آمده بود، باید برعکس عمل می‌کردیم.

حال بیایید از صحت اکسترکت شدن فایل‌های درون تاربال در دایرکتوری Extract اطمینان حاصل کنیم:

```
[root@localhost ~]# ls -lh Extract/
total 4.1G
-rw-r--r--. 1 root root 2.0G Jun 29 12:49 testfile
-rw-r--r--. 1 root root 1.0G Jun 29 10:27 testfile2
-rw-r--r--. 1 root root 1.0G Jun 29 10:29 testfile3
-rw-r--r--. 1 root root 100M Jun 29 12:44 testfile4
```

همانطور که می‌بینید، فایل‌ها به درستی درون دایرکتوری Extract قرار گرفته‌اند. بیایید یک قدم به عقب برداریم و نگاهی به حجم فایل آرشیو خود بیاندازیم:

```
[root@localhost ~]# du -h ar-testfile.tar
4.1G  ar-testfile.tar
```

همانطور که می‌بینید، این فایل در حال حاضر ٤,١ گیبی‌بایت حجم دارد. خیلی از اوقات ما از تاربال‌ها برای بک‌آپ گرفتن از تعداد زیادی فایل استفاده می‌کنیم، اما اگر حجم تاربال ما خیلی بالا باشد، به سرعت به مشکل کمبود فضا می‌خوریم. اینجاست که باید مواردی که در بخش فشرده‌سازی اطلاعات یاد گرفتیم را به یاد آوریم. ما می‌توانیم با استفاده از هر کدام از ابزارهای فشرده‌سازی، تاربال‌ها را فشرده‌سازی کنیم. اما راه ساده‌تری نیز وجود دارد. ما می‌توانیم با اعمال برخی آپشن‌ها به دستور tar، از این دستور بخواهیم که به صورت اتوماتیک و

با استفاده از ابزارهای فشرده‌سازی که قبلا معرفی کردیم، اقدام به فشرده‌سازی فایل‌ها کند و آنها را در قالب یک تاربال به ما تحویل دهد.

خوبی این امر این است که ما می‌توانیم با یک دستور، فایل‌های مورد نظر را آرشیو و همچنین فشرده‌سازی کنیم. برای ایجاد یک تاربال فشرده‌سازی شده، کافی است به همان آپشن cvf- که از آن برای ایجاد آرشیو استفاده می‌کردیم، یکی از آپشن‌های زیر را اضافه کنیم:

- z
  این آپشن، فایل‌ها را با gzip فشرده‌سازی می‌کند.
- j
  این آپشن فایل‌ها را bzip2 فشرده‌سازی می‌کند.
- J
  این آپشن، فایل‌ها را با xz فشرده‌سازی می‌کند.

بیایید چهار فایل ساختگی خود را تبدیل به یک آرشیو فشرده‌سازی شده کنیم. برای این کار:

```
[root@localhost ~]# tar cvfz comp-ar-testfile.tar testfile*
testfile
testfile2
testfile3
testfile4
```

همانطور که می‌بینید، ما با استفاده از آپشن cvf (که آن برای ایجاد آرشیو استفاده می‌کردیم) به علاوه‌ی آپشن z، از tar خواستیم که این ٤ فایل را با gzip فشرده‌سازی کرده و سپس درون یک آرشیو به نام cmp-ar-testfile.tar قرار دهد.

**نکته:** اگر دقت کرده باشید، در دستور بالا، بر خلاف همیشه، برای اعمال آپشن‌ها به tar، از علامت - استفاده نکردیم. استفاده از علامت - در اعمال آپشن‌ها به tar، اختیاری می‌باشد. یعنی ما چه در اعمال آپشن‌ها از علامت - استفاده کنیم و چه نکنیم، عملکرد tar یکسان خواهد بود.

اما در اینجا نکته‌ی مهمی وجود دارد. اگر آپشن‌های دستوری که در بالا مشاهده می‌کنید (tar cvfz) را با علامت - اعمال کنیم (tar -cvfz)، دستور tar به ما پیغام خطا می‌دهد. دلیل این امر، این است که در اکثر اوقات، هنگام اعمال آپشن‌ها با -، ترتیب قرارگیری آپشن‌ها برای tar مهم خواهند شد؛ در حالی که در صورت عدم استفاده از -، ترتیب اعمال آپشن‌ها برای tar اهمیتی نخواهد داشت.

در واقع دلیل این که tar -cvfz به ما خطا می‌دهد، این است که آپشن f (که نام فایل آرشیوی که میخواهیم ایجاد کنیم را مشخص می‌کند)، به عنوان آخرین آپشن اعمال نشده است. به عبارت دیگر، این آپشن به دلیل طبیعتی که دارد، باید در انتهای آپشن‌ها بیاید، وگرنه tar به ما پیغام خطا می‌دهد. از طرفی دیگر، ترتیب اعمال آپشن‌های c، v و z اصلا اهمیتی ندارد و آنها می‌توانند (در این مورد خاص) با هر ترتیبی به این دستور اعمال شوند. اما موقعیت f، بسیار مهم می‌باشد. پس در اینجا اگر می‌خواستیم آپشن‌های این دستور را با - اعمال کنیم، باید دستور را به صورت tar -czvf اعمال می‌کردیم.

از آنجایی که اعمال آپشن‌ها بدون علامت -، چنین دردسرهایی ندارد، در اکثر مواقع، آپشن‌های tar را بدون علامت - اعمال خواهیم کرد. در نهایت، دقت داشته باشید که استفاده یا عدم استفاده از -، تاثیری روی سایر آرگمان‌های اعمالی به دستور tar ندارد. برای مثال ما هنوز باید پس از اعمال آپشن، ابتدا نام فایل آرشیوی که میخواهیم ایجاد کنیم و پس از آن، فایل‌هایی که میخواهیم درون فایل آرشیو قرار گیرند را مشخص کنیم.

حال بیایید نگاهی به حجم این فایل بیاندازیم:

```
[root@localhost ~]# ls -lh
total 8.2G
-rw-r--r--. 1 root root 4.1G Jun 29 12:55 ar-testfile.tar
-rw-r--r--. 1 root root 4.1M Jun 29 13:34 comp-ar-testfile.tar
drwxr-xr-x. 2 root root   73 Jun 29 13:11 Extract
-rw-r--r--. 1 root root 2.0G Jun 29 12:49 testfile
-rw-r--r--. 1 root root 1.0G Jun 29 10:27 testfile2
-rw-r--r--. 1 root root 1.0G Jun 29 10:29 testfile3
-rw-r--r--. 1 root root 100M Jun 29 12:44 testfile4
```

همانطور که می‌بینید این فایل اکنون ٤,١ مبی‌بایت حجم دارد. یعنی ما موفق شدیم با آرشیو و فشرده‌سازی کردن، ٤ فایل که در مجموع ٤,١ گیبی‌بایت حجم داشتند را در قالب یک فایل ٤,١ مبی‌بایتی داشته باشیم.

نکته‌ی جالب این است که ما می‌توانیم عمل فشرده‌سازی را تا چند مرحله انجام دهیم. یعنی:

```
[root@localhost ~]# tar cvfz comp-ar-testfile-2.tar comp-ar-testfile.tar
comp-ar-testfile.tar
```

در اینجا، ما همان آپشن‌های قبلی را به tar دادیم، سپس نام فایل آرشیو جدید را مشخص کردیم و سپس به tar گفتیم که تاربال قبلی را، بار دیگر فشرده‌سازی کرده و درون یک تاربال جدید قرار دهد. بیایید نگاهی به حجم این تاربال جدید بیاندازیم:

```
[root@localhost ~]# ls -lh
total 8.2G
-rw-r--r--. 1 root root 4.1G Jun 29 12:55 ar-testfile.tar
-rw-r--r--. 1 root root  12K Jun 29 13:40 comp-ar-testfile-2.tar
-rw-r--r--. 1 root root 4.1M Jun 29 13:34 comp-ar-testfile.tar
...
```

همانطور که می‌بینید، تاربال جدید اکنون فقط ١٢ کیبی‌بایت حجم دارد. ما می‌توانیم بار دیگر این فایل جدید را توسط tar و gzip فشرده‌سازی کنیم. یعنی:

```
[root@localhost ~]# tar cvfz comp-ar-testfile-3.tar comp-ar-testfile-2.tar
comp-ar-testfile-2.tar
```

حال بیایید نگاهی به حجم این فایل جدید بیاندازیم:

```
[root@localhost ~]# ls -lh
total 8.2G
-rw-r--r--. 1 root root 4.1G Jun 29 12:55 ar-testfile.tar
-rw-r--r--. 1 root root  12K Jun 29 13:40 comp-ar-testfile-2.tar
-rw-r--r--. 1 root root 1.6K Jun 29 13:42 comp-ar-testfile-3.tar
-rw-r--r--. 1 root root 4.1M Jun 29 13:34 comp-ar-testfile.tar
...
```

همانطور که می‌بینید، اکنون این فایل، ١,۶ کیبی‌بایت حجم دارد. ما می‌توانیم همینطور تا چند مرحله عملیات آرشیو و فشرده‌سازی را انجام دهیم، اما از جایی به بعد، حجم فایل به جای کاهش، افزایش پیدا می‌کند. برای این که مقایسه‌ای بین روش‌های فشرده‌سازی و میزان فشرده‌سازی در هر مرحله را ببینید، به این لینک مراجعه کنید.

لازم است بار دیگر بگوییم که دلیل این که ما توانستیم به این حد از فشرده‌سازی برسیم، این است که فایل‌های ما، ساختگی می‌باشند و در واقع یک سری فایل هستند که در آنها تعداد زیادی صفر نوشته شده است. در محیط کاری و برای فایل‌های واقعی، رسیدن به چنین حدی از فشرده‌سازی دشوار می‌باشد.

**نکته:** مسئله‌ی مهمی که باید در مورد روش‌های فشرده‌سازی و انتخاب آنها بدانید، این است که ما باید با توجه به نوع فایل، همیشه سعی کنیم بین میزان فشرده‌سازی و زمان طی شده جهت فشرده‌سازی، تعادل را حفظ کنیم. برای مثال، اگر روشی در عرض ۱۰ ثانیه بتواند یک فایل ۱ گیگ را به یک فایل ۱۰ مگ تقلیل دهد و در مقابل، اگر یک روش در عرض ۱ دقیقه فایل ۱ گیگ را به ۷ مگ تقلیل دهد، معمولا ترجیح ما، استفاده از روش ۱۰ ثانیه‌ای می‌باشد. علاوه بر این، ما باید زمان سپری شده جهت اکتسرکت را نیز در نظر بگیریم. همچنین، هنگام استفاده از روش‌های جدیدتر، باید سازگاری آن روش با سیستم‌های قدیمی‌تر را نیز در نظر بگیریم. یکی از دلایلی که هنوز gzip از معروفیت بالایی برخوردار است این است که gzip در همه‌ی سیستم‌ها وجود دارد.

اگر بخواهیم این فایل‌ها را از حالت فشرده در آوریم، کافی است آپشن x را جایگزین آپشن c کنیم. یعنی:

```
[root@localhost ~]# tar xvfz comp-ar-testfile.tar -C New_Extract/
testfile
testfile2
testfile3
testfile4
```