به نام خدا

نگارنده: سینا آقاسی

دانشگاه شهید بهشتی
دانشکده مهندسی برق و کامپیوتر
آزمایشگاه کامپیوتر
پاییز ۱۳۹۲

لینوکس پیشرفته

Shell چیست؟

Shell و ترمینال

- Shell یک واسط کاربر است که امکان دسترسی به سرویس‌های هسته سیستم عامل را فراهم می‌کند.
- در واقع برای انجام یک درخواست در کامپیوتر، مثلاً کپی یک فایل، لازم است مجموعه‌ای از تغییرات و کارها انجام گیرد. از آنجایی انجام همه این کارها توسط کاربر دشوار است، این وظیفه به سیستم عامل محول شده است. اما برای آنکه کاربر بتواند با سیستم عامل تعامل داشته باشد و درخواست‌های خود را به آن بدهد، نیاز به واسطه‌ای می‌باشد. Shell همین واسطه است.
- به تعبیر دیگر می‌توان معنای Shell یعنی «پوسته» را در نظر گرفت. Shell همانند یک پوسته، جزئیات انجام گرفته برای اجرای درخواست کاربر توسط سیستم عامل را از دید کاربر مخفی می‌کند.

انواع Shell

- Shell بر دو نوع است:
  - واسط گرافیکی کاربر (Graphical User Interface = GUI)
  - واسط خط فرمان (Command Line Interface = CLI)
- «واسط گرافیکی کاربر» همان واسطی است که در یک محیط گرافیکی با استفاده از صفحه‌کلید، ماوس و نمایشگر امکان کار را فراهم می‌آورد.
- Gnome، KDE و دیگر واسط‌های گرافیکی که در لینوکس مقدماتی نام برده شد، نمونه‌هایی از Shell با واسط گرافیکی کاربر هستند.
- «واسط خط فرمان» نوعی از واسط است که تنها به صورت متنی (Text) تعامل با سیستم عامل را ممکن می‌سازد. در این نوع Shell، کلیک ماوس نقشی ندارد.
- bash، sh، tcsh نمونه‌هایی از Shell با واسط خط فرمان هستند. bash قدرتمندترین Shell با واسط خط فرمان است. طبق ساختار دایرکتوری‌های لینوکس که در لینوکس مقدماتی توضیح داده شد، توقع داریم این Shellها در /bin/ قرار داشته باشند.

ترمینال چیست؟

Shell و ترمینال

- «ترمینال» یک دستگاه سخت‌افزاری است که برای وارد کردن داده به کامپیوتر یا نمایش اطلاعات کامپیوتر استفاده می‌شود.
- ترمینال‌ها جنبه تاریخی دارند و امروزه استفاده نمی‌شوند.
- «مقلد ترمینال» یا Terminal Emulator یک برنامه است که ترمینال را درون یک معماری نمایش دیگر، تقلید می‌کند.
- Xterm، Gnome-Terminal نمونه‌های معروف مقلد ترمینال هستند.
- مقلد ترمینال پیش‌فرض در ابونتو، Gnome-Terminal است که دارای امکانات خوبی است.

```
[root@localhost var]# ls -la
total 72
drwxr-xr-x. 18 root root 4096 Jul 30 22:43 .
drwxr-xr-x. 23 root root 4096 Sep 14 20:42 ..
drwxr-xr-x.  2 root root 4096 May 14 00:15 account
drwxr-xr-x. 11 root root 4096 Jul 31 22:26 cache
drwxr-xr-x.  3 root root 4096 May 18 16:03 db
```

قدرت خط فرمان

- سیستم‌های مبتنی بر یونیکس مثل لینوکس در ابتدا فقط دارای واسط‌های خط فرمان بودند؛ لذا خط فرمان آنها بر خلاف ویندوز به شدت قوی است.
- به طور کلی هر کاری را با خط فرمان لینوکس می‌توان انجام داد. تمام کارهای مربوط به سیستم عامل لینوکس و بسیاری از ابزارهای مخصوص لینوکس توسط خط فرمان قابل انجام است.
- برنامه‌نویسان یکی از طرفداران اصلی خط فرمان لینوکس هستند چون خیلی از کارها توسط خط فرمان قابل انجام است. همچنین خروجی خط فرمان به راحتی قابل استفاده در زبان‌های برنامه‌نویسی است.
- در ادامه به طور پیش‌فرض از Gnome-Terminal به عنوان مقلد ترمینال و از Bash به عنوان Shell پیش‌فرض استفاده می‌کنیم.

اعلان آمادگی

- Bash برای اعلان آمادگی دریافت دستور از قالب زیر تبعیت می‌کند:

```
User@Computer:WD$
```

  - User: نام کاربری که در آن قرار دارد را مشخص می‌کند.
  - Computer: نام کامپیوتری که در آن قرار دارد را مشخص می‌کند.
  - WD: دایرکتوری کاری یا دایرکتوری جاری را مشخص می‌کند. بعداً در این مورد صحبت خواهیم کرد.
  - $ یا # سطح دسترسی را مشخص می‌کند. # به معنی کاربر با بیشترین سطح دسترسی است. در غیر این صورت از $ استفاده می‌شود.
- مشاهده قالب بالا به معنی آن است که Bash منتظر است که کاربر دستور را از طریق صفحه کلید وارد کند.

تاریخچه

- از قابلیت‌های Bash نگهداری لیست دستورهای قبلی است که وارد شده است.
- برای دسترسی به دستورهای قبلی می‌توان از کلیدهای مکان‌یابی بالا و پایین استفاده کرد. کلید بالا، دستور قبل را پدیدار می‌کند در حالیکه کلید پایین، دستور بعد را.
- این قابلیت باعث سرعت بخشیدن به کار با خط فرمان می‌شود.
- دستور history با قالب زیر لیست دستورهای وارد شده تاکنون را نشان می‌دهد.

```
$ history
```

دایرکتوری کاری

- هر ترمینال در هر لحظه تنها در یک دایرکتوری می‌تواند قرار داشته باشد. این دایرکتوری را «دایرکتوری کاری» یا Working Directory می‌نامیم.
- در برخی منابع به آن «دایرکتوری جاری» یا Current Directory نیز گویند.
- در واقع یک ترمینال مانند یک پنجره مدیریت فایل در حالت گرافیکی است. همانطور که پنجره ابزار مدیریت فایل، در هر لحظه در یک دایرکتوری قرار دارد، ترمینال نیز در هر لحظه در یک دایرکتوری قرار دارد.

انواع آدرس

- همانطور که در لینوکس مقدماتی دیدیم، در لینوکس همه چیز از / شروع می‌شود. حتی دستگاه‌های مختلفی که به کامپیوتر متصل می‌شوند در یکی از زیر شاخه‌های / قرار می‌گیرند.
- آدرس‌ها به دو دسته مطلق و نسبی تقسیم می‌شود.
- «آدرس مطلق» یعنی آدرس یک فایل یا پوشه نسبت به / به طور کامل بیان شود. برای مثال آدرس مطلق ابزار gedit برابر با /usr/bin/gedit است.
- «آدرس نسبی» یعنی آدرس یک فایل یا پوشه نسبت به **دایرکتوری کاری** بیان شود. برای مثال اگر دایرکتوری کاری /usr/ باشد، آدرس نسبی gedit برابر با bin/gedit است و اگر دایرکتوری کاری /usr/bin باشد، آدرس نسبی این ابزار gedit است.
- واضح است که شروع آدرس با / به معنی آدرس مطلق است.
- اکثر ابزارها هر دو آدرس مطلق و نسبی را قبول می‌کنند.

نمادهای خاص آدرس

- در آدرس‌دهی برخی نمادها وجود دارند که معنی خاصی دارند. در واقع فرم کوتاه شده هستند:
  - . : به معنی دایرکتوری کاری است. از نوع آدرس آن مطلق است.
  - .. : دایرکتوری قبل از دایرکتوری کاری. یعنی همان دایرکتوری که دایرکتوری کاری در آن واقع شده است. از نوع آدرس نسبی است.
  - ~: دایرکتوری Home کاربر جاری. برای کاربر root این مقدار برابر با /root و برای سایر کاربران برابر با /home/Username است. از نوع آدرس مطلق است.

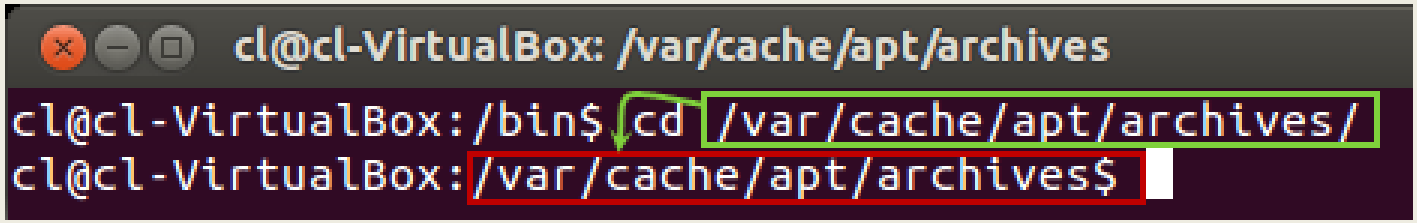
تغییر دایرکتوری کاری

- دستور cd برای تغییر دایرکتوری کاری استفاده می‌شود. قالب دستور:

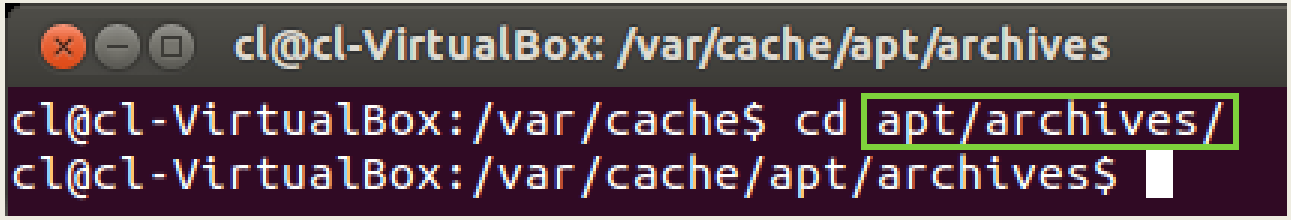
```
$ cd DIR
```

که DIR دایرکتوری مقصد را مشخص می‌کند.

- نمونه آدرس مطلق:



- نمونه آدرس نسبی:

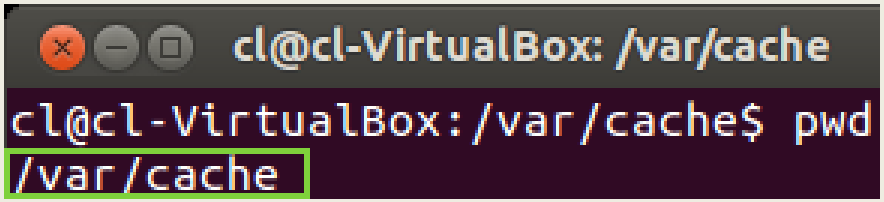


نمایش دایرکتوری کاری

- برای نمایش دایرکتوری کاری از pwd استفاده می‌شود. قالب دستور:

$ pwd

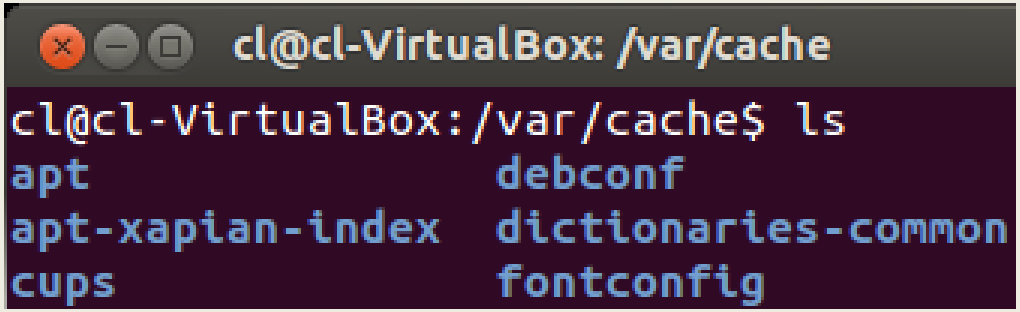
که DIR دایرکتوری مقصد را مشخص می‌کند.



نمایش لیست فایل‌ها و پوشه‌ها

- دستور ls برای نمایش فایل‌ها و پوشه‌های موجود در دایرکتوری کاری استفاده می‌شود. قالب دستور به صورت زیر است.

```
$ ls
```



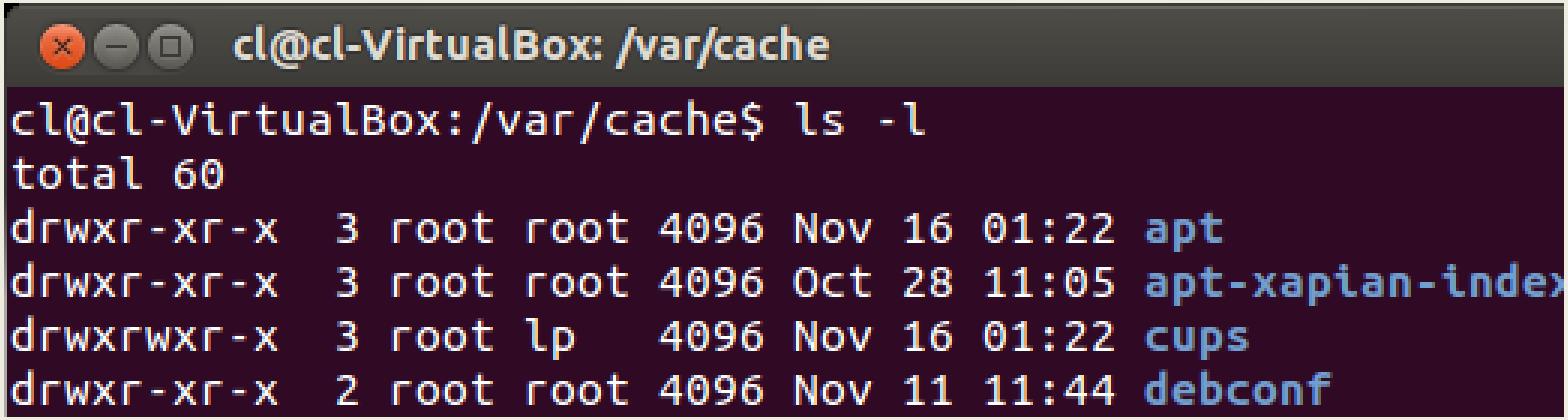
- به صورت پیش‌فرض فایل‌ها و پوشه‌ها مخفی را نشان نمی‌دهد. برای این کار از سوئیچ a- استفاده می‌شود.

```
$ ls -a
```

نمایش لیست فایل‌ها و پوشه‌ها (ادامه)

- اگر کلیه اطلاعات مربوط به محتویات دایرکتوری جاری را بخواهیم باید از سوئیچ l– استفاده کنیم:

```
$ ls -l
```



- ستون پنجم اندازه بر حسب بایت را بیان می‌کند که برای انسان به راحتی قابل خواندن نیست. برای نمایش اندازه فایل یا پوشه بر حسب واحدهای قابل درک برای انسان از دستور زیر استفاده می‌شود:

```
$ ls -lh
```

نمایش اندازه محتویات پوشه

- ls اندازه همه پوشه‌ها را ۴۰۹۶ بایت نمایش می‌دهد و به اندازه فایل‌های درون آن اهمیت نمی‌دهد.
- برای نمایش اندازه محتویات پوشه از du با قالب زیر استفاده می‌شود.

```
$ du DIR
```

DIR آدرس دایرکتوری مورد نظر است. اگر ذکر نشود، دایرکتوری کاری در نظر گرفته می‌شود.

- این دستور به طور پیش‌فرض اندازه همه زیر پوشه‌های موجود در مسیر تعیین شده را نشان می‌دهد. برای جلوگیری از این کار از دستور زیر استفاده می‌شود.

```
$ du -s DIR
```

- برای نمایش اندازه‌ها بر حسب واحدهای قابل خواندن توسط انسان سوئیچ h– استفاده می‌شود.

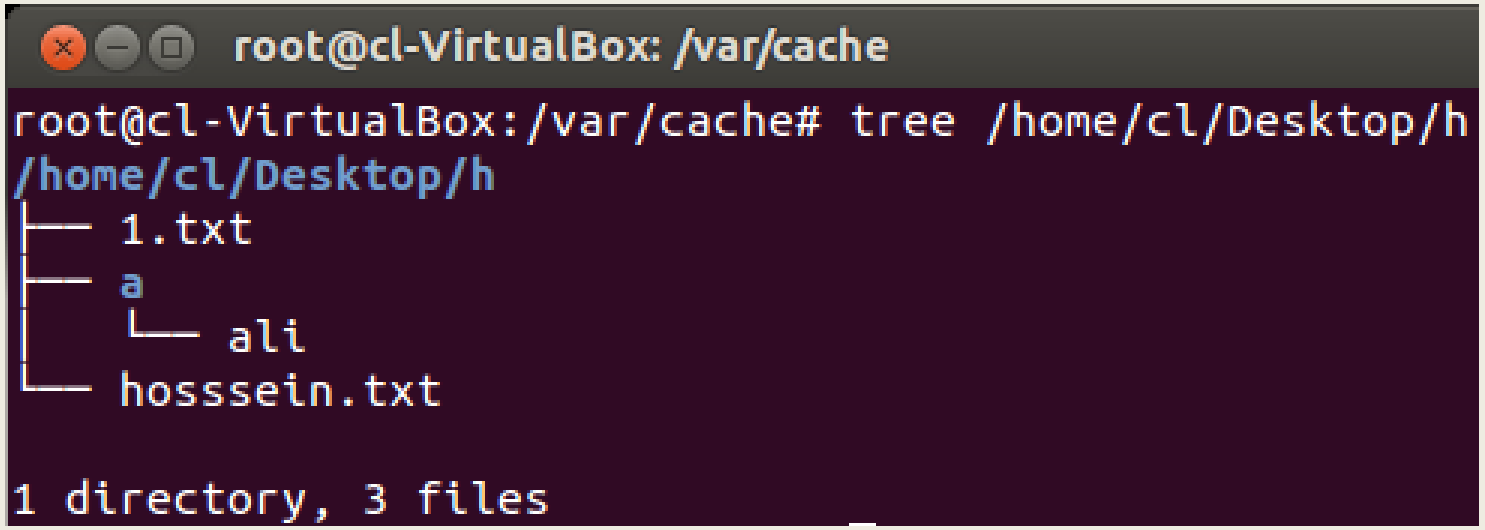
```
$ du -h DIR
```

نمایش ساختار درختی

- ابزاری به نام tree وجود دارد که یک ساختار درختی از مسیر تعیین شده در خط ترمینال نمایش می‌دهد. اینکه فایل یا پوشه‌ای درون پوشه دیگری واقع است را با تو رفتگی نمایش می‌دهد. قالب دستور به صورت زیر است:

```
$ tree DIR
```

DIR آدرس دایرکتوری مورد نظر است. اگر ذکر نشود، دایرکتوری کاری در نظر گرفته می‌شود.



ایجاد و حذف نام فایل

- برای ایجاد فایل از هر ابزار ویرایشگر فایل می‌توان استفاده کرد.
- اگر هدف فقط ساختن یک فایل باشد، از دستور زیر استفاده می‌شود:

```
$ touch NAMe
```

NAME نام فایل مورد نظر است.

- برای حذف فایل از rm به شکل زیر استفاده می‌شود:

```
$ rm NAME
```

جابه‌جایی و تغییر نام فایل

- برای انتقال یک فایل از یک دایرکتوری به دایرکتوری دیگر همچنین تغییر نام آن از mv استفاده می‌شود. قالب دستور به صورت زیر است:

```
$ mv SRC DST
```

SRC آدرس فایل مبدا است و DST آدرس مقصد است.

- اینکه تغییر نام انجام گیرد یا جابه‌جایی به مقدار آدرس مقصد بستگی دارد. ممکن است هر دو با هم انجام گیرند.
  - اگر آدرس مقصد به / ختم شود، یعنی آدرس مقصد یک دایرکتوری را نشان می‌دهد؛ لذا جابه‌جایی انجام می‌گیرد و تغییر نام نداریم.
  - وگرنه نام انتهایی به عنوان نام جدید فایل در نظر گرفته می‌شود. اگر آدرس مقصد دایرکتوری جدیدی را معرفی کند، جابه‌جایی نیز انجام می‌گیرد.

جابه‌جایی و تغییر نام فایل (ادامه)

- فرض کنید درون /home/test/ قرار داریم و فایل old.txt درون این پوشه قرار گرفته است. برای تغییر نام این فایل به new.txt از دستور زیر استفاده می‌شود:

```
$ mv old.txt new.txt
```

- فرض کنید می‌خواهیم همان فایل را به /tmp/test/ منتقل کنیم بدون اینکه نام فایل تغییر یابد. برای این کار از دستور زیر استفاده می‌کنیم:

```
$ mv old.txt /tmp/test/
```

- اگر بخواهیم همین فایل را علاوه بر انتقال به /tmp/test/، به new.txt نیز تغییر دهیم، از دستور زیر استفاده می‌کنیم:

```
$ mv old.txt /tmp/test/new.txt
```

ایجاد و حذف پوشه

- برای ایجاد پوشه از mkdir به شکل زیر استفاده می‌شود:

```
$ mkdir NAME
```

NAME نام پوشه مورد نظر است.

- برای حذف پوشه خالی از rmdir به صورت زیر استفاده می‌شود.

```
$ rmdir NAME
```

- اگر پوشه خالی نباشد، نمی‌توان با rmdir آن را پاک کرد. در این صورت از دستور rm با سوئیچ r– استفاده می‌کنیم. این سوئیچ که برگرفته از Recursive است به این معنی است که تمام فایل‌ها و پوشه‌های داخل پوشه تعیین شده با همراه خود آن را پاک شود. برای مثال دستور زیر پوشه sample و تمام محتویات آنرا پاک می‌کند:

```
$ rm -r sample
```

جابه‌جایی و تغییر نام پوشه

- همانند فایل برای انتقال یک پوشه (با تمام محتویات آن) یا تغییر نام آن نیز از mv به صورت زیر استفاده می‌شود.

```
$ mv SRC DST
```

- اینکه تغییر نام انجام گیرد یا جابه‌جایی به مقدار آدرس مقصد بستگی دارد. ممکن است هر دو با هم انجام گیرند.
  - اگر آدرس مقصد به / ختم شود، یعنی آدرس مقصد یک دایرکتوری را نشان می‌دهد؛ لذا جابه‌جایی انجام می‌گیرد و تغییر نام نداریم.
  - اگر آدرس مقصد مسیر پوشه‌ای که وجود دارد را نشان دهد، پوشه مبدا به داخل پوشه مقصد منتقل می‌شود.
  - وگرنه نام انتهایی به عنوان نام جدید پوشه در نظر گرفته می‌شود. اگر آدرس مقصد دایرکتوری جدیدی را معرفی کند، جابه‌جایی نیز انجام می‌گیرد.

جابه‌جایی و تغییر نام فایل (ادامه)

- فرض کنید درون /home/test/ قرار داریم و پوشه old درون این مسیر قرار گرفته است. برای تغییر نام این پوشه به new از دستور زیر استفاده می‌شود:

```
$ mv old new
```

- فرض کنید می‌خواهیم همان پوشه را به /tmp/test/ منتقل کنیم بدون اینکه نام آن تغییر یابد. برای این کار از دستور زیر استفاده می‌کنیم:

```
$ mv old /tmp/test/
```

- اگر بخواهیم همین پوشه را علاوه بر انتقال به /tmp/test/، به new نیز تغییر دهیم، از دستور زیر استفاده می‌کنیم:

```
$ mv old /tmp/test/new
```

البته فرض کردم که پوشه new از قبل وجود ندارد، وگرنه پوشه old به داخل پوشه new منتقل می‌شد.

کپی فایل

- دستور cp با قالب زیر برای کپی فایل SRC به DST انجام می‌گیرد:

```
$ cp SRC DST
```

- این دستور می‌تواند نام جدیدی برای فایل مقصد نیز تعیین کند. کافی است DST به / ختم نشود (یعنی DST آدرس فایل باشد نه آدرس پوشه).
- برای مثال دستور زیر فایل name.txt را در مسیر /home/test/ کپی می‌کند. یعنی پس از اجرای دستور زیر، در مسیر /home/test/ فایل name.txt قرار دارد.

```
$ mv name.txt /home/test/
```

- اما در دستور زیر علاوه بر کپی کردن فایل، نام فایل جدید را new.txt قرار می‌دهد. یعنی پس از اجرای دستور زیر، در مسیر /home/test/ فایل new.txt قرار خواهد داشت.

```
$ cp name.txt /home/test/new.txt
```

کپی پوشه

- باید در نظر داشت که پوشه حاوی چند فایل یا پوشه می‌تواند باشد؛ لذا نباید انتظار داشته باشیم که دستور cp به همان شکل برای کپی پوشه نیز استفاده شود.
- برای کپی پوشه از دستور cp با سوئیچ r– به معنی انجام کپی به صورت بازگشتی استفاده می‌شود.

```
$ cp -r SRC DST
```

- قوانینی که در مورد کپی فایل صادق بود، در اینجا نیز صادق است.
- در کپی پوشه به شیوه گفته شده، خواص پوشه مثل سطح دسترسی‌ها حفظ نمی‌شود. برای یکسان ماندن این سطح دسترسی‌ها از سوئیچ a– به معنی حفظ خاصیت‌ها به صورت زیر استفاده می‌کنیم(چون می‌خواهیم پوشه کپی کنیم، وجود r– بدیهی است):

```
$ cp -ra SRC DST
```

نمایش محتویات فایل متنی

- برای نمایش محتویات یک فایل متنی از هر ابزار ویرایشگر متن می‌توان استفاده کرد. اما گاهی اوقات استفاده از ابزارهای ویرایشگر متن با آن همه امکانات ضروری نیست. بدین منظور از ابزارهای نمایشگر متن ساده استفاده می‌شود.
- سه ابزار cat، more و less با قالب یکسان به صورت زیر قابل استفاده هستند.

```
$ cat FILE
```

- cat محتویات را در خط فرمان نمایش داده و سریعاً به خط فرمان باز می‌گردد.
- less و more یک محیط ساده برای نمایش را فراهم می‌آورند که قابلیت جابه‌جایی در متن وجود دارد. تفاوت less و more در آن است که اولی امکان هر جابه‌جایی در متن را فراهم آورده در حالیکه دومی فقط امکان پایین رفتن دارد.

Symbolic Link

- Symbolic Link یا Symlink یا Softlink نوعی فایل است که به فایل دیگر اشاره می‌کند.
- هرگونه عملی بر روی Symlink روی جایی که به آن اشاره می‌کند انجام می‌گیرد.
- Symlink با دستور زیر ساخته می‌شود:

```
$ Ln -s SRC TRG
```

- SRC فایل یا پوشه‌ای که می‌خواهیم از آن Symlink بسازیم را مشخص می‌کند و TRG نام فایل جدیدی که قرار است به SRC اشاره کند را مشخص می‌سازد.

مثال Symlink از فایل

- gedit درون مسیر /usr/bin/gedit قرار دارد. می‌خواهیم در دایرکتوری کاری لینکی به این ویرایشگر بسازیم و نام آن را MyFavEditor قرار دهیم، بدین منظور از دستور زیر استفاده می‌کنیم:

```
$ ln -s /usr/bin/gedit MyFavEditor
```

اجرای MyFavEditor با دستور زیر سبب اجرای gedit خواهد شد.

```
$ ./MyFavEditor
```

- در مثال دیگر می‌خواهیم یک Symlink از /etc/passwd بسازیم تا برای دسترسی‌های بعدی لازم نباشد آدرس کامل آن را وارد کنید. بدین منظور از دستور زیر استفاده می‌کنیم:

```
$ ln -s /etc/passwd users
```

مشاهده محتویات با دستور زیر سبب مشاهده محتویات passwd خواهد شد:

```
$ cat users
```

مثال Symlink از پوشه

- مدیر سیستم می‌خواهد به طور مکرر به Desktop کاربر ali دست یابد. می‌دانیم Desktop این کاربر در مسیر زیر قرار دارد.

```
/home/ali/Desktop
```

مدیر سیستم برای راحتی کار یک Symlink با دستور زیر تولید می‌کند:

```
# ln -s /home/ali/Desktop AliDesktop
```

اکنون برای ورود به آن از دستور زیر استفاده می‌کند:

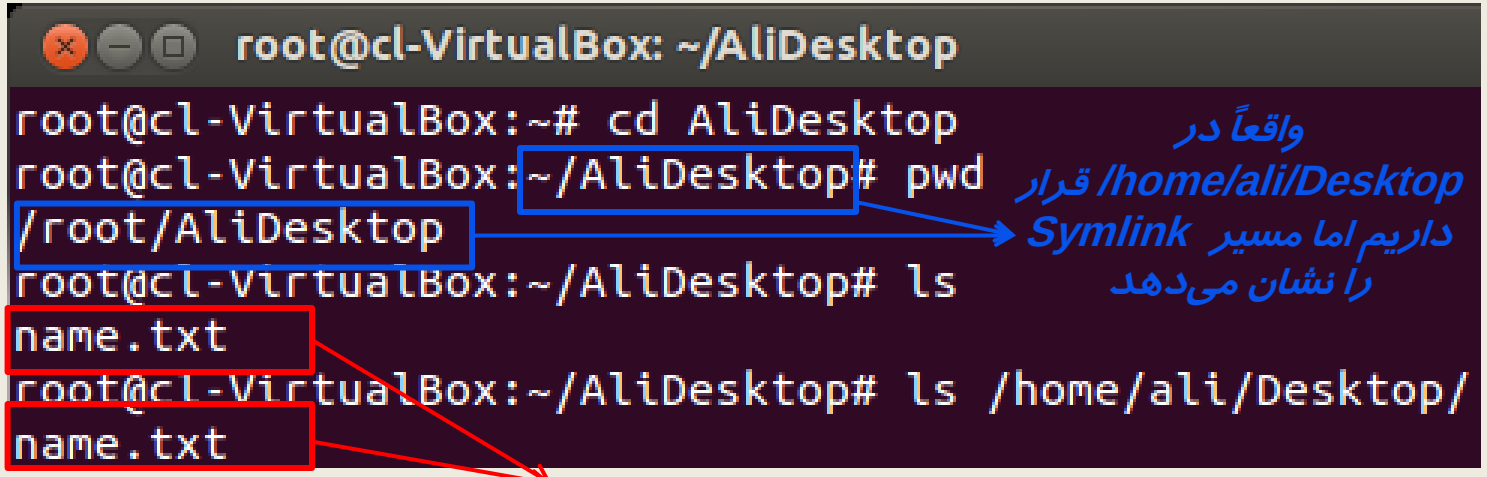
```
# cd AliDesktop
```

حال اگر این مدیر بخواهد فایلِ name.txt را به Desktop علی منتقل کند، دو راه زیر را خواهد داشت (مسلماً راه اول بهتر است زیرا کوتاهتر است):

```
# mv name.txt AliDesktop/
# mv name.txt /home/ali/Desktop
```

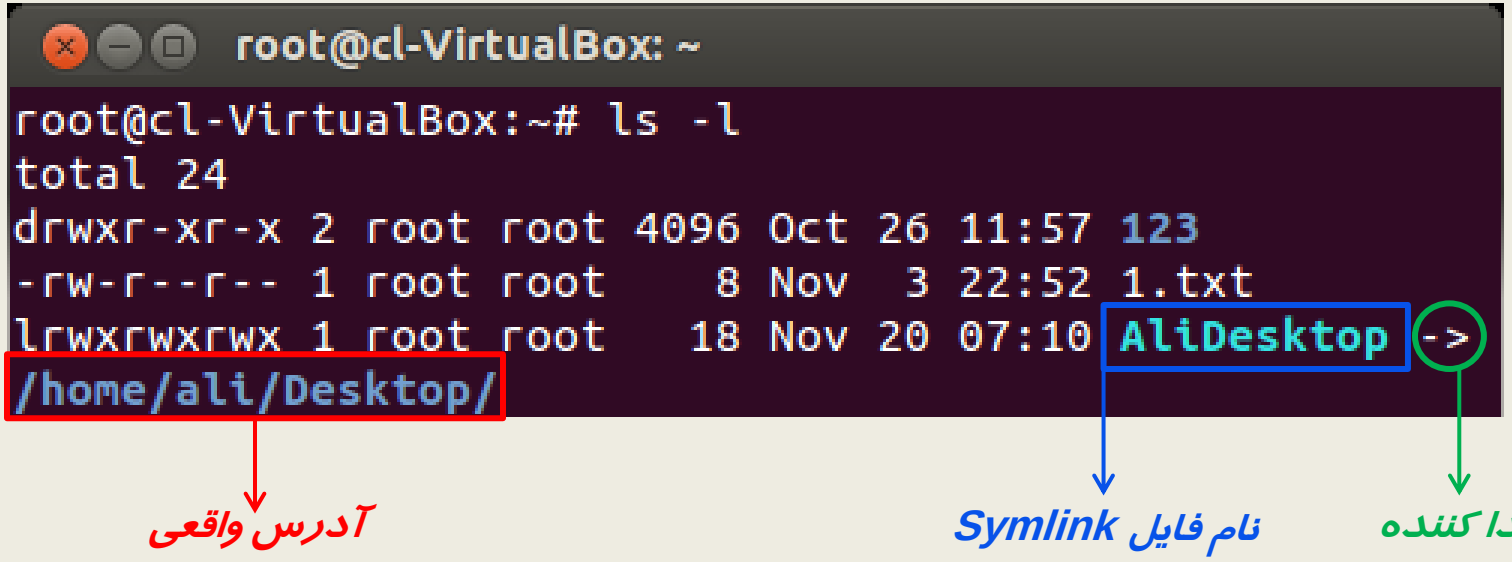
دایرکتوری کاری در Symlink

- اگر به مثال قبل دقت کرده باشید وقتی از دستور cd برای ورود به دایرکتوری استفاده کردیم، دایرکتوری کاری نام Symlink نشان داده می‌شد در حالیکه اگر درون آن ls بگیرید، لیست فایل‌های درون Desktop علی دیده می‌شود.
- دلیل این امر استفاده از Symlink است.
- حتی pwd نیز مسیر واقعی را نشان نمی‌دهد.



تشخیص مسیر واقعی Symlink

- برای مشاهده مسیر واقعی Symlink باید از دستور ls با سوئیچ l– استفاده کنیم.
- مسیر واقعی پس از علامت -> ظاهر می‌شود.
- حتی pwd نیز مسیر واقعی را نشان نمی‌دهد.



چند کاربرد Symlink

- یکی از کاربردهای Symlink کوتاه کردن آدرس‌ها است. این کاربرد را در اسلایدهای قبل دیدیم.
- کاربرد دیگر یکپارچه‌سازی در آدرس‌ها است. فرض کنید در شرکتی کامپیوترها دارای هارد دیسک یکسان نباشند لذا ممکن است پارتیشن‌های مختلفی وجود داشته باشد و فایل‌ها در یک سری از کامپیوترها در پارتیشن دوم باشد در حالیکه در برخی کامپیوترها در پارتیشن سوم باشد. این کار سبب می‌شود که آدرس فایل‌ها متفاوت باشد. یعنی مدیر سیستم‌های آن شرکت باید بداند که در چه کامپیوتری، فایل‌ها در کجا ذخیره شده‌اند. این مشکل زمانی خود را نشان می‌دهد که کاربران دائماً از مدیر سیستم‌ها در مورد مشکلات خود کمک می‌خواهند. مدیری سیستم‌ها می‌تواند با تعریف یک Symlink و دخیل کردن آن در آدرس‌ها، کار خود را ساده سازد. کافی است یکبار Symlink را به مسیر مناسب تنظیم کرده و از دفعات بعد فقط با Symlink کار کند.

چند کاربرد Symlink (ادامه)

- کاربرد دیگر انتقال مسیر است. برای مثال برنامه‌ای بسیار مهم تنها تنظیم شده که فایل را از مسیر /etc/sample.txt بخواند؛ اما ما می‌خواهیم در کار با این برنامه فایل را دائماً عوض کنیم. یک راه آن است که فایل خود را هر بار در مسیر /etc کپی کرده و نام آن را به sample.txt تغییر دهیم. اما این کار برای فایل‌های بزرگ مناسب نیست. یا حتی برای زمانیکه تعداد دفعاتی که این کار را انجام می‌دهیم زیاد باشد. نمی‌توانیم تشخیص دهیم در هر لحظه کدام فایل جای sample.txt قرار گرفته است. بهترین راه حل تعریف یک Symlink به نام sample.txt است که به فایل ما اشاره کند. دیگر نیاز به کپی‌ها و تغییر نام‌های خسته کننده نیست. در ضمن هر لحظه بخواهیم می‌توانیم با ls –l مطمئن شویم Symlink به کدامین فایل اشاره می‌کند.

uid و gid

- هر کاربر در لینوکس دارای یک شماره شناسایی منحصر به فرد در همان سیستم عامل است. در واقع هر نام کاربری دارای یک شماره شناسایی منحصر به فرد است. به این شماره شناسایی، uid گویند.
- برای دسته‌بندی کاربران و سطح دسترسی آن‌ها گروه‌ها ساخته می‌شوند. گروه‌ها برای کارهای متفاوت با سطوح دسترسی متفاوت هستند. هر گروه علاوه بر نام منحصر به فرد، دارای شماره شناسایی منحصر به فرد به نام gid هستند.
- هر کاربر در هر لحظه می‌تواند عضو چند گروه باشد، اما همواره یکی از آنها به عنوان گروه اصلی شناخته می‌شود و سایر گروه‌ها فرعی هستند. اینکه کدام گروه اصلی باشد، توسط مدیر سیستم تعیین می‌شود.
- اینکه نام کاربر با نام گروه برابر باشد مشکلی ایجاد نمی‌کند بلکه مهم آن است که نام یک کاربر با نام کاربر دیگر برابر نباشد. این قانون در مورد uid و gid نیز وجود دارد.

بدست آوردن اطلاعات کاربر

- دستور زیر نام کاربر فعلی ترمینال را مشخص می‌کند:

```
$ whoami
```

- برای بدست آوردن uid و gid کاربر فعلی ترمینال از دستور زیر استفاده می‌شود:

```
$ id
```

- خروجی چیزی شبیه به خط زیر خواهد بود:

```
uid=1000(cl) gid=1000(cl) groups=1000(cl),4(adm),…
```

- عدد مقابل uid یعنی ۱۰۰۰ بیان می‌کند که برای کاربر cl، uid برابر با ۱۰۰۰ است. عدد مقابل gid نیز gid گروه اصلی کاربر را مشخص می‌کند. groups تمامی گروه‌هایی که این کاربر عضو آن‌ها است را مشخص می‌کند.
- برای مشاهده اطلاعات کاربر USR از دستور زیر استفاده می‌کنیم:

```
$ id USER
```

انواع کاربر

- در ابونتو سه دسته کاربر وجود دارد:
  - Root : دارای سطح دسترسی کامل است و می‌تواند هر کاری انجام دهد.
  - Standard : دارای کمترین سطح دسترسی است. می‌تواند بسیاری از برنامه‌ها را اجرا کند یا بسیاری از فایل را مشاهده کند؛ اما امکان تغییر بسیاری از فایل‌ها یا اجرای برنامه‌های حساس را ندارد.
  - Administrator : در حالت معمولی دارای دسترسی همانند کاربر نوع Standard است اما می‌تواند در صورت لزوم سطح دسترسی خود را تا Root بالا بکشد و سپس در صورت لزوم به حالت قبل برگردد.
- به طور پیش‌فرض کاربری با نام root و uid برابر صفر در ابونتو وجود دارد که از دسته Root می‌باشد.
- کاربری که در هنگام نصب ابونتو ساخته می‌شود در دسته Administrator قرار می‌گیرد.
- کاربرانی که بعداً ساخته می‌شوند به صورت پیش‌فرض در دسته Standard قرار دارند.

Sudoer

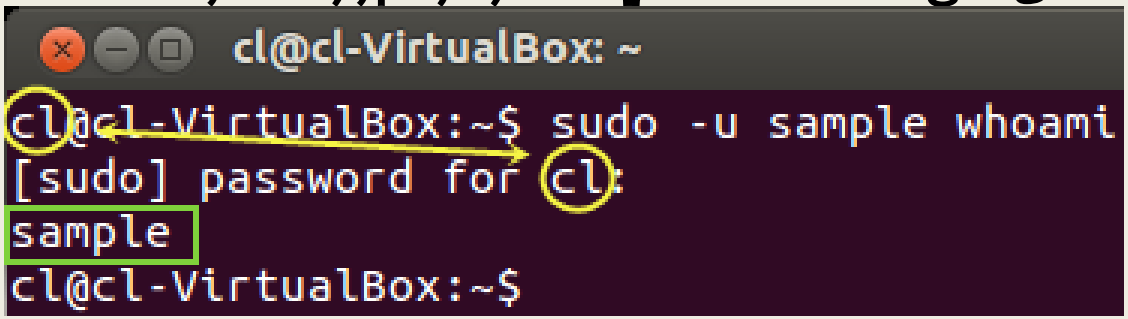
- کاربری که اجازه اجرای دستورها به نیابت از دیگر کاربران را داشته باشد، Sudoer است.
- در واقع کاربر Sudoer می‌تواند هر دستوری را از طرف کاربر دیگری اجرا کند به شرط آنکه کاربر مورد نظر دسترسی انجام آن کار را داشته باشد.
- نوع Sudoer همان نوع Administrator است.
- یکی از کاربردهای نوع Sudoer، اجرای دستورها به نیابت از کاربر root است. در این صورت بدون وارد شدن به کاربر root می‌توان دستورهای لازم را اجرا کرد. این همان بالا کشیدن سطح دسترسی تا سطح Root است.

Sudo

- برای اجرای دستور CMD به نیابت از کاربر USR از دستور زیر استفاده می‌شود:

```
$ sudo -u USR CMD
```

- با اجرای این دستور ابتدا رمز عبور کاربر فعلی خواسته می‌شود.
- شکل زیر نتیجه اجرای دستور whoami به نیابت از کاربر sample توسط کاربر cl را نشان می‌دهد. دقت کنید که رمز عبور cl خواسته شده است.



- اگر نام کاربر ذکر نشود به معنی کاربر root است.

```
$ sudo CMD
```

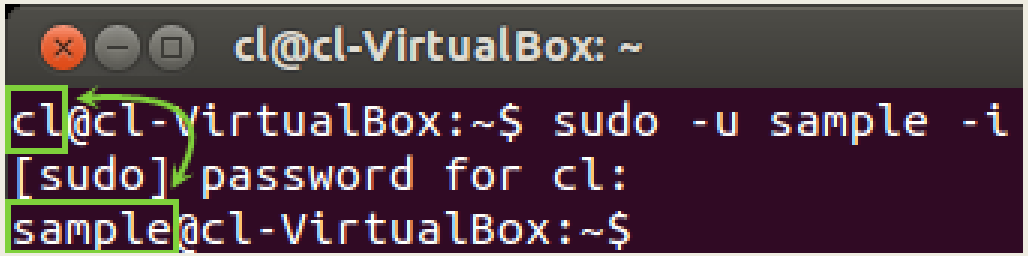
اجرا به نیابت از root

تغییر کاربر با Sudo

- هنگامیکه تعداد دستورهایی که می‌خواهیم به نیابت از کاربر دیگر اجرا کنیم زیاد باشد، استفاده از دستور sudo می‌تواند عذاب‌آور باشد.
- برای رفع مشکل می‌توان کاربر جاری خط فرمان را عوض کرد. بدین منظور یکی از دو دستور زیر استفاده می‌شود:

```
$ sudo -u USR -i
$ sudo -u USR -s
```

- پس از اجرای هر یک از دستورهای خط بالا، خط فرمان به کاربر تعیین شده تغییر کاربری داده و دستورها در کاربر جدید قابل اجرا است. شکل زیر این مورد را نشان می‌دهد:



- اگر نام کاربر تعیین نشود، تغییر به کاربر root انجام می‌گیرد.

Switch User

- گاهی اوقات لازم است خط فرمان به کاربر دیگری تغییر یابد.
- تاکنون با استفاده از sudo این کار را انجام دادیم؛ ولی تنها Sudoer اجازه استفاده از sudo را دارد.
- راه دیگر استفاده از دستور su (مخفف Switch User) است. این دستور خط فرمان را به کاربر مورد نظر تغییر می‌دهد.
- قالب دستور برای تغییر به کاربر USR به صورت زیر است:

```
$ su -l USR
```

- پس از اجرای دستور بالا، رمز عبور کاربر USR خواسته می‌شود.
- اگر نام کاربر تعیین نشود، تغییر به کاربر root انجام می‌گیرد. یعنی دستور زیر برای تغییر کاربر به root است:

```
$ su
```

تفاوت su و sudo

- همانطور که دیدید، برای تغییر کاربر به طور دائم از su و sudo (سوئیچ i– و s–) می‌توان استفاده کرد.
- تفاوت این دو در زیر آمده است:
  - هر کاربری اجازه استفاده از su را دارد در حالیکه sudo را فقط Sudoer می‌تواند استفاده کند.
  - در su رمز کاربر مقصد خواسته می‌شود در حالیکه در sudo رمز کاربر فعلی (که باید Sudoer باشد) خواسته می‌شود.
- در ابونتو دستور sudo بیشتر از su استفاده می‌شود. (این هدف سازندگان ابونتو است که از sudo به جای su استفاده شود)

افزودن و حذف کاربر

- برای افزودن کاربر باید سطح دسترسی Root داشته باشیم.
- افزودن کاربر جدید با نام USR توسط دستور زیر انجام می‌گیرد:

```
# useradd USR
```

- uid توسط سیستم عامل تعیین می‌شود.
- پس از افزودن کاربر، باید رمز عبوری به آن اختصاص یابد وگرنه قابل استفاده نیست.
- همچنین باید پوشه‌ای با نام کاربر با سطح دسترسی مناسب درون /home ساخته شود وگرنه کاربر امکان ورود نخواهد داشت زیرا دایرکتوری Home آن موجود نیست.
- حذف کاربر نیز از طریق دستور زیر قابل انجام است:

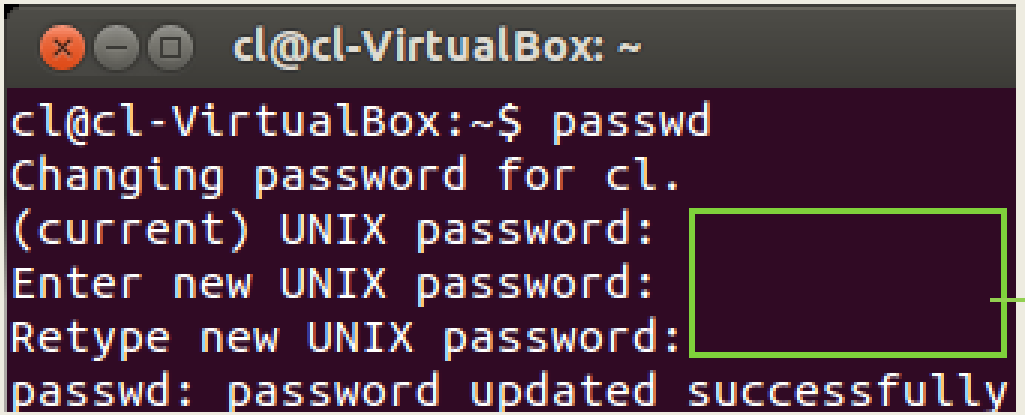
```
# userdel USR
```

تغییر رمز عبور

- برای هرگونه تغییر یا تعیین رمز عبور از passwd استفاده می‌شود.
- دستور زیر رمز عبور کاربر فعلی را عوض می‌کند:

```
$ passwd
```

- پس از اجرای این دستور ابتدا رمز عبور فعلی کاربر خواسته می‌شود و در ادامه از کاربر می‌خواهد که رمز جدید را دوبار وارد کند. دقت کنید که رمز وارد شده اصلاً نمایش داده نمی‌شود (حتی به صورت ستاره یا کاراکترهای دیگر)



تغییر رمز عبور کاربر دیگر

- برای تغییر رمز عبور کاربر دیگر نیاز به دسترسی Root است.
- نام کاربر باید در دستور passwd ذکر شود:

```
# passwd USR
```

- برخلاف حالت قبل، رمز عبور قبلی خواسته نمی‌شود. دلیل این امر آن است که دستور را کاربر با سطح دسترسی Root اجرا می‌کند.
- وقتی کاربر برای اولین بار ساخته می‌شود نیز با همین دستور باید رمز عبور برای آن تعیین شود وگرنه غیرقابل استفاده است.

تغییر Shell پیش‌فرض

- همان‌طور که قبلاً گفتیم، Shellهای مختلف در لینوکس وجود دارد که Bash بهترین آنها است.
- هنگامیکه کاربر جدیدی توسط خط فرمان تولید می‌شود، Shell پیش‌فرض آن sh است نه Bash.
- برای تغییر Shell پیش‌فرض از دستور زیر استفاده می‌شود:

```
# chsh -s SHELL USER
```

در اینجا USER نام کاربر مورد نظر و SHELL آدرس Shell مورد نظر است.

- آدرس Bash عبارت است از:

```
/bin/bash
```

تغییر گروه اصلی کاربر

- برای تغییر گروه اصلی کاربر USR به گروه GRP از دستور زیر استفاده می‌شود:

```
# usermod -g GRP USR
```

- برای مثال دستور زیر کاربر sample که جدید ساخته شده است را در گروه root قرار می‌دهد:

```
# usermod -g root sample
```

ساخت و حذف گروه

- برای ساخت گروه GRP از دستور زیر استفاده می‌شود:

```
# groupadd GRP
```

- برای حذف گروه نیز از دستور زیر استفاده می‌شود:

```
# groupdel GRP
```

- دقت کنید که برای حذف یک گروه، باید هیچ کاربری در آن گروه قرار نداشته باشد. لذا قبل از پاک کردن یک گروه باید تمام کاربران عضو آن گروه را از عضویت گروه مورد نظر خارج کنید.

فایل حاوی اطلاعات کاربران

- فایل زیر حاوی اطلاعات کاربران است:

`/etc/passwd`

- هر خط اطلاعات مربوط به یک کاربر را نشان می‌دهد که دارای قالب زیر است:

`Uername:Password:uid:gid:Comment:HomeDir:DefaultShellDir`

- فیلدهای با : از هم جدا می‌شوند و عبارتند از:
  - **Username:** نام کاربری
  - **Password:** رمز عبور. اگر به جای آن x قرار داشته باشد، یعنی رمز عبور در فایل /etc/shadow به صورت Hash قرار دارد.
  - **uid:** شماره شناسایی کاربر
  - **gid:** شماره شناسایی گروه اصلی کاربر
  - **Comment:** توضیحاتی در مورد کاربر (بی‌اثر)
  - **HomeDir:** مسیر دایرکتوری Home این کاربر (همان ~) است.
  - **DefaultShellDir:** مسیر Shell پیش‌فرض کاربر.

آشنایی با دارنده و سطح دسترسی

سطوح دسترسی

- هر فایل یا پوشه دارای یک سری سطوح دسترسی است که مشخص می‌کند چه کسی اجازه چه عملی بر روی آن فایل یا پوشه را دارد.
- همچنین هر فایل یا پوشه دارای یک کاربر دارنده و یک گروه دارنده است.
- کاربر دارنده و گروه دارنده دارای سطح دسترسی مجزا می‌باشند؛ به همین علت کاربر و گروه دارنده باید تعیین شود.
- هر فایل یا پوشه دارای سه سطح دسترسی به صورت زیر است:

(سایر) (گروه دارنده) (کاربر دارنده)

- سطح دسترسی هر عبارت داخل پرانتز، مشخص می‌کند که آن کاربر یا گروه چه عملی بر روی آن فایل یا پوشه می‌تواند انجام دهد.
- در واقع پرانتز اول مشخص می‌کند کاربر دارنده چه عملی می‌تواند انجام دهد، پرانتز دوم در مورد اعضای گروه دارنده صحبت می‌کند و پرانتز سوم در مورد کاربری که کاربر دارنده نیست و در گروه دارنده قرار ندارد، صحبت می‌کند.

اجزا سطح دسترسی

- هر سطح دسترسی (پرانتزهای اسلاید قبل) می‌تواند شامل یک یا چند جزء (عمل) زیر باشد:
  - خواندن (R)
  - نوشتن (W)
  - اجرا (X)
- خواندن در مورد فایل به معنی مشاهده محتویات داخل فایل و در مورد پوشه به معنی مشاهده لیست فایل‌ها و پوشه‌های درون آن می‌باشد.
- نوشتن در مورد فایل به معنی حذف، تغییر نام، جابه‌جایی و تغییر محتویات آن است در حالیکه در مورد پوشه به معنی حذف، تغییر نام، جابه‌جایی و ایجاد فایل یا پوشه جدید در آن می‌باشد.
- اجرا در مورد فایل به معنی اجرای فایل توسط سیستم عامل است در حالیکه در مورد پوشه به معنی ورود به پوشه می‌باشد.

نحوه اعمال سطح دسترسی

- وقتی کاربر U می‌خواهد بر روی فایل یا پوشه‌ای عملی انجام دهد، ابتدا بررسی می‌شود کاربر U در کدام دسته است یعنی کدام سطح دسترسی باید اعمال شود. سپس در مورد اجازه یا عدم اجازه بررسی انجام می‌گیرد.
- الگوریتم به صورت زیر است:

1. اگر U کاربر دارنده است، سطح دسترسی کاربر دارنده را برای آن در نظر بگیر.
2. اگر U عضو گروه دارنده است، سطح دسترسی گروه دارنده را برای آن در نظر بگیر.
3. وگرنه سطح دسترسی سایر را برای آن در نظر بگیر.
4. بررسی کن عمل مورد نظر در سطح متناظر وجود دارد یا نه. اگر وجود دارد، کاربر مجاز به انجام عمل مورد نظر است وگرنه پیام مناسب نمایش داده شود.

مثال نحوه اعمال سطح دسترسی

- فرض کنید فایل name.txt دارای کاربر دارنده ali و گروه دارنده student است. همچنین ali و reza دو کاربر عضو گروه student هستند و cl در این گروه نیست. سطح دسترسی این فایل به صورت زیر باشد (علامت – به معنی عدم وجود یک جزء در سطح دسترسی است):

```
rwx r-x r--
```

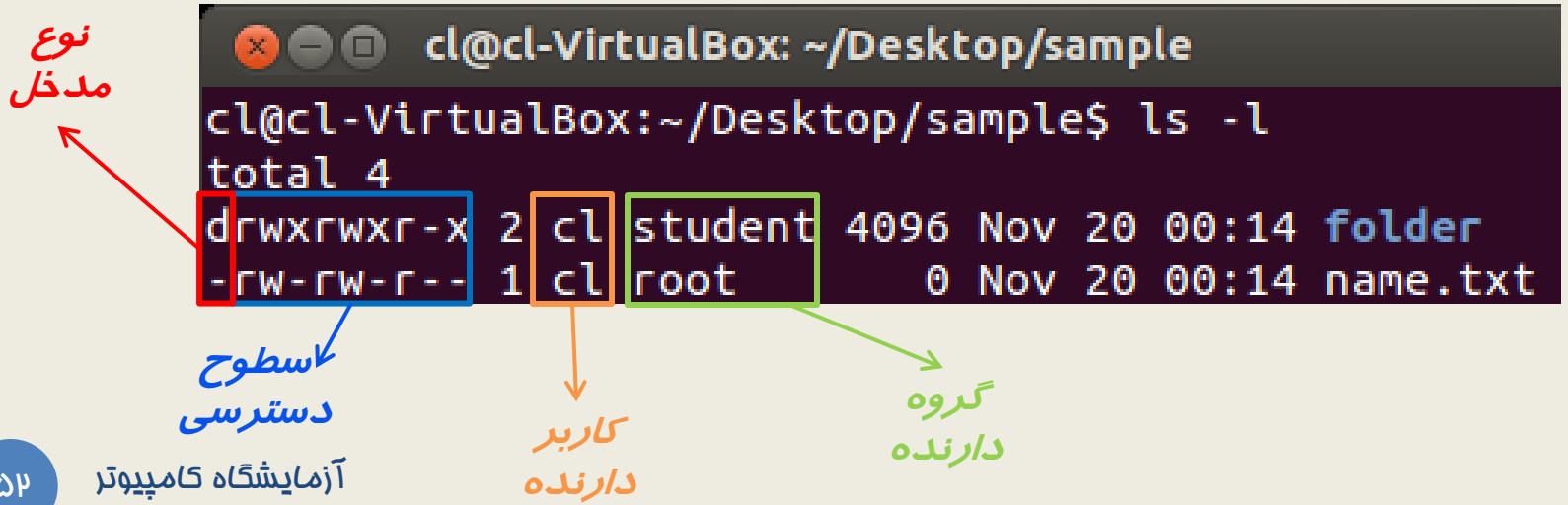
- اعمال زیر انجام می‌شود:
  - کاربر ali می‌خواهد فایل را ویرایش کند. ابتدا بررسی می‌شود که ali کاربر دارنده هست یا نه. چون ali کاربر دارنده است، سطح دسترسی اول یعنی rwx در مورد آن صادق است. چون w در سطح دسترسی آن وجود دارد، پس **می‌تواند** فایل را ویرایش کند.
  - کاربر reza می‌خواهد فایل را ویرایش کند. ابتدا بررسی می‌شود این کاربر همان کاربر دارنده است یا نه؛ که این شرط برقرار نیست. سپس بررسی می‌شود که reza عضو گروه دارنده هست یا نه. چون reza عوض گروه student است پس عضو گروه دارنده است لذا r-x در مورد آن صادق است. از آنجایی که w وجود ندارد پس reza اجازه ویرایش (نوشتن) را **ندارد**. اما می‌تواند فایل را اجرا کند یا محتویات آن را ببیند.
  - کاربر cl می‌خواهد فایل را ویرایش کند. از آنجایی که نه کاربر دارنده است و نه در عضو گروه دارنده، دسترسی سایر یا همان r-- در مورد آن صادق است پس اجازه ویرایش **ندارد**.

مشاهده دارنده و سطوح دسترسی

- برای مشاهده دارنده و سطوح دسترسی از دستور زیر استفاده می‌کنیم:

```
# ls -l
```

- ستون اول سطوح دسترسی به همان ترتیبی که گفته شد، بیان می‌کند. ستون سوم کاربر دارنده است و ستون چهارم گروه دارنده.
- در ستون اول، اولین کاراکتر یکی از موارد d، - یا l است که به ترتیب به معنی پوشه، فایل و Symlink می‌باشد.



تغییر دارنده

- برای تغییر دارنده یک فایل یا پوشه به نام NAME از دستورهای زیر استفاده می‌شود:

```
# chown USR NAME
# chown USR:GRP NAME
```

- دستور اول فقط کاربر دارنده را تغییر می‌دهد در حالیکه دستور دوم هم کاربر دارنده را تغییر می‌دهد و هم گروه دارنده را. USR و GRP به ترتیب نام کاربر و گروه دارنده جدید است.

تغییر سطح دسترسی

- تغییر سطح دسترسی با دستور زیر قابل انجام است:

```
# chmod PERM NAME
```

- NAME نام فایل یا پوشه مورد نظر و PERM مشخص کننده سطح دسترسی است.
- PERM می‌تواند به صورت کاراکتری یا عددی وارد شود.
- شیوه کاراکتری دارای انعطاف‌پذیری بالایی است اما شیوه عددی مرسوم است.

سطح دسترسی کاراکتری

- سطح دسترسی کاراکتری به صورت مجموعه‌ای از سطح دسترسی است که با کاما از هم جدا می‌شوند.
- هر سطح دسترسی تغییری را ایجاد می‌کند و دارای قالب زیر است:

(عمل) (نوع تغییر) (کاربر یا گروه)

- کاربر یا گروه: u برای کاربر دارنده، g برای گروه دارنده، o برای سایر، a یا عدم ذکر به معنی همه
- نوع تغییر: = برای انتساب، + برای افزودن، - برای کم کردم
- عمل: r برای خواندن، w برای نوشتن، x برای اجرا، - عدم وجود یک عمل

مثال سطح دسترسی کاراکتری

- افزودن سطح دسترسی نوشتن به گروه دارنده:

```
# chmod g+w NAME
```

- افزودن سطح دسترسی نوشتن به گروه دارنده و اجرا به سایر:

```
# chmod g+w,o+x NAME
```

- فقط اجازه خواندن به سایرین داده شود. خواندن و نوشتن به گروه دارنده اضافه شود:

```
# chmod g+rw,o=r NAME
```

- عدم اجازه هرگونه دسترسی به سایر:

```
# chmod o=- NAME
```

- افزودن اجازه اجرا به همه:

```
# chmod +x NAME
```

سطح دسترسی عددی

- در روش عددی، معادل عددی هر سطح دسترسی را محاسبه و در کنار سایر سطح دسترسی‌ها قرار می‌دهند تا یک عدد سه رقمی تعیین شود. این عدد سطح دسترسی جدید را مشخص می‌کند.
- در واقع روش عددی فقط انتساب را انجام می‌دهد.
- عدد معادل خواندن ۴، نوشتن ۲ و اجرا ۱ است.
- اگر یک عمل وجود داشته باشد، عدد مربوطه در جمع لحاظ می‌شود.
- برای مثال r-x معادل ۴+۱ یعنی ۵ است و rwx معادل ۷ و -wx معادل ۳ است.
- همچنین ۷۵۱ معادل rwx r-x --x و عدد ۷۱۰ معادل rwx --x --- می‌باشد.

مثال سطح دسترسی عددی

- فقط کاربر دارنده دسترسی داشته باشد:

```
# chmod 700 NAME
```

- کاربر دارنده دسترسی کامل، گروه دارنده دسترسی اجرا و خواندن، سایرین اجازه اجرا:

```
# chmod 751 NAME
```

- همه اجازه هر کاری داشته باشند:

```
# chmod 777 NAME
```

ایجاد دایرکتوری Home کاربر

- گفتیم که وقتی کاربر جدید ساخته می‌شود، باید پوشه‌ای همنام با نام کاربری آن در /home ساخته شده و سطح دسترسی مناسب اعمال گردد.
- پس از ساخت پوشه، باید کاربر دارنده آن به همان کاربر مورد نظر تغییر یابد و سطح دسترسی کاربر دارنده کامل شود. همچنین سطح دسترسی سایر نباید امکان نوشتن داشتن باشد.
- برای مثال بعد از ساخت کاربر ali دستورهای زیر را اعمال می‌کنیم:

```
# mkdir /home/ali
# chown ali /home/ali
# chmod 755 /home/ali
```

سطح دسترسی و Symlink

- گفتیم که Symlink فایلی است که به مکان دیگری اشاره می‌کند. پس خود دارای کاربر و گروه دارنده و سطح دسترسی است.
- سوال این است که در مورد Symlink باید کدام سطح دسترسی و دارنده‌ای در نظر گرفته شود؟
- پاسخ این است که دارنده و سطح دسترسی مسیر واقعی اعمال می‌شود. برای مثال /etc/shadow فایلی است که دارنده آن root است و دسترسی آن ۶۴۰ می‌باشد؛ یعنی سایرین هیچ دسترسی نخواهند داشت. اگر pass.txt یک Symlink به این فایل باشد، اجرای دستور زیر توسط هر کاربری جز root سبب تولید خطای Permission Denied خواهد شد. زیرا سطح دسترسی مسیر اصلی واقعی می‌شود.

```
$ cat pass.txt
```

ورودی/خروجی استاندارد

Pipe & Redirecting

- در لینوکس هر برنامه دارای یک ورودی استاندارد و یک خروجی استاندارد می‌باشد.
- ورودی استاندارد با نام Stdin و خروجی استاندارد با نام Stdout شناخته می‌شود.
- به طور معمول ورودی استاندارد، صفحه کلید است؛ یعنی برنامه داده‌های مورد نیاز را از طریق صفحه کلید دریافت می‌کند.
- همچنین به طور معمول خروجی استاندارد پنجره ترمینال می‌باشد؛ یعنی برنامه خروجی‌های خود را به ترمینال می‌دهد تا ترمینال آن را نمایش دهد.
- برخی برنامه‌ها می‌توانند از فایل بخوانند یا در آن بنویسند یعنی به جای آنکه نتایج در ترمینال نمایش داده شود، درون فایل قرار گیرد و به جای آنکه برنامه داده‌ها را از صفحه کلید دریافت کند، از فایل بخواند.
- اما برخی برنامه‌ها توان کار با فایل را ندارند.

Redirecting

- با استفاده از Redirecting می‌توان ورودی/خروجی استاندارد یک برنامه را از هر چیزی که هست به فایل منتقل کرد.
- یعنی اگر برنامه قابلیت خواندن از فایل را ندارد، با تکنیک Redirecting آن را مجبور کرد از فایل بخواند، بدون آنکه برنامه متوجه این تغییر شود.
- برای انتقال خروجی استاندارد دستور CMD به فایل FILE از دستور زیر استفاده می‌شود:

```
$ CMD > FILE
```

- دقت شود که دستور بالا فایل را بازنویسی می‌کند؛ یعنی داده‌های قبلی فایل در صورت وجود نابود خواهند شد.
- برای انتقال ورودی استاندارد دستور CMD به فایل FILE از دستور زیر استفاده می‌شود:

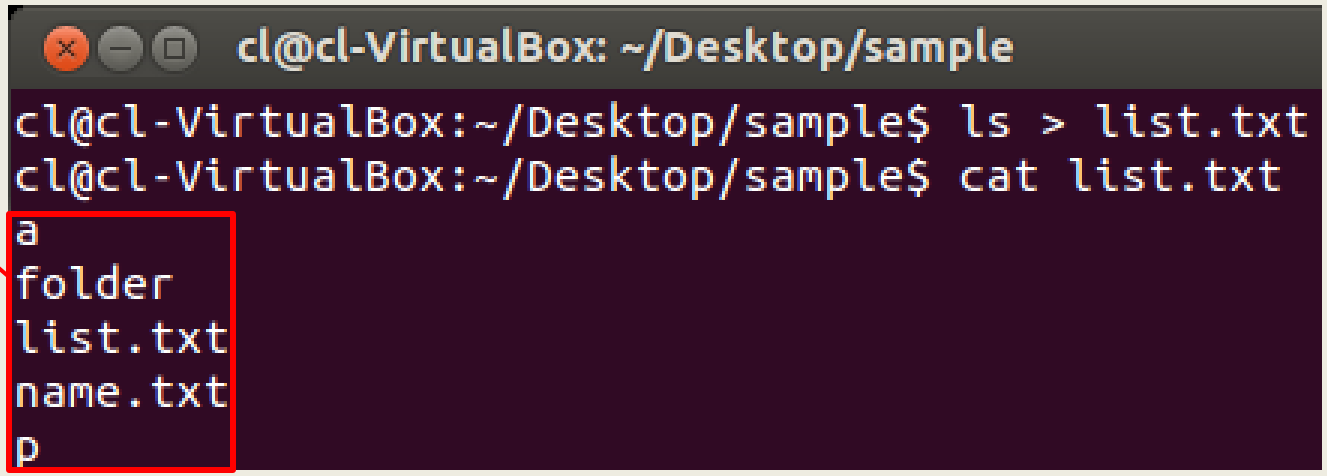
```
$ CMD < FILE
```

Redirecting (مثال خروجی)

- دستور ls لیست فایل‌ها و پوشه‌های موجود در دایرکتوری را نشان می‌دهد. اما قابلیت خروجی در فایل را ندارد. با استفاده از Redirect خروجی آن را به جای ترمینال به فایل منتقل می‌کنیم.
- دستور زیر خروجی ls را به فایل list.txt منتقل می‌کند:

```
$ ls > list.txt
```

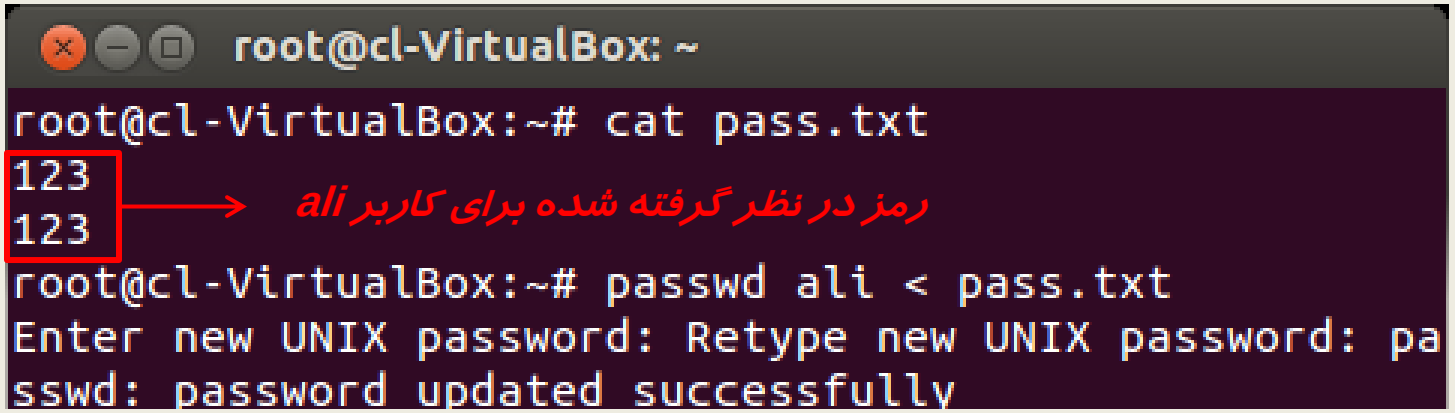
- تصویر زیر نتیجه این مثال را نشان می‌دهد:



Redirecting (مثال ورودی)

- ابزار passwd برای تعیین رمز عبور استفاده می‌شود. این برنامه قابلیت خواندن از فایل را ندارد و فقط می‌تواند از صفحه کلید رمزها را بخواند.
- در این مثال passwd را مجبور می‌کنیم از فایل pass.txt رمزها را بخواند:

```
# passwd ali < pass.txt
```



- دقت شود که چون passwd از کاربر می‌خواهد که رمز عبور را دو بار وارد کند، فایل pass.txt باید رمز عبور را دو بار تکرار کند. همچنین کاراکتر انتهای خط یا همان \n معادل فشردن کلید Enter است.

اتصال خروجی/ورودی برنامه‌ها

Pipe & Redirecting

- گاهی اوقات می‌خواهیم خروجی یک برنامه به ورودی برنامه دیگر داده شود. یک راه آن است که خروجی برنامه اول را به فایل منتقل کرده و سپس ورودی فایل دوم را به همان فایل منتقل کنیم.
- برای مثال فرض کنید ابزار passgen، رمز عبوری را به صورت تصادفی تولید و دوبار تکرار کند. می‌خواهیم رمز عبور کاربر ali را برابر با رمز عبوری این برنامه قرار دهیم تا از سربار این کار برای خودمان بکاهیم. یک راه استفاده از دو خط زیر است:

```
# passgen > pass.txt
# passwd ali < pass.txt
```

- این کار اگرچه خوب است اما فایل pass.txt ایجاد می‌شود که پاک کردن آن بر عهده خودمان است.

Pipe

- راه دیگر برای حل مشکل اسلاید قبل، استفاده از Pipe است.
- با استفاده از Pipe یک فایل موقت ایجاد می‌شود و خروجی برنامه اول و ورودی برنامه دوم به این فایل منتقل می‌شود. پس از اجرای کامل دستور، فایل موقت پاک می‌شود.
- شکل کلی استفاده از Pipe به صورت زیر است:

```
$ CMD1 | CMD2
```

- این دستور خروجی CMD1 را به ورودی CMD2 وصل می‌کند. کاراکتر بین این دو، کاراکتر Pipe با نماد | می‌باشد.
- برای مثال سناریو اسلاید قبل با دستور زیر به راحتی قابل انجام است:

```
# passgen | passwd
```

- می‌توان هر تعداد Pipe در یک خط داشت. همچنین Pipe و Redirect با هم نیز قابل استفاده هستند.

کاربر Pipe

Pipe & Redirecting

- معمولاً از Pipe برای فیلترگذاری و جداکردن بخشی از خروجی استفاده می‌شود؛ یعنی قسمتی از خروجی که لازم است تولید شود.
- ابزارهای بسیاری برای فیلترگذاری و جداکردن بخشی از خروجی وجود دارد، برخی از معروف‌ترین آنها عبارتند از:
  - grep
  - cut
  - tr
  - head
  - tail
  - awk
- در ادامه به بررسی برخی آنها می‌پردازیم.

grep

- یک ابزار فیلترگذاری است که ورودی را خط به خط بررسی کرده و دنبال الگوی مورد نظر می‌گردد.
- اگر الگو یافت شود، آن خط را نمایش داده و هر رخداد آن را رنگی می‌کند.
- قالب دستور به صورت زیر است:

```
grep PATTERN
```

PATTERN همان الگویی است که دنبال آن می‌گردیم.

- برای مثال در دایرکتوری کاری دنبال فایل یا پوشه‌هایی هستیم که نام آنها حاوی عبارت Net است، بدین منظور از دستور زیر استفاده می‌کنیم:

```
$ ls | grep  Net
```

- در مثال دیگر دنبال فایل یا پوشه با سطح دسترسی 770 هستیم:

```
$ ls -l | grep  rwxrwx---
```

cut

- وقتی بخواهیم قسمتی از خط را نمایش دهیم از این ابزار استفاده می‌شود.
- قالب دستور به صورت زیر است:

```
cut TYPE LIST
```

- TYPE مشخص می‌کند جداسازی بر اساس چه چیزی باشد. c– برای کاراکتر و f– برای فیلد به کار می‌رود.
- LIST لیستی از کاراکترها یا فیلدهای مورد نظر را نشان می‌دهد. هر لیست از یک سری محدوده تشکیل می‌شود که با کاما از هم جدا می‌شوند. محدوده‌ها می‌توانند به یکی از شکل‌های زیر باشند:
  - **N:** کاراکتر یا فیلد Nام. شمارش از یک شروع می‌شود.
  - **N-:** کاراکتر یا فیلد Nام تا آخرین.
  - **N-M:** کاراکتر یا فیلد Nام تا Mام (شامل خود آن‌ها).
  - **M-:** از کاراکتر یا فیلد اول تا Mام.

cut (ادامه)

Pipe & Redirecting

- برای مثال اگر در خروجی یک برنامه بخواهیم کاراکتر ششم نمایش داده شود، از دستور زیر استفاده می‌کنیم:

```
$ CMD | cut -c 6
```

- اگر کاراکترهای ۷ تا ۱۰ مد نظر باشد از دستور زیرا استفاده می‌کنیم:

```
$ CMD | cut -c 7-10
```

- اگر کاراکترهای ۳ و ۵ تا ۱۵ مد نظر باشد از دستور زیرا استفاده می‌شود:

```
$ CMD | cut -c 3,5-15
```

- اگر کاراکترهای ۲ تا آخر مد نظر باشد از دستور زیرا استفاده می‌شود:

```
$ CMD | cut -c 2-
```

تعریف فیلد

- یک «فیلد» (Field) مجموعه‌ای از کاراکترها است که با استفاده از یک «جداکننده» (Delimiter) از مجموعه دیگر جدا شده است.
- برای مثال خط زیر را در نظر بگیرد:

```
This is sample text
```

- اگر جدا کننده را i در نظر بگیریم، دارای سه فیلد خواهیم بود. هر فیلد با رنگ از بقیه متمایز شده است:

```
This is sample text
```

- حال اگر جدا کننده فاصله باشد، ۴ فیلد خواهیم داشت:

```
This is sample text
```

فیلد در cut

Pipe & Redirecting

- ابزار cut توانایی تشخیص فیلد را نیز دارد. همانطور که گفته شد برای جداسازی بر اساس فیلد از f– استفاده می‌شود.
- به صورت پیش‌فرض جداکننده کاراکتر \t می‌باشد. برای تغییر جداکننده از سوئیچ d– استفاده می‌شود:
- برای مثال اگر فیلد دوم خروجی با جداکنندگی کاراکتر \t مورد نظر باشد از دستور زیر استفاده می‌کنیم:

```
$ CMD | cut -f 2
```

- حال اگر بخواهیم کاراکتر : جداکننده باشد از دستور زیر استفاده می‌کنیم:

```
$ CMD | cut -f 2 -d :
```

- برای جداکنندگی فاصله از دستور زیر استفاده می‌کنیم:

```
$ CMD | cut -f 2 -d ' '
```

فیلد در cut (مثال)

- **صورت مسئله:** می‌خواهیم uid کاربر ali را بدست بیاوریم.
- **دانسته‌ها:**
  - می‌دانیم که اطلاعات کاربران در فایل /etc/passwd موجود است.
  - می‌دانیم که uid فیلد سوم است و هر فیلد با : جدا می‌شود.
- **راه حل:**
  - ابتدا باید محتوای فایل مورد نظر را به grep فرستاده و سطر مربوط به کاربر ali را جدا کنیم.

```
$ cat /etc/passwd | grep ali
```

  - سپس فیلد سوم در این سطر با جداکنندگی : را نمایش می‌دهیم.

```
$ cat /etc/passwd | grep ali | cut -f 3 -d :
```

  - دقت کنید ممکن است عبارت ali در فیلدهای دیگر مخصوصاً Comment سایر سطرها نیز یافت شود. برای حل این مشکل از دستور زیر استفاده می‌کنیم:

```
$ cat /etc/passwd | cut -f 1,3 -d : | grep ali | cut -f 2 -d :
```

tr

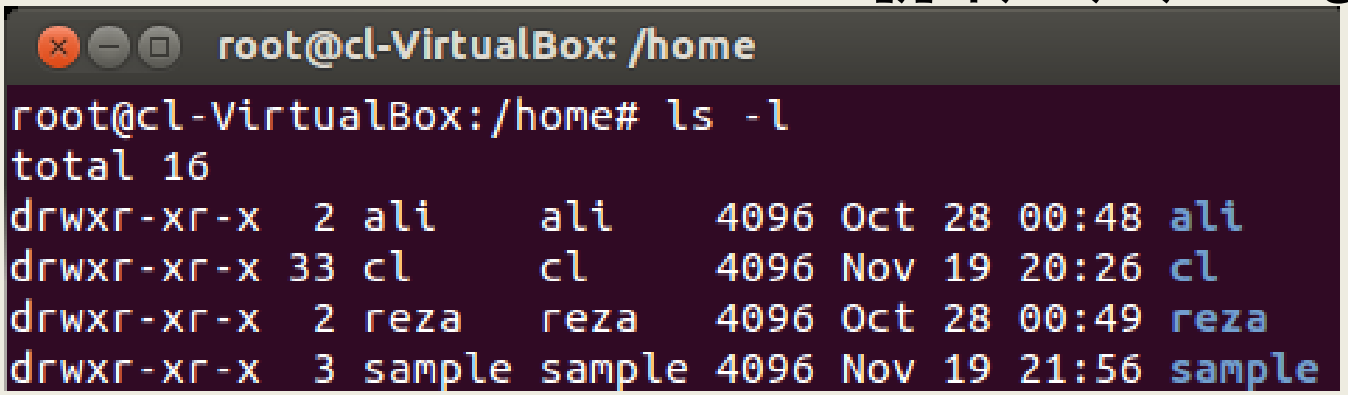
- از این ابزار برای تبدیل یا حذف قسمتی از کاراکترها استفاده می‌شود.
- سوئیچ s– در این ابزار به معنی Squeeze Repeats است یعنی تکرارهای متوالی از کاراکتر تعیین شده را یکی کن.
- قالب دستور به صورت زیر است:

```
tr -s CHAR
```

- CHAR همان کاراکتری است که تکرارهای متوالی آن باید یکی شود.
- برای مثال اگر دستور tr –s 0 بر روی 10002003 اجرا شود، حاصل 10203 خواهد بود زیرا صفرهای متوالی آن یکی می‌شوند.

مثالی از کاربرد tr

- خروجی ls –l را در نظر بگیرید:



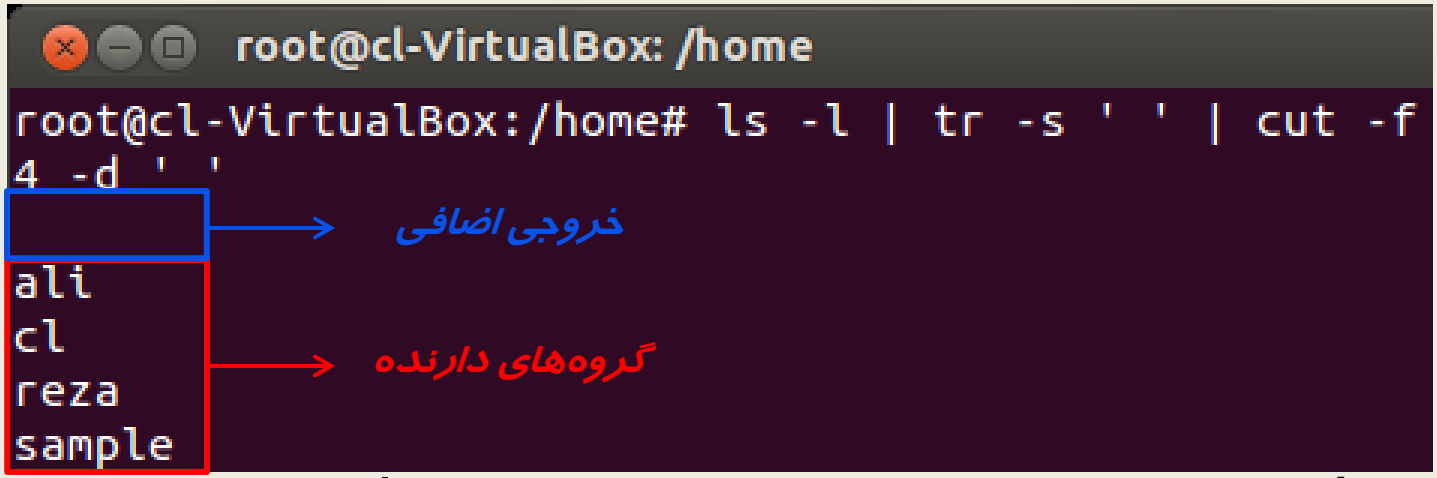
- اگر بخواهیم گروه دارنده را فقط نمایش دهیم بهترین راه آن است که نتیجه را به ابزار cut داده و با جداکنندگی فاصله، فیلد چهارم را بخواهیم. اما این کار خواسته ما را برآورده نمی‌کند.
- دلیل آن وجود فاصله‌های مختلف در هر سطر است. cut با رسیدن به هر فاصله فکر می‌کند به فیلد بعدی رسیده است در حالیکه ls برای نمایش بهتر قرار گرفتن خروجی در یک ستون به تعداد لازم فاصله گذاشته است.

مثالی از کاربرد tr (ادامه)

- برای حل این مشکل کافی است با tr فاصله‌های متوالی را یکی کنیم. دستور زیر ما را به مقصودمان می‌رساند:

```
$ ls -l | tr -s ' ' | cut -f 4 -d ' '
```

- خروجی به صورت زیر خواهد بود:



- دقت شود که خط اول اضافی است زیرا خط اول ls حجم محتویات را نشان می‌دهد و دارای قالب متفاوتی از سایر سطرها است.

head

- ابزاری است برای جدا کردن تعداد خط‌های مورد نیاز از ابتدای خروجی است.
- به صورت پیش‌فرض ۱۰ خط اول را نشان می‌دهد. برای مثال دستور زیر ۱۰ خط اول خروجی ls –l را نمایش می‌دهد:

```
$ ls -l | head
```

- اگر از سوئیچ n– استفاده کنیم، می‌توان تعداد خطوط را تعیین کنیم. برای مثال دستور زیر ۲۰ خط اول را نشان می‌دهد:

```
$ ls -l | head -n 20
```

tail

- همانند head است با این تفاوت که از انتها عمل می‌کند.
- برای مثال دستور زیر ۱۰ خط آخر را نشان می‌دهد:

```
$ ls -l | tail
```

- اگر از سوئیچ n– استفاده کنیم، می‌توان تعداد خطوط را تعیین کنیم. برای مثال دستور زیر ۲۰ خط آخر را نشان می‌دهد:

```
$ ls -l | tail -n 20
```

- مثال tr را در نظر بگیرید که سطر اول آن اضافی بود. برای حذف این سطر از tail به صورت زیر بهره می‌گیریم:

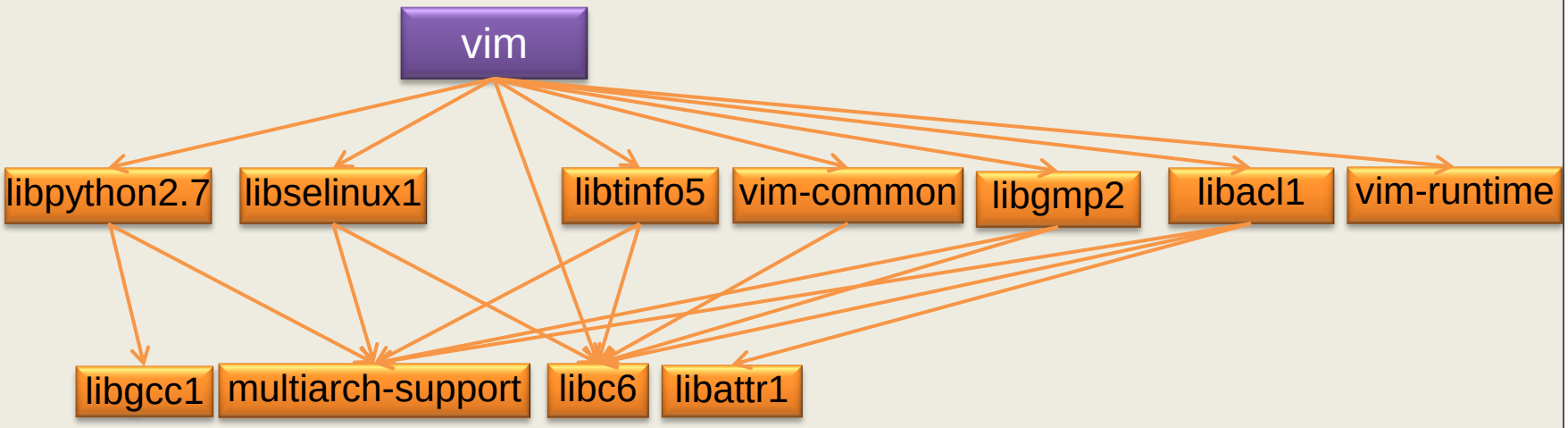
```
$ ls -l | tr -s ' ' | cut -f 4 -d ' ' | tail -n 4
```

پکیج و وابستگی

- در لینوکس مقدماتی دیدیم که برنامه‌ها، کتابخانه‌ها و ... به صورت پکیج ارائه می‌شوند؛ یعنی مثلاً برای نصب یک برنامه باید پکیج آن را نصب کنیم.
- به منظور استفاده مجدد از کدهای نوشته شده در سایر ابزارها، قسمت‌های مشترک بین ابزارها تحت عنوان پکیج جداگانه‌ای عرضه می‌شود.
- در نتیجه برای استفاده از یک پکیج، شاید لازم باشد پکیج‌های دیگری را نیز داشته باشیم.
- در واقع یک پکیج ممکن است به پکیج‌های دیگر وابسته باشد. به این ارتباط «وابستگی» (Dependency) گوئیم.
- یک وابستگی مشخص می‌کند که نسخه مشخصی از یک پکیج به چه نسخه‌ای از پکیج‌های دیگر وابسته است.
- وابستگی ممکن است دقیقاً به نسخه خاصی از یک پکیج باشد یا محدوده‌ای از نسخه‌ها.

گراف وابستگی

- اگر پکیج‌ها را با گره و وابستگی را با یال جهت‌دار نشان دهیم، هر پکیج دارای یک گراف وابستگی خواهد بود که مشخص می‌کند برای نصب آن ابتدا باید چه پکیج‌هایی نصب شوند.
- برای مثال گراف وابستگی vim به صورت زیر است. از ذکر نسخه پکیج اجتناب شده است. همچنین به علت تعدد وابستگی‌ها، از آوردن برخی وابستگی‌ها صرف‌نظر کردیم.

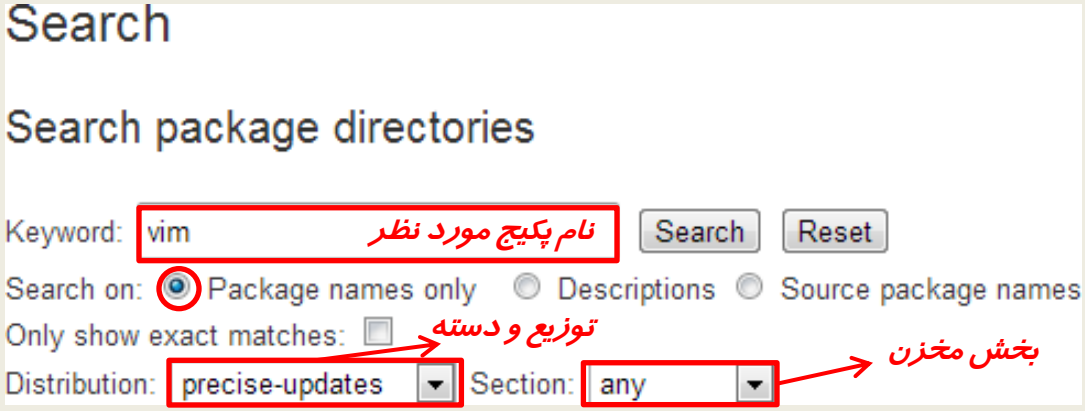


استخراج وابستگی‌ها

- از آنجایی که پکیج‌ها ممکن است به هم وابستگی داشته باشند، گام اول در نصب پکیج، تشخیص وابستگی آن پکیج است؛ زیرا باید ابتدا مطمئن شویم پکیج‌های مورد نیاز نصب هستند، سپس به نصب پکیج خود بپردازیم.
- یکی از راه‌های تشخیص وابستگی‌های یک پکیج، دریافت آن و تعیین وابستگی‌ها از روی اطلاعات داخل پکیج است. بعداً با این روش آشنا می‌شویم.
- راه بالا دشوار و زمان‌بر است، زیرا برای کشف وابستگی‌ها، باید تمام پکیج‌های لازم را دریافت کنیم.
- راه دیگر استفاده از سایت‌هایی است که این اطلاعات را در اختیار ما می‌گذارند. نمونه این سایت‌ها، packages.ubuntu.com است که برای پکیج‌های ابونتو ساخته شده است.

یافتن پکیج در سایت ابونتو

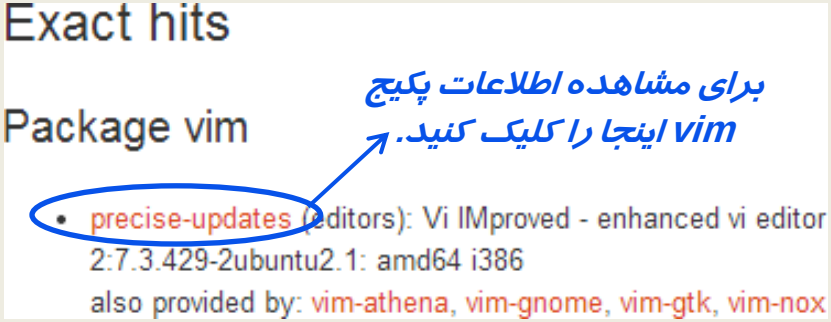
- پس از مراجعه به آدرس http://packages.ubuntu.com در وسط صفحه امکان جستجوی پکیج همانند شکل زیر وجود دارد:



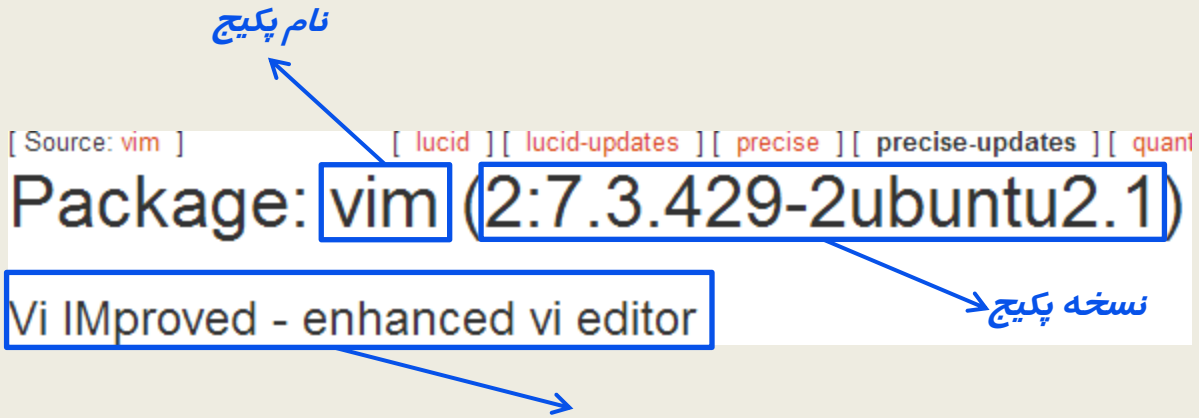
- توزیع و دسته مشخص می‌کند پکیج در کدام توزیع و در کدام دسته جستجو شود. دسته‌ها با خط فاصله از توزیع جدا می‌شوند. اولین نسخه از پکیج درون دسته precise قرار می‌گیرد. اگر نسخه جدیدی از آن پکیج تولید شود، درون precise-updates قرار می‌گیرد.
- بخش مخزن یکی از موارد any، main،restricted و universe و multiverse می‌تواند باشد که در لینوکس مقدماتی با آن آشنا شدیم.

یافتن پکیج در سایت ابونتو (ادامه)

- در صورت پیدا شدن پکیج، همانند شکل امکان ورود به اطلاعات پکیج مورد نظر وجود دارد.

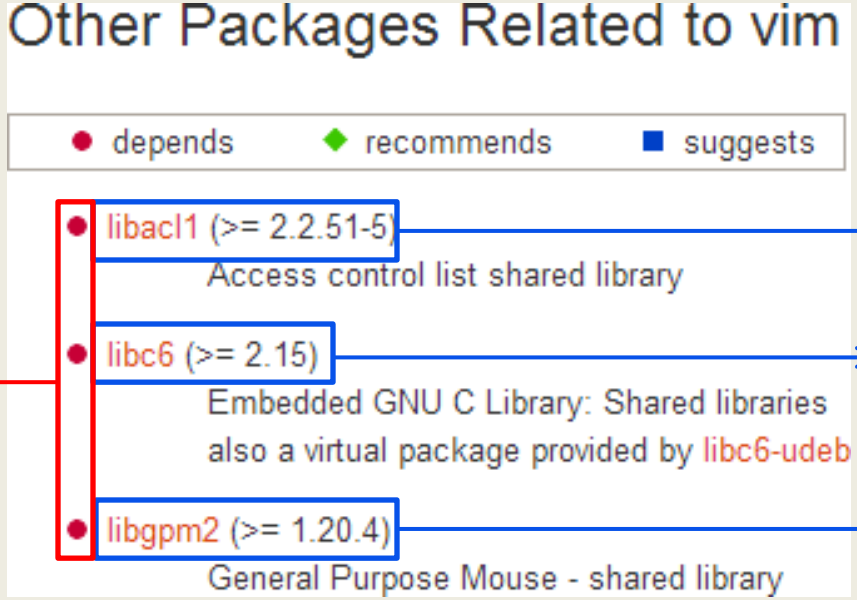


- در صفحه بعد کلیه اطلاعات پکیج از جمله نام، نسخه، نام نویسنده، وابستگی‌ها و امکان دانلود فراهم شده است.



استخراج وابستگی‌ها از سایت ابونتو

- یکی از اطلاعاتی که در packages.ubuntu.com پس از یافتن پکیج قابل بدست آوردن است، لیست وابستگی‌های پکیج مورد نظر است.
- وابستگی‌ها با رنگ قرمز در زیر عبارت Other Packages Related to قابل مشاهده است. هر گزینه یک پکیج مورد نیاز به همراه نسخه آن را نشان می‌دهد.
- برای مثال در اسلاید قبل پکیج vim را در سایت ابونتو یافتیم و اطلاعات آن را مشاهده کردیم. در زیر برخی از وابستگی‌های پکیج vim را مشاهده می‌کنیم.



این نسخه از vim به پکیج libacl1 نسخه‌های بزرگتر یا مساوی 2.2.51-5 نیاز دارد.

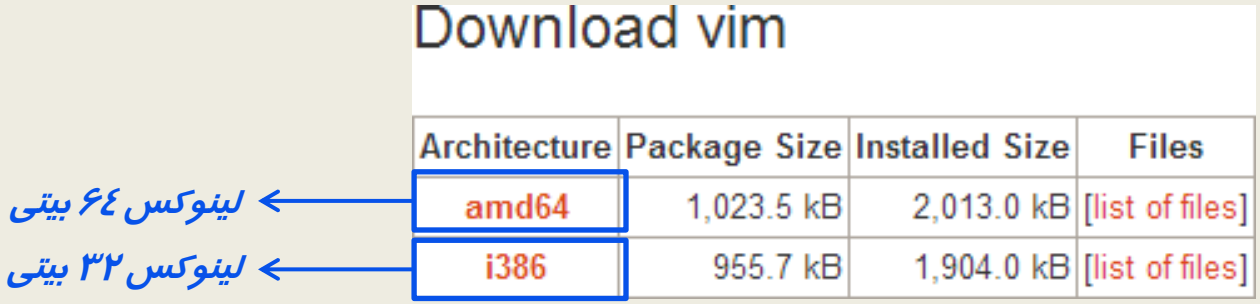
این نسخه از vim به پکیج libc6 نسخه‌های بزرگتر یا مساوی 2.15 نیاز دارد.

این نسخه از vim به پکیج libgpm2 نسخه‌های بزرگتر یا مساوی 1.20.4 نیاز دارد.

پکیج vim به این پکیج‌ها وابسته است

دریافت پکیج از سایت ابونتو

- از دیگر فواید سایت ابونتو امکان دریافت پکیج است.
- در پایین صفحه زیر عبارت Download می‌توان پکیج را برای معماری‌های مختلف دریافت کرد.



- i386 برای لینوکس ۳۲ بیتی و amd64 برای لینوکس ۶۴ بیتی است. all نیز به معنی قابل نصب بودن پکیج بر روی هر دو گونه لینوکس است.
- پس از انتخاب معماری، صفحه‌ای پدیدار می‌گردد که امکان دانلود پکیج (با پسوند deb.) را از سرورهای مختلف در سراسر دنیا را فراهم می‌کند.
- انتخاب از سرورهای نزدیک به محل زندگی، سرعت بیشتری را در پی خواهد داشت.

نصب پکیج با dpkg

- پس از دریافت پکیج‌های لازم باید آن‌ها را به ابزار مناسب بدهیم تا برای ما نصب کند.
- ابزار نصب پکیج در دبیان، dpkg می‌باشد.
- برای نصب پکیج PKG.deb از دستور زیر استفاده می‌شود:

```
# dpkg -i PKG.deb
```

- توجه داریم که قبل از نصب یک پکیج، باید تمام پکیج‌هایی که به آن وابسته است را نصب کرده باشیم وگرنه پیام خطایی مبنی بر عدم رعایت وابستگی‌ها دریافت خواهیم کرد.
- اگر تمام پکیج‌های مورد نیاز موجود باشد، می‌توان همه آنها را در یک خط به dpkg داد؛ کافی است نام فایل‌های deb. با فاصله از هم جدا شود.
dpkg به صورت هوشمند ابتدا پکیج‌هایی که بقیه به آن‌ها وابسته هستند را نصی می‌کند، سپس پکیج‌هایی که دارای وابستگی هستند.

مثال نصب با dpkg

- فرض کنید می‌خواهیم vim را نصب کنیم.
- با مراجعه به سایت ابونتو از وابستگی‌های آن مطلع می‌شویم.
- بررسی می‌کنیم که همه وابستگی‌ها بجز vim-common و vim-runtime نصب است.
- پس لازم است علاوه بر vim، دو پکیج دیگر را نیز دریافت کنیم.
- راه اول استفاده از سه دستور dpkg برای نصب پکیج‌ها با رعایت وابستگی آن‌ها است:

```
# dpkg -i vim-common_7.3.429-2ubuntu2.1_i386.deb
# dpkg -i vim-runtime_7.3.429-2ubuntu2.1_all.deb
# dpkg -i vim_7.3.429-2ubuntu2.1_i386.deb
```

- راه دوم استفاده از یک دستور dpkg و معرفی هر سه پکیج به صورت یکجا است(به علت کمبود جا، دستور در سه خط نشان داده می‌شود ولی در واقعیت یک خط است):

```
# dpkg -i vim-common_7.3.429-2ubuntu2.1_i386.deb vim-
runtime_7.3.429-2ubuntu2.1_all.deb vim_7.3.429-
2ubuntu2.1_i386.deb
```

مثال خطای وابستگی در dpkg

- فرض کنید در همان سناریو نصب vim، پکیج‌های vim-common و vim-runtime را دریافت نکرده باشیم.
- تلاش برای نصب vim سبب تولید خطای وابستگی به شکل زیر خواهد شد:

```
root@cl-VirtualBox: ~
root@cl-VirtualBox:~# dpkg -i vim_7.3.429-2ubuntu2.1_i386.deb
Selecting previously unselected package vim.
(Reading database ... 153745 files and directories currently insta
lled.)
Unpacking vim (from vim_7.3.429-2ubuntu2.1_i386.deb) ...
dpkg: dependency problems prevent configuration of vim:
 vim depends on vim-common (= 2:7.3.429-2ubuntu2.1); however:
  Package vim-common is not installed.
 vim depends on vim-runtime (= 2:7.3.429-2ubuntu2.1); however:
  Package vim-runtime is not installed.
dpkg: error processing vim (--install):
 dependency problems - leaving unconfigured
Errors were encountered while processing:
 vim
```

پکیج vim به vim-cmmon و vim-runtime وابسته است

حذف پکیج با dpkg

- وقتی به پکیجی نیاز نداریم، باید آن را حذف کنیم.
- در حذف پکیج با dpkg برخلاف نصب، اسم پکیج آورده می‌شود. این در حالی است که در هنگام نصب نام فایل deb. ذکر می‌شد.
- برای حذف پکیج PKGNAME از دستور زیر استفاده می‌شود:

```
# dpkg -r PKGNAME
```

- برای مثال برای حذف vim از دستور زیر استفاده می‌کنیم:

```
# dpkg -r vim
```

- سوئیچ r– به معنی Remove است و هر چیزی در اثر نصب پکیج مورد نظر ایجاد شده است را پاک می‌کند بجز فایل تنظیمات (که معمولاً در /etc/ قرار دارد).
- برای پاک کردن کل فایل‌های یک پکیج (شامل فایل تنظیمات) از سوئیچ P– به معنی Purge استفاده می‌شود:

```
# dpkg -P PKGNAME
```

نکاتی در مورد حذف پکیج با dpkg

- وقتی به dpkg دستور می‌دهید تا پکیجی را پاک کند، فقط همان پکیج را پاک می‌کند و به دیگر پکیج‌ها کاری ندارد. نتیجه آنکه اگر پکیجی مثل vim-common و vim-runtime به خاطر نصب vim، نصب شده باشند، در هنگام حذف vim، حذف نخواهند شد.
- نمی‌توان پکیجی را پاک کرد که پکیج دیگری به آن وابسته است. البته این امر بدیهی است زیرا مثلاً وقتی vim به vim-runtime نیاز دارد، در صورت پاک شدن vim-runtime، vim نیز از کار می‌افتد.
- dpkg فقط فایل‌هایی را می‌تواند پاک کند که پکیج مربوطه آن‌ها را اعلام کرده باشد. برای مثال سندی که توسط vim ایجاد می‌شود، تحت تاثیر حذف vim قرار نمی‌گیرد. همین مسئله در مورد فایل تنظیمات که در حین اجرای برنامه به صورت پویا ایجاد شده نیز صادق است.

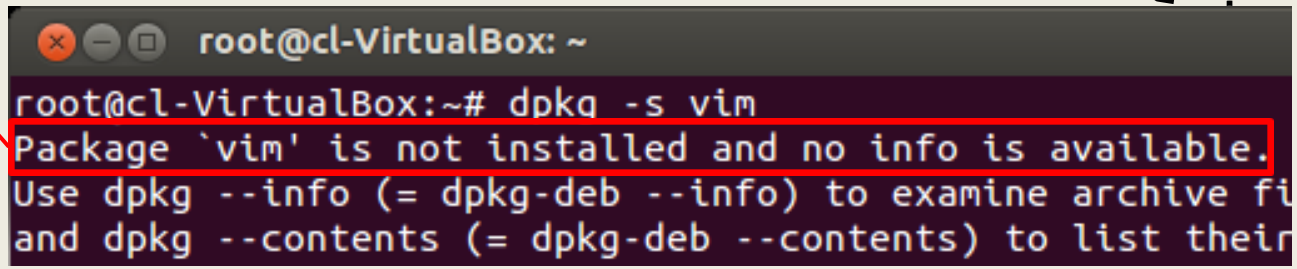
مشاهده اطلاعات پکیج نصب شده

- برای مشاهده اطلاعات پکیج نصب شده از dpkg با سوئیچ s- یا p– استفاده می‌شود.
- برای مثال برای مشاهده اطلاعات پکیج نصب شده vim از دستور زیر استفاده می‌شود:

```
# dpkg -s vim
```

- در صورت یافت شدن پکیج، اطلاعات آن نمایش داده می‌شود؛ وگرنه پیام خطایی مبنی بر عدم نصب پکیج، نمایش داده می‌شود:

پکیج vim نصب نیست.



- دقت کنید dpkg فقط در مورد پکیج‌هایی که توسط این ابزار نصب شده‌اند اطلاع دارد نه پکیج‌هایی که از روش‌های دیگر نصب شده‌اند.
- تفاوت s– و p– در اطلاعاتی است که نشان می‌دهند اما بسیاری از فیلدها در هر دو مشترک است.

مشاهده اطلاعات پکیج نصب شده (ادامه)

- مهم‌ترین فیلدهای خروجی dpkg با سوئیچ s– و p– در زیر توضیح داده شده‌اند:
  - **Package:** نام پکیج
  - **Status:** وضعیت پکیج در سیستم (نصب شده، استخراج شده و ...)
  - **Section:** دسته‌ای که پکیج در آن قرار می‌گیرد. دسته‌بندی بر اساس زمینه کاربرد پکیج است.
  - **Architecture:** معماری که پکیج برای آن ساخته شده است (مثلاً ۳۲ بیتی یا ۶۴ بیتی)
  - **Version:** شماره نسخه کامل پکیج
  - **Depends:** پکیج‌هایی که به آن‌ها وابسته است.
  - **Conffiles:** آدرس فایل‌های تنظیمات این پکیج
  - **Description:** توضیحاتی در مورد پکیج و کاربرد آن

مشاهده اطلاعات پکیج نصب نشده

- گاهی اوقات لازم داریم اطلاعات پکیج deb. که در اختیار داریم را بدست آوریم؛ یعنی پکیج هنوز نصب نشده و ما به اطلاعات آن لازم داریم.
- برای این کار با سوئیچ I– از ابزار dpkg این کار را انجام می‌دهیم:

```
# dpkg -I PKG.deb
```

- برای مثال فایل پکیج vim را از سایت ابونتو دریافت کرده و با دستور زیر اطلاعات آن را بدست می‌آوریم:

```
# dpkg -I vim_7.3.429-2ubuntu2.1_i386.deb
```

- سوئیچ I– نیز حاوی اطلاعاتی شبیه خروجی s– و p– می‌باشد.
- از اطلاعات مهم خروجی سوئیچ I– فیلد Depends است که مشخص می‌کند فایل پکیجی که در اختیار دارید چه وابستگی‌هایی دارد.
- استفاده از سوئیچ I– یکی دیگر از راه‌های استخراج وابستگی است.

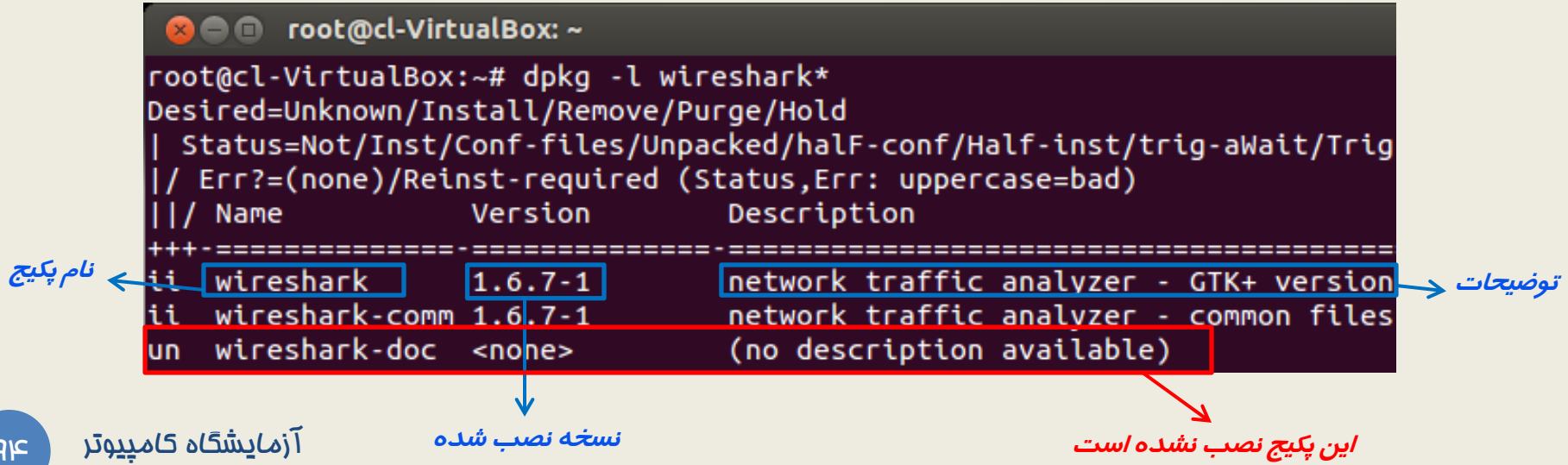
نمایش لیست پکیج‌های نصب شده

- برای نمایش لیست پکیج‌های نصب شده از dpkg با سوئیچ l– به صورت زیر استفاده می‌شود:

```
# dpkg -l
```

- اگر به دنبال همین اطلاعات برای پکیج یا دسته خاصی از پکیج‌ها باشیم می‌توان اسم یا دسته پکیج‌ها را در جلوی l– به صورت زیر مشخص کرد.
- مثلاً برای مشاهده اطلاعات کلیه پکیج‌هایی که با wireshark شروع می‌شوند:

```
# dpkg -r wireshark*
```



تاثیر نصب پکیج بر فایل‌ها

- هر پکیج حاوی یک سری فایل است که باید در سیستم عامل در جای مناسب قرار گیرد تا بتوان گفت پکیج نصب شده است.
- نصب یک پکیج علاوه بر ایجاد فایل‌ها یا پوشه‌های جدید، سبب تغییر در فایل‌ها یا پوشه‌های جدید نیز می‌شود.
- برای یافتن فایل‌ها یا پوشه‌هایی که در اثر نصب پکیج vim تحت تاثیر قرار گرفته‌اند، از دستور زیر استفاده می‌شود:

```
# dpkg -L vim
```

- برای یافتن فایل‌ها و پوشه‌هایی که در اثر نصب پکیج vimی که فایل deb. آن را در اختیار داریم، تحت تاثیر خواهند گرفت، از دستور زیر استفاده می‌شود:

```
# dpkg -c vim_7.3.429-2ubuntu2.1_i386.deb
```

- برای یافتن لیست پکیج‌هایی که بر روی فایل یا پوشه با مسیر PATH تاثیر گذاشته‌اند از دستور زیر استفاده می‌شود:

```
# dpkg -S PATH
```

- دقت شود که L– نام پکیج نصب شده را می‌گیرد در حالیکه c– نام فایل پکیج نصب نشده.

ابزار APT

- دیدیم که نصب پکیج با استفاده از dpkg مشکل است زیرا در نظر گرفتن وابستگی‌ها و دانلود پکیج‌های لازم برای انسان کاری طاقت فرسا است.
- APT یا Advanced Package Tool مجموعه ابزاری است که همه کارهای لازم برای نصب یک پکیج را به صورت خودکار انجام می‌هد.
- برخی وظایف مهم که APT به جای کاربر انجام می‌دهند عبارت است از:
  - یافتن و در نظر گرفتن وابستگی‌ها
  - حل ناسازگاری بین پکیج‌ها
  - دریافت پکیج‌های لازم
  - نصب پکیج‌های لازم
  - حذف پکیج‌های غیرضروری
  - به روزرسانی ساده پکیج‌ها
- APT برای انجام وظایف خود از مخزن‌هایی که به آن شناسانده شده است، استفاده می‌کند.

لیست مخازن و مکانیزم کاری APT

- به صورت پیش‌فرض محل مخازن در فایل زیر برای APT تعریف می‌شوند:

`/etc/apt/sources.list`

- ابزار APT با مراجعه به این مخازن و دریافت اطلاعات آن‌ها، گرافی از وابستگی‌ها و مکان فایل پکیج‌ها ساخته و بر روی کامپیوتری که در حال اجرا است، تحت عنوان «فایل‌های شاخص» (Index Files) ذخیره می‌کند.
- پس از تشکیل فایل‌های شاخص، الگوریتم کاری APT برای نصب پکیج به صورت زیر است:

1) نصب نبودن پکیج درخواستی بررسی می‌شود.
2) در صورت نصب نبودن، به فایل‌های شاخص مراجعه کرده و لیست پکیج‌های لازم را استخراج می‌کند.
3) پکیج‌های لازم را از محل تعیین شده در فایل‌های شاخص دریافت می‌کند.
4) پکیج‌ها را به dpkg می‌دهد تا آن‌ها را نصب کند.

قالب sources.list

- هر خط در مورد یک مخزن توضیح می‌دهد و مشخص می‌کند چه بخشی از پکیج‌های آن مخزن باید در نظر گرفته شود. در واقع ممکن است ما فقط بخشی از پکیج‌های یک مخزن را بخواهیم.
- قالب هر خط به صورت زیر است:

```
deb URI DIST [COM1] [COM2] …
```

- URI آدرس مخزن را مشخص می‌کند.
- DIST مشخص می‌کند چه توزیع و دسته‌ای از پکیج‌ها مورد نظر هستند. برخی گزینه‌های مهم عبارتند از:
  - precise: پکیج‌های اصلی توزیع precise
  - precise-updates: پکیج‌های به روز شده برای توزیع precise
  - precise-security: پکیج‌های ابزارهای امنیتی برای توزیع precise
- COM1 و COM2 و ... بخش مورد نظر از مخزن را مشخص می‌کنند. گزینه‌های موجود همان main، restricted، universe و multiverse هستند.

به روز رسانی فایل‌های شاخص

- برای به روز رسانی فایل‌های شاخص از ابزار apt-get در مجموعه APT به صورت زیر استفاده می‌شود.

```
# apt-get update
```

- در صورت موفقیت خروجی شبیه شکل زیر خواهد بود:

```
Ign http://194.225.24.72 precise/universe Translati
Ign http://194.225.24.72 precise/universe Translati
Ign http://194.225.24.72 precise-updates/main Trans
Ign http://194.225.24.72 precise-updates/main Trans
Ign http://194.225.24.72 precise-updates/universe T
Ign http://194.225.24.72 precise-updates/universe T
Fetched 396 B in 0s (667 B/s)
Reading package lists... Done
```

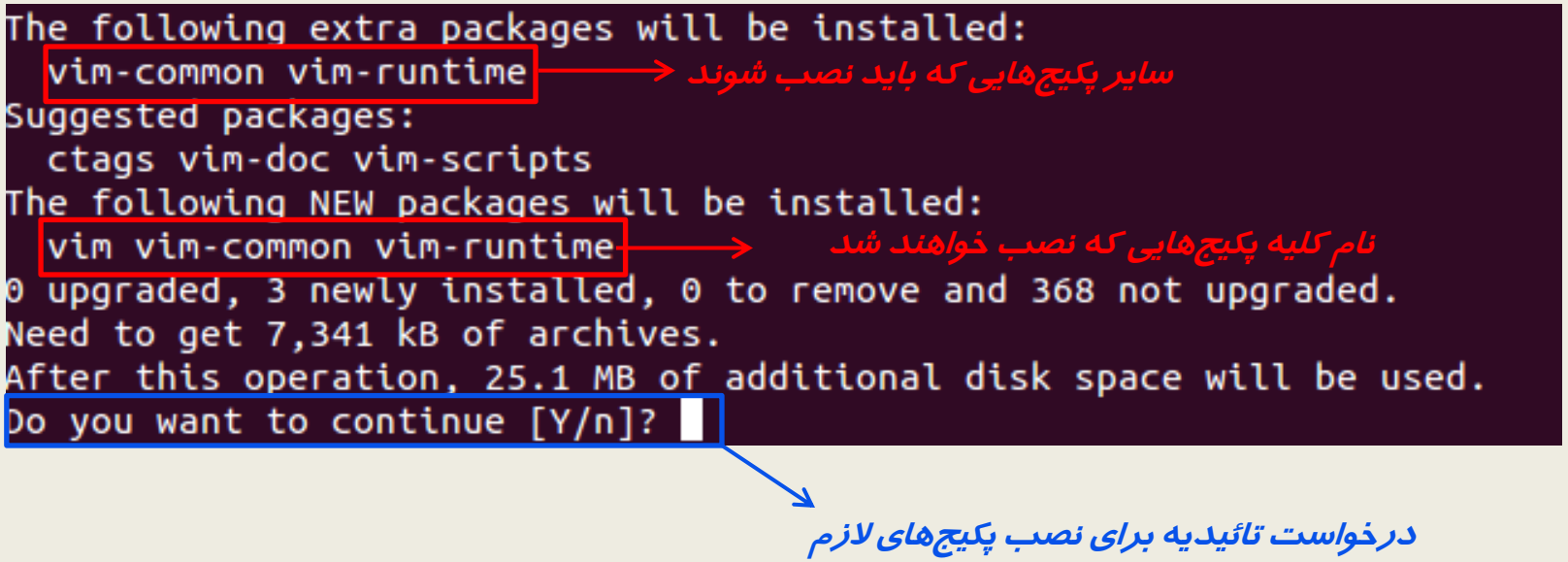
فایل‌های شاخص به درستی ساخته شده‌اند.

نصب پکیج با APT

- برای نصب پکیج NAME کافی است از دستور زیر استفاده کنید:

```
# apt-get install NAME
```

- اگر تمام وابستگی‌ها نصب شده باشند، پکیج مورد نظر نصب خواهد شد وگرنه با اعلام پکیج‌هایی که لازم است نصب شوند، از شما تائیدیه‌ای برای نصب می‌خواهد. با وارد کردن Y و فشردن Enter به نصب ادامه دهید.



حذف پکیج با APT

- برای حذف پکیج NAME از دستور زیر استفاده می‌شود:

```
# apt-get remove NAME
```

- دستور remove در apt-get همانند سوئیچ –r در dpkg به فایل‌های تنظیمات کاری ندارد.
- برای پاک کردن کلیه فایل‌های مرتبط با پکیج از دستور زیر استفاده می‌شود:

```
# apt-get --purge remove NAME
```

- دقت شود که برای نصب پکیجی مثل vim، پکیج‌هایی مانند vim-common و vim-runtime نیز نصب شده‌اند. حال اگر vim را حذف کنیم، دیگر به این دو پکیج نیازی نداریم.
- از دستور زیر پکیج‌هایی که دیگر به آن‌ها نیازی نیست را پاک می‌کند:

```
# apt-get autoremove
```

به‌روز رسانی با APT

- یکی از قابلیت‌های APT تشخیص نسخه‌های جدید پکیج و به‌روز رسانی ساده آن‌ها است.
- کافی است APT از به‌روز رسانی از طریق مخزن مطلع شود و آن را در فایل‌های شاخص خود ثبت کند. سپس با دستور زیر می‌توان پکیج مورد نظر را به آخرین نسخه موجود در مخزن‌ها به‌روز کرد:

```
# apt-get upgrade NAME
```

- در صورتی که بخواهیم تمام پکیج‌ها را به‌روز کنیم از دستور زیر استفاده می‌شود:

```
# apt-get upgrade
```

- دقت داشته باشید که چون قبل به‌روز کردن لازم است فایل‌های شاخص به روز شوند، بهتر است قبل به‌روز رسانی از apt-get update استفاده شود.

مخزن APT

- برای کاهش پهنای باند مصرفی و جلوگیری از بارگزاری‌های مجدد یک پکیج، ابزار APT مخزنی از پکیج‌ها برای خود ایجاد می‌کند.
- در واقع قبل از بارگزاری پکیج از مخزن اینترنتی، وجود پکیج در مخزن APT بررسی می‌شود. در صورت وجود پکیج‌های لازم در مخزن APT، بارگزاری برای پکیج‌های موجود انجام نمی‌شود.
- پس از بارگزاری پکیج‌های لازم در مخزن APT، نصب پکیج‌ها از مخزن APT آغاز می‌شود.
- مخزن APT در مسیر زیر واقع شده است:

```
/var/cache/apt/archives
```

- اگر بخواهیم فقط پکیج‌ها بارگزاری شده و درون مخزن APT قرار گیرد، از دستور زیر استفاده می‌کنیم:

```
# apt-get -d install NAME
```

نصب با استفاده از کد منبع

- تقریباً برای بسیاری از ابزارها پکیج دبیان ساخته شده است؛ اما همواره بسیاری از ابزارهای اساسی و سودمند بدون پکیج دبیان منتشر می‌شوند.
- یکی از راه‌های دیگر نصب ابزار در لینوکس استفاده از کد منبع می‌باشد.
- با در اختیار داشتن کد منبع می‌توان ابزار را کامپایل کرده و فایل‌های مرتبط را در مکان مناسب قرار داد تا استفاده از ابزار میسر شود.
- اما نصب با استفاده از روش یاد شده همانند نصب با استفاده از فایل پکیج آسان نیست زیرا وابستگی‌هایی که قبلاً توسط ابزارهای آماده بررسی می‌شد، اکنون باید توسط کاربر انجام گیرد.
- همچنین فرآیند کامپایل نیز زمان‌بر و پیچیده است؛ زیرا ابزارها معمولاً شامل صدها یا حتی هزاران فایل هستند که باید تک تک کامپایل شده و سپس لینک شوند تا فایل اجرایی تولید شود.

نصب با استفاده از کد منبع (ادامه)

- اگرچه فرآیند نصب با استفاده از کد منبع دشوار است اما ابزارهایی ارائه شده‌اند تا سربار این کار را از دوش کاربر برداشته و به صورت خودکار انجام دهند.
- فایل Configuration و ابزار make دو ابزاری هستند که در کنار هم نصب با استفاده از کد منبع را ساده می‌کنند.
- به جای make از ابزارهای دیگری مثل cmake نیز می‌توان استفاده کرد اما مفاهیم همه ابزارها تقریباً یکی است زیرا همگی کامپایل را خودکار انجام می‌دهند.
- فرآیند نصب با استفاده از کد منبع شامل مراحل زیر است:
  - تولید Makefile
  - کامپایل کد
  - نصب برنامه

Makefile چیست؟

- Makefile یک فایل اسکریپت است که دستورهای داخل آن مشخص می‌کنند چگونه باید کد منبع را کامپایل کرد.
- در واقع Makefile زبان خاصی دارد که ابزار make معنی آن را می‌فهمد.
- Makefile دو دسته است:
  - **مستقل از معماری:** نحوه کامپایل شدن بر روی هر سیستم عامل با هر پردازنده‌ای را مشخص می‌سازد. اگرچه مستقل از معماری است اما واقعاً قابل اجرا نیست؛ زیرا برخی فیلدهای آن خالی گذاشته شده تا بر اساس معماری و سیستم‌عامل مقصد تعیین شود.
  - **وابسته به معماری:** نحوه کامپایل شدن بر روی یک سیستم عامل خاص با پردازنده خاص را مشخص می‌کند. قابل اجرا است اما برای همان معماری.
- در واقع در Makefile مستقل از معماری برخی فیلدها خالی گذاشته می‌شود تا قبل از کامپایل با توجه به معماری و سیستم‌عامل مقصد مقدار این فیلدها تعیین شده و Makefile وابسته به معماری تولید شود.

Makefile چیست؟ (ادامه)

- Makefile مستقل از معماری معمولاً با نام Makefile.In و نسخه وابسته به معماری آن با نام Makefile مشاهده می‌شود.
- معمولاً Makefile توسط ابزارهای آماده مثل automake یا IDEهای پیشرفته مثل Eclipse به صورت خودکار تولید می‌شود که از حوصله این درس خارج است.
- اگرچه انسان هم می‌تواند Makefile را ایجاد کند اما برای برنامه‌های متوسط و بزرگ ممکن نیست.

فایل پیکره‌بندی

- فایل «پیکره‌بندی» یا Configuration یک اسکریپت خط فرمان لینوکس (Bash Script) است که دو وظیفه مهم را برعهده دارد:
  - بررسی وابستگی‌ها و وجود آن‌ها
  - کشف مسیرهای مورد نیاز و قرار دادن در Makefile مستقل از معماری به منظور تولید Makefile وابسته به معماری
- فایل پیکره‌بندی بررسی می‌کند کتابخانه‌ها، سرآیندها و سایر ابزار مورد نیاز برای کامپایل و لینک کردن کد منبع وجود دارند یا نه.
- همچنین مسیرهای کتابخانه‌ها، سرآیندها و ابزارها را یافته و درون Makefile مستقل از معماری قرار می‌دهد تا Makefile وابسته به معماری تولید شود.

تولید Makefile

- فرآیند تولید Makefile (وابسته به معماری) با اجرای فایل پیکره‌بندی به صورت زیر تولید می‌شود:

```
# ./configure
```

- برخی ابزارها فایل پیکره‌بندی ندارند و از قبل Makefile وابسته به معماری آن‌ها تولید شده و همراه با کد منبع منتشر می‌گردد.
- باید خروجی اجرای فایل پیکره‌بندی را به دقت مشاهده کرد زیرا خطاهای آن باید مرتفع گردند و گرنه انجام گام بعدی ممکن نیست.
- عدم وجود برخی پکیج‌ها که در خروجی فایل پیکره‌بندی اعلام می‌شود به معنی خطا نیست، در صورت خطا پیام واضحی (معمولاً چند خطی) تولید می‌شود که به معنی خطا است.

کامپایل کد، نصب برنامه و حذف آن

- پس از تولید Makefile نوبت به کامپایل کد می‌رسد. این کار به سادگی با استفاده از دستور زیر انجام می‌گیرد:

```
# make
```

- اگر خطایی رخ ندهد، با اجرای دستور زیر فایل‌های لازم به مکان مناسب انتقال یافته و برنامه نصب شده محسوب می‌شود وگرنه باید خطاها را رفع کرده و دوباره make را اجرا کنیم:

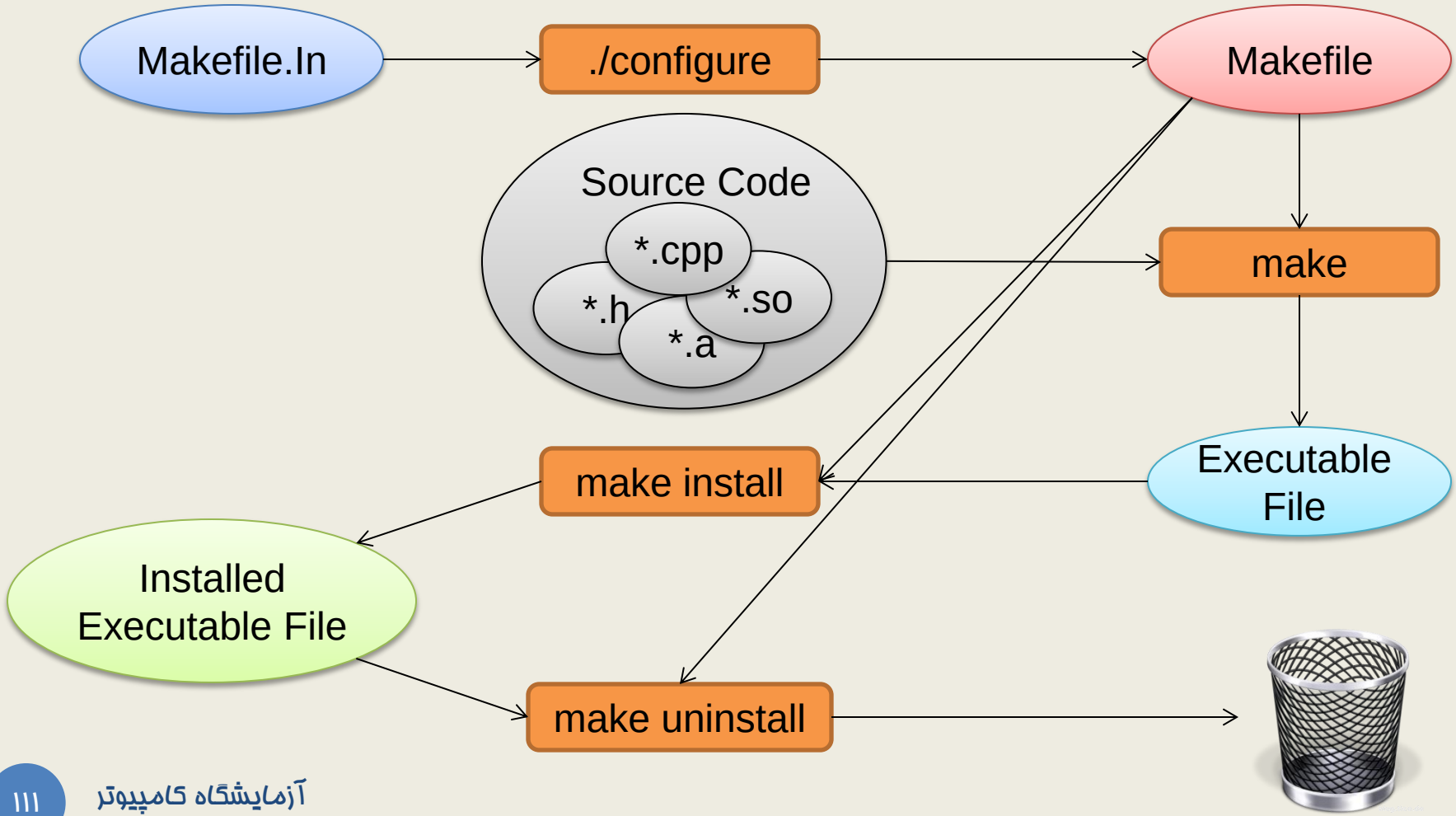
```
# make install
```

- دقت شود که برنامه نصب شده توسط این روش مسیر دیگری را طی کرده است که آن مسیر از dpkg عبور نمی‌کند؛ لذا dpkg و ابزارهای وابسته در مورد نصب برنامه‌هایی که با کد منبع نصب می‌شوند، خبر ندارد.
- با توجه به نکته بالا، برای پاک کردن برنامه نصب شده به این روش نمی‌توان از dpkg یا ابزارهای وابسته استفاده کرد. بدین منظور از دستور زیر استفاده می‌کنیم:

```
# make uninstall
```

- برخی کدهای منتشر شده این قابلیت را ندارند و حذف آن‌ها به سختی و دستی خواهد بود.

خلاصه فرآیند نصب با استفاده از کد منبع



فرآیند چیست؟

- «فرآیند» (Process) دارای تعریف دقیقی است که در درس سیستم‌عامل با آن آشنا می‌شوید اما به صورت ساده می‌توان گفت با *هر بار اجرای* یک برنامه، یک فرآیند ایجاد می‌شود.
- دقت شود که هر بار اجرای یک برنامه، فرآیندی را ایجاد می‌کند که از سایر اجراهای برنامه مستقل است.
- در واقع هر فرآیند نماینده‌ای از یک برنامه در حال اجرا است.
- برای پایان دادن به اجرای یک برنامه، باید به فرآیند مرتبط با آن پایان دهیم.
- پایان دادن به فرآیند را در اصطلاح «کشتن» (Kill) یا «بستن» فرآیند گویند.

PID چیست؟

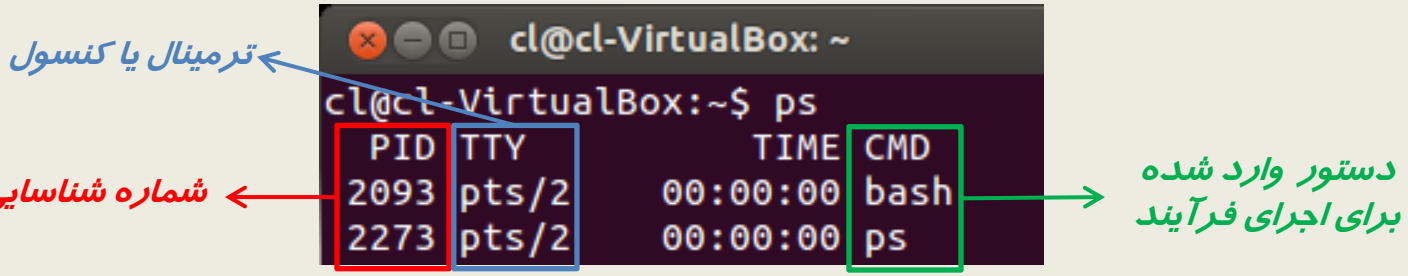
- به هر فرآیند در حال اجرا یک «شماره شناسایی فرآیند» یا (Process ID) اختصاص می‌یابد.
- این شماره شناسایی یک عدد صحیح یکتا بوده و دستگیره‌ای برای دسترسی به فرآیند در آینده خواهد بود.
- PID توسط سیستم عامل به صورت تصادفی تعیین می‌شود و برنامه‌ها نمی‌توانند در تعیین این عدد نقشی ایفا کنند.
- ممکن است پس از بسته شدن یک فرآیند، فرآیند دیگری اجرا شود که دارای همان PID است. این موضوع تناقضی با یکتا بودن PID ندارد زیرا PID در هر لحظه از زمان یکتا است نه در هر فاصله زمانی.

مشاهده لیست فرآیندها

- ابزار ps برای مشاهده لیست فرآیندها استفاده می‌شود.
- برای مشاهده لیست فرآیندها از دستور زیر استفاده می‌کنیم:

```
$ ps
```

- خروجی شبیه شکل زیر خواهد بود:



- ستون اول همان PID را نشان می‌دهد.
- ستون دوم کنسول یا ترمینالی که فرآیند از طریق آن اجرا شده و به آن تعلق دارد را مشخص می‌کند.
- ستون چهارم دستوری را مشخص می‌کند که باعث اجرای فرآیند شده است. نام برنامه در این قسمت قرار دارد.

مشاهده لیست همه فرآیندها

- وقتی دستور ps به تنهایی اجرا شود، فقط لیست فرآیندهایی را نشان می‌هد که متعلق به ترمینال یا کنسول است.
- برای نمایش لیست همه فرآیندها از دستور زیر استفاده می‌کنیم.

```
$ ps aux
```

- ستون‌های مهم خروجی آن عبارتند از:
  - ستون اول کاربری که فرآیند تحت آن اجرا می‌شود را نشان می‌دهد. در واقع فرآیند همان سطح دسترسی را دارد که کاربر این ستون دارد.
  - ستون دوم همان PID است.
  - ستون ششم کنسول یا ترمینالی که فرآیند به آن تعلق دارد را نشان می‌هد. علامت سوال نشان‌گر اجرا توسط GUI یا سیستم‌عامل می‌باشد.
  - ستون آخر نشانگر دستوری است که برای اجرای فرآیند وارد شده است. در این حالت آدرس مطلق برنامه نمایش داده می‌شود. همچنین سوئیچ‌ها و آرگومان‌های وارد شده نیز وجود دارند.

بستن فرآیند

- معمولاً فرآیندها ایجاد و پس از مدتی به دلیل اتمام کار برنامه، بسته می‌شوند.
- اگر فرآیندی به طور خودکار بسته نشود یا حتی در حین اجرای فرآیند بخواهیم آن را ببندیم، از ابزار kill استفاده می‌کنیم.
- با داشتن PID فرآیند از دستور زیر برای بستن آن استفاده می‌کنیم:

```
# kill PID
```

kill و سیگنال

- ابزار kill فقط برای بستن فرآیندها نیست؛ بلکه ابزاری برای ارسال سیگنال به فرآیندها است.
- در واقع kill می‌تواند سیگنالی را برای فرآیند ارسال کند و فرآیند را از اتفاقی با خبر سازد. فرآیند نیز بنا به نوع سیگنال، نسبت به آن واکنش نشان می‌دهد.
- برای مثال وقتی kill PID اجرا می‌شود، ابزار kill سیگنالی موسوم به SIGTERM برای فرآیندی که با شماره شناسایی PID ارسال می‌کند. چون معنی این سیگنال درخواست بسته شدن است، فرآیند در صورت صلاح دید به کار خود پایان داده و بسته می‌شود.
- سیگنال‌های دیگری نیز وجود دارند که هر یک کاربرد خاص خود را دارند.
- برخی سیگنال‌ها برای کرنل در مورد یک فرآیند ارسال می‌شود نه فرآیند تعیین شده.

سیگنال SIGKILL

- سیگنال SIGTERM به فرآیند ارسال شده و پیشنهاد می‌دهد که فرآیند خاتمه یابد.
- اما برخی فرآیندها در برابر بسته شدن مقاومت می‌کنند (برخی برنامه‌ها طوری نوشته شده‌اند که نمی‌خواهند بسته شوند؛ مثل ویروس‌ها)
- از آنجایی که سیستم عامل مدیر منابع است پس اختیار آن‌ها را دارد و می‌تواند فرآیندی را از آن محروم سازد.
- پردازنده یکی از منابع است، پس سیستم عامل می‌تواند فرآیندی را از پردازنده محروم سازد، در واقع آن را بیرون انداخته و فرآیند را ببندد بدون اینکه فرآیند بتواند در این خصوص تصمیم بگیرد.
- سیگنال SIGKILL برخلاف SIGTERM به کرنل می‌گوید که فرآیند تعیین شده را ببند. از آنجایی که کرنل این کار را انجام می‌دهد، فرآیند نمی‌تواند در برابر آن مقاومت کند.
- برای ارسال سیگنال SIGKILL در مورد فرآیند PID از دستور زیر استفاده می‌کنیم:

```
# kill -9 PID
```

بستن فرآیند با نام

- دستور kill نیاز به شماره شناسایی فرآیند داشت.
- بدست آوردن شماره شناسایی فرآیند دشوار است.
- ابزار killall برای بستن تمام فرآیندهایی است که دارای نام تعیین شده می‌باشند.
- برای بستن تمام فرآیندهایی که دارای نام NAME هستند از دستور زیر استفاده می‌شود:

```
# killall NAME
```

- برای مثال برای بستن تمام vimهای باز شده از دستور زیر استفاده می‌کنیم:

```
# killall vim
```

- دقت کنید که killall بین تمام فرآیندهای با نام مورد نظر را می‌بندد در حالیکه kill تنها یک فرآیند را می‌بندد. دلیل این امر امکان وجود چند فرآیند با یک نام است.

لیست فایل‌های فرآیند

- وقتی برنامه‌ای اجرا می‌شود، حتماً در طول اجرای خود با فایل‌هایی کار خواهد کرد.
- پس هر فرآیند نیز در هر لحظه با تعدادی فایل در ارتباط است.
- برای بدست آوردن لیست فایل‌هایی که یک فرآیند در حال کار با آن‌ها است از دستور زیر استفاده می‌کنیم. در این دستور PID همان شماره شناسایی فرآیند است:

```
# lsof -p PID
```

- ستون‌های مهم خروجی عبارتند از:
  - ستون اول نام فرآیند را مشخص می‌کند.
  - ستون دوم PID فرآیند را مشخص می‌کند.
  - ستون سوم نام کاربری که فرآیند متعلق به آن است را مشخص می‌کند.
  - ستون آخر مسیر فایل را مشخص می‌کند.

لیست فرآیندهای یک فایل

- گاهی اوقات لازم داریم لیست فرآیندهایی را که با یک فایل در حال کار هستند را بدست آوریم.
- برای یافتن فرآیندهایی که در حال کار با فایل FILE هستند از دستور زیر استفاده می‌کنیم:

```
# lsof FILE
```

- FILE می‌تواند آدرس مطلق یا نسبی فایل باشد.
- اگر FILE آدرس یک پوشه باشد، تمام فرآیندهایی که در حال کار با این پوشه یا فایل‌ها یا پوشه‌های زیرمجموعه‌های آن هستند را نمایش می‌دهد.
- خروجی این دستور تقریباً مثل lsof –p است.

فایل بایگانی چیست؟

- «فایل بایگانی» (Archive File) یک فایل است که مجموعه‌ای از فایل‌های دیگر به همراه اطلاعاتی در مورد آن‌ها را در خود جای داده است.
- کاربرد مهم فایل‌های بایگانی انتقال چند فایل در قالب یک فایل است. در واقع شما برای انتقال صدها یا هزاران فایل، تنها یک فایل را جابه‌جا می‌کنید.
- به فرآیند ساخت فایل‌های درون فایل بایگانی، «استخراج» (Extract) گویند.
- فرمت‌های بسیاری برای فایل بایگانی وجود دارند که معروف‌ترین آن‌ها عبارتند از:
  - Rar
  - Zip
  - Tar خانواده
  - Disc Images (مثل iso و nrg و ...)

مزایای فایل بایگانی

- مزیت عمده فایل‌های بایگانی عبارتند از:
  - **سهولت جابه‌جایی:** کافی است یک فایل را جابه‌جا یا کپی کنید؛ زیرا همان فایل حاوی همه فایل‌های مورد نیاز شما است.
  - **حفظ خواص:** برخی فرمت‌های فایل بایگانی، برخی خواص فایل‌های درون خود قبل از بایگانی شدن را ذخیره می‌کنند تا در آینده قابل استفاده باشند.
  - **امکان فشرده‌سازی:** اکثر فرمت‌ها امکان فشرده‌سازی را می‌دهند.
  - **امکان تعیین رمز عبور:** می‌توان برای دسترسی به فایل‌های درون یک فایل بایگانی رمز عبور تعیین کرد.
  - **تکه‌تکه کردن:** برخی فرمت‌ها مثل Rar امکان تکه تکه کردن فایل را دارند. برای مثال یک فایل یک گیگابایتی را به ۱۰ فایل ۱۰۰ مگابایتی می‌شکنند تا انتقال آن ساده‌تر باشد.

فرمت tar.gz

- معروف‌ترین فرمت فایل بایگانی در لینوکس است.
- بیشتر کدهای منبع در قالب این فایل منتشر می‌شوند.
- برای ساخت فایل tar.gz از دستور زیر استفاده می‌کنیم:

```
# tar czvf DST.tar.gz SRC
```

  - DST نام فایل tar.gz است که می‌خواهیم بسازیم.
  - SRC مشخص می‌کند چه فایل‌ها و پوشه‌هایی باید بایگانی شوند. اگر SRC آدرس یک یا چند فایل باشد، فقط همان فایل(ها) بایگانی می‌شود؛ اما اگر آدرس پوشه باشد، آن پوشه و تمام زیر مجموعه‌های آن بایگانی می‌شوند.
- برای استخراج فایل tar.gz از دستور زیر استفاده می‌کنیم:

```
# tar xzvf FILE
```

که FILE آدرس (مطلق یا نسبی) فایل بایگانی را نشان می‌دهد.

فرمت Zip

- یکی دیگر از فرمت‌های معروف فایل بایگانی چه در ویندوز و چه در لینوکس است.
- برای ساخت فایل بایگانی از دستور زیر استفاده می‌شود:

```
# zip DST SRC1 SRC2 SRC3 …
```

  - DST نام فایل zip است که می‌خواهیم ساخته شود.
  - SRCها آدرس فایل‌های که می‌خواهیم درون این فایل بایگانی قرار گیرد.
- برای مثال دستور زیر دو فایل pass.txt و user.txt را درون فایلی با نام user_info.zip بایگانی می‌کند:

```
# zip user_info.zip user.txt pass.txt
```

- برای استخراج فایل‌های درون فایل بایگانی از دستور زیر استفاده می‌شود:

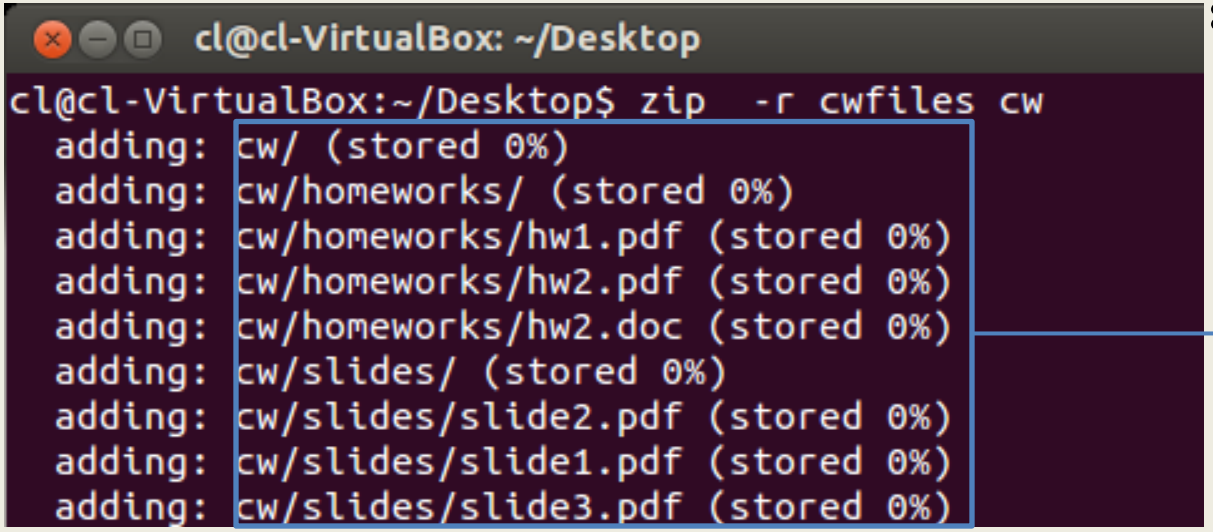
```
# unzip FILE
```

فرمت Zip (ادامه)

- اگر پوشه‌ای برای بایگانی شدن انتخاب شود، در حالت عادی تنها خود پوشه بایگانی می‌شود و نه محتویات آن.
- برای شامل شدن محتویات پوشه از سوئیچ معروف r– استفاده می‌کنیم:

```
# zip -r DST SRC1 SRC2 SRC3 …
```

- شکل زیر خروجی حاصل از بایگانی کردن محتویات یک پوشه را نشان می‌دهد:



دیسک و نام‌گذاری آن

- همانطور که قبلاً گفته شد هر دستگاهی در لینوکس دارای یک شبه‌فایل در مسیر /dev/ می‌باشد. دیسک‌های متصل به کامپیوتر نیز دارای شبه‌فایل‌هایی می‌باشند.
- دیسکی که از فناوری SATA بهره می‌برد شبه فایلی با نام sdX خواهد داشت. به جای X اندیس دیسک قرار می‌گیرد که می‌تواند یکی از حروف کوچک الفبای انگلیسی باشد. برای مثال /dev/sda به دیسک اول اشاره می‌کند.
- مسلماً هر دیسکی حاوی یک یا چند پارتیشن است. پارتیشن‌ها نیز دارای شبه‌فایل sdXY هستند که X همان اندیس دیسک و Y اندیس پارتیشن است. برای مثال /dev/sdb3 به پارتیشن سوم از دیسک دوم اشاره می‌کند.
- اندیس پارتیشن بر حسب قرارگیری فیزیکی پارتیشن بر روی دیسک سنجیده می‌شود.

نمایش اطلاعات دیسک

- برای کار با دیسک از ابزار گرافیکی Gparted یا ابزار خط فرمان fdisk می توان استفاده کرد.
- دستور fdisk با سوئیچ l– تمام دیسک‌ها و پارتیشن‌ها آن‌ها را لیست می‌کند.
- شکل زیر خروجی این دستور را نشان می‌دهد. همانطور که مشاهده می‌کنید دیسک sda دارای سه پارتیشن با نام‌های sda1 و sda2 و sda5 می‌باشد. دقت کنید که پارتیشن Extended خود یک پارتیشن است که حاوی یک سری پارتیشن منطقی است.

```
root@cl-VirtualBox:~# fdisk -l

Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders, total 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 identifier: 0x000d3ec8

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048    39845887    19921920   83  Linux
/dev/sda2        39847934    41940991     1046529    5  Extended
/dev/sda5        39847936    41940991     1046528   82  Linux swap / Solaris
```

اطلاعات دیسک اول نمایش داده می‌شود

سه پارتیشن بر روی این دیسک موجود است

فرمت کردن پارتیشن

- یکی از اعمال مورد نیاز در کار با دیسک، فرمت کردن یک پارتیشن است.
- برای بیشتر فرمت‌های معروف، ابزار خط فرمان وجود دارد. قالب همه این ابزارها به صورت زیر است:

```
# FT PARTITION
```

FT نام ابزار فرمت کردن است و PARTITION آدرس شبه‌فایل پارتیشن مورد نظر.

- برای مثال در دستورهای پارتیشن دوم از دیسک اول را به فرمت‌های NTFS و FAT و EXT4 و EXT3 فرمت می‌کنیم:

```
# mkfs.ntfs /dev/sda2
# mkfs.vfat /dev/sda2
# mkfs.ext4 /dev/sda2
# mkfs.ext3 /dev/sda2
```

اتصال دستگاه‌های ذخیره‌سازی

- یک قانون ساده در مورد لینوکس وجود دارد و این است که هر دستگاه ذخیره‌سازی (مثل Hard Disk، CD Rom، Flash Memory و ...) برای استفاده در لینوکس ابتدا باید به نقطه‌ای از ساختار فایلی لینوکس متصل شود.
- همانطور که در لینوکس مقدماتی دیدیم، به این نقطه، نقطه اتصال (Mount Point) و به این عمل، اتصال (Mount) گوئیم.
- هر عملی که بر روی نقطه اتصال انجام گیرد، بر روی دستگاه متصل به آن نقطه انجام می‌گیرد.
- قالب دستور اتصال به صورت زیر است:

```
# mount -t FS DEV MP
```

FS نام سیستم فایل دستگاه مورد نظر است. DEV آدرس شبه‌فایل دستگاه است و MP نقطه اتصال است. نقطه اتصال باید یک پوشه باشد.

نام ابزار فرمت کردن است و PARTITION آدرس شبه‌فایل پارتیشن مورد نظر

اتصال دستگاه‌های ذخیره‌سازی (ادامه)

- برای مثال فرض کنید دیسک جدیدی با نام sdb به کامپیوتر اضافه کرده‌ایم و قصد پارتیشن اول آن را به /media/newdisk لینوکس متصل کنیم تا بتوانیم فایل‌های درون آن را بخوانیم. بدین منظور از دستور زیر استفاده می‌شود:

```
# mount -t ext4 /dev/sdb1 /media/newdisk
```

- تصویر زیر نشان می‌دهد که پوشه newdisk را ساخته‌ایم و ابتدا محتویات آن را نمایش می‌دهیم. سپس دستور بالا را اجرا کرده و دوباره محتویات را نمایش می‌دهیم. فایل‌های داخل پارتیشن مورد نظر نشان داده می‌شوند.

```
root@cl-VirtualBox:/media# mkdir newdisk
root@cl-VirtualBox:/media# cd newdisk
root@cl-VirtualBox:/media/newdisk# ls
root@cl-VirtualBox:/media/newdisk# cd ..
root@cl-VirtualBox:/media# mount -t ext4 /dev/sdb1 /media/newdisk/
root@cl-VirtualBox:/media# cd newdisk/
root@cl-VirtualBox:/media/newdisk# ls
lost+found  scores.txt  students.txt
```

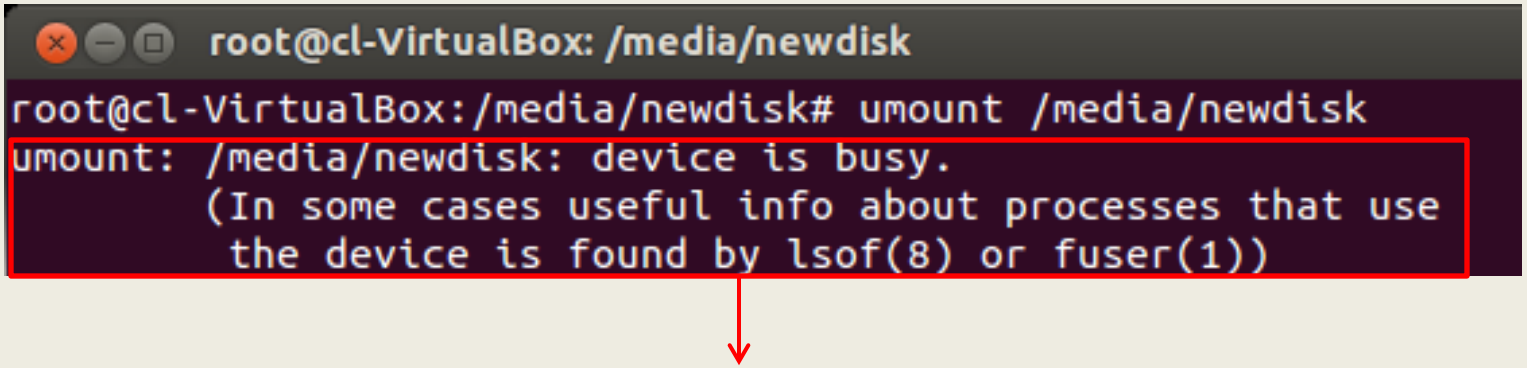
فایل‌های درون پارتیشن متصل شده

قطع اتصال

- برای قطع اتصال باید نقطه اتصال را داشته باشیم و با دستور umount اقدام به این کار نمائیم.
- برای مثال دستور زیر دستگاه متصل به /media/newdisk را قطع می‌کند.

```
# umount /media/newdisk
```

- باید دقت داشت که قبل از قطع یک دستگاه باید ارتباط هر فرآیندی با فایل‌های درون آن را قطع کرد وگرنه اجازه قطع ارتباط داده نمی‌شود. برای مثال تصویر زیر پیام خطای گفته شده را نشان می‌دهد.



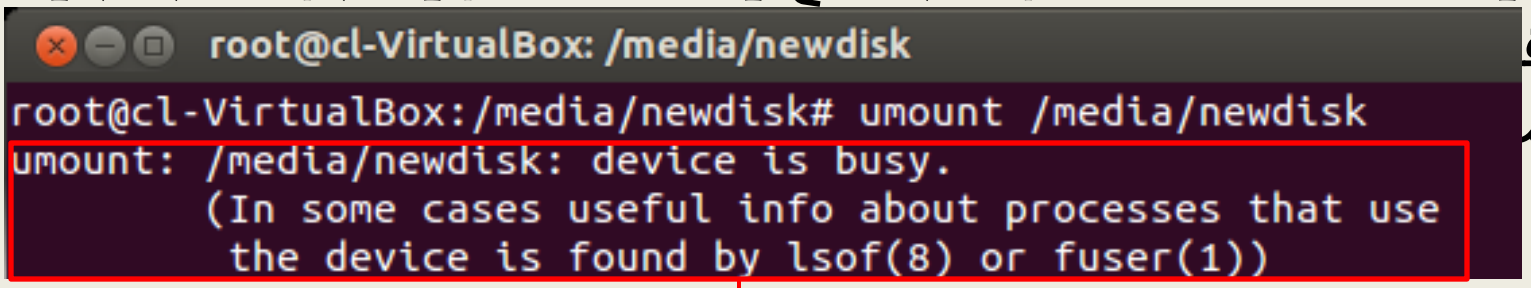
خطا در قطع ارتباط به دلیل استفاده از دستگاه مورد نظر

تنظیمات اتصال

- گاهی اوقات لازم است اتصال به گونه خاصی انجام گیرد. برای مثال اگر بخواهیم پارتیشنی فقط قابل خواندن باشد، باید از تنظیمات استفاده کرد.
- تنظیمات در جلوی سوئیچ o– قرار می‌گیرند و با کاما نیز جدا می‌شوند.
- برای مثال اگر بخواهیم دستگاهی را فقط خواندنی متصل کنیم، از ro استفاده می‌کنیم:

```
# mount -t ntfs -o ro /dev/sdb1 /media/newdisk
```

- باید دقت داشت که قبل از قطع یک دستگاه باید ارتباط هر فرآیندی با
ف
برای
م



خطا در قطع ارتباط به دلیل استفاده از دستگاه مورد نظر

اتصال فایل ISO

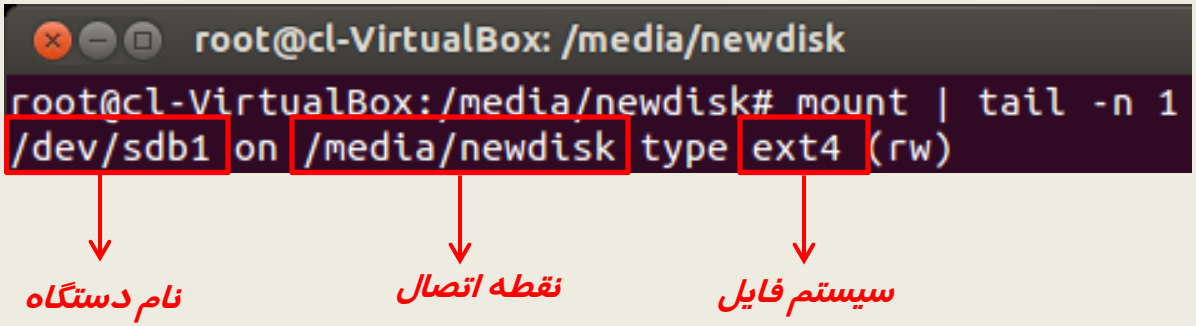
- همانطور که گفته شد یکی از فایل‌های بایگانی معروف، فایل ISO می‌باشد.
- فایل ISO یک استاندارد برای بایگانی کردن کل CD یا DVD می‌باشد.
- نرم‌افزارهای بسیاری برای ساختن درایو مجازی و قرار دادن فایل‌های ISO درون آن در سیستم عامل ویندوز تولید شده‌اند؛ مثل نرم‌افزار DaemonTools.
- اما این کار از قابلیت‌های داخلی لینوکس است که توسط دستور mount انجام می‌گیرد. در واقع فایل ISO نیز یک دستگاه ذخیره‌سازی مجازی است.
- قالب دستور به صورت زیر است:

```
# mount -o loop ISO_PATH MP
```

که ISO_PATH مسیر فایل ISO و MP نقطه اتصال مورد نظر است.

نمایش لیست اتصال‌ها

- گاهی اوقات لازم است اینکه دستگاهی به کجا متصل شده یا به یک نقطه اتصال چه دستگاهی متصل شده است را بدست آوریم. این کار توسط دستور mount بدون هیچ پارامتری انجام می‌گیرد.
- هر خط خروجی مربوط به یک اتصال است. برای مثال خط مربوط به مثال‌های قبلی به صورت زیر است:



جانشینی دستور

- «جانشینی دستور» (Command Substitution) یعنی قرار دادن خروجی یک دستور به عنوان پارامتر دستور دیگر. دستوری که باید خروجی آن محاسبه و مقدارش جانشین شود در $() قرار می‌گیرد.
- دقت کنید که جانشینی دستور با Pipe متفاوت است. در Pipe خروجی یک دستور به ورودی دستور دیگر منتقل می‌شود در حالیکه در این حالت خروجی یک دستور به عنوان پارامتر دستور دیگر قرار می‌گیرد.
- برای مثال دستور زیر یک عدد تصادفی در بازه ۱ تا ۱۰۰۰ تولید می‌کند:

```
# shuf -n 1 -i 1-1000
```

- می‌خواهیم پوشه‌ای بسازیم که نام آن یک عدد تصادفی در بازه ۱ تا ۱۰۰۰ باشد. از دستور زیر استفاده می‌کنیم:

```
# mkdir $(shuf -n 1 -i 1-1000)
```

- دستور زیر نیز سبب ایجاد فایلی با نام stdXXX.txt می‌شود:

```
# touch std$(shuf -n 1 -i 1-1000).txt
```

جانشینی دستور

- «جانشینی دستور» (Command Substitution) یعنی قرار دادن خروجی یک دستور به عنوان پارامتر دستور دیگر. دستوری که باید خروجی آن محاسبه و مقدارش جانشین شود در $() قرار می‌گیرد.
- دقت کنید که جانشینی دستور با Pipe متفاوت است. در Pipe خروجی یک دستور به ورودی دستور دیگر منتقل می‌شود در حالیکه در این حالت خروجی یک دستور به عنوان پارامتر دستور دیگر قرار می‌گیرد.
- برای مثال دستور زیر یک عدد تصادفی در بازه ۱ تا ۱۰۰۰ تولید می‌کند:

```
# shuf -n 1 -i 1-1000
```

- می‌خواهیم پوشه‌ای بسازیم که نام آن یک عدد تصادفی در بازه ۱ تا ۱۰۰۰ باشد. از دستور زیر استفاده می‌کنیم:

```
# mkdir $(shuf -n 1 -i 1-1000)
```

- دستور زیر نیز سبب ایجاد فایلی با نام stdXXX.txt می‌شود:

```
# touch std$(shuf -n 1 -i 1-1000).txt
```

پایان