Pazhoheshi.ir | پژوهشی دات آی آر

بررسی کارکرد پروتکل های مورد استفاده در سیستم عامل ها از دید امنیت

آموزش فارسی لینوکس : قسمت چهارم

یکشنبه, ۱۵ تیر ۱۳۹۳، ۱۲:۱۷ ق.ظ

104.6: Create and Change Hard and Symblic Link

Description : Candidates should be able to create and manage hard and symbolic links to a file

Key Knowledge Areas:

Create links.
Identify hard and/or soft links.
Copying versus linking files.
Use links to support system administration tasks.

The following is a partial list of the used files, terms and utilities:

ln

همان طور که در ویندوز راه میانبری به نام shortcut داریم , در لینوکس هم راه های میانبری برای دسترسی به یک فایل داریم که به دو دسته ی زیر تقسیم می شوند :

1. Symbolic link

2. Hard link

Creating symbolic link ...

تقریبا مفاهیمی که برای shortcut ها در ویندوز داریم , این جا در symbolic link نیز داریم به این صورت که این فایل ها (symlink ها) فایلی هایی هستند که یک آدرس درون خود دارند و متصل هستند به فایل اصلی در مبدا . طبیعتا اگر فایل اصلی (در مبدا) از بین روند , این symbolic link ها دیگر ارزشی ندارند .

یا می توان به این صورت تعریف کرد : Symbolic link ها در صورتی ارزشمند هستند که آدرس درون آن ها موجود باشد و به فایل یا دایرکتوری خاصی اشاره کنند .
توجه داشته باشید که Symbolic link ها هم فایل ها و هم دایرکتوری ها را پوشش می دهند .

برای ساخت Symbolic link ها و یا Hard link ها از دستور ln استفاده می کنیم با این تفاوت که سویچ s- مشخصه ی ساخت Symbolic لینک است . به عنوان مثال :

```
$echo test symlink > file
$ln -s file file.sym.ln
```

نکته : پرمیشن پیش فرض تمامی Symbolic link ها 777 می باشد . اما این به آن معنا نیست که می توان به هر فایلی که پرمیشن لازم را نداشته باشد دسترسی داشته باشیم ! به عنوان مثال :

```
$echo test symlink > file
$chmod 000 file
$ln -s file file.ln
$ls -l
```

اگر خروجی دستور بالا را در Standard output (که در این جا همان ترمینال است) نگاه کنید , خواهید دید که پرمیشن فایل Symbolic link بالا ترین پرمیشن یعنی 777 و پرمیشن فایل اصلی 000 است . حال اگر فایل Symbolic link را cat کنید خواهید دید که با اروری مبنی بر نداشتن پرمیشن مواجه خواهید شد .

طبقه بندی موضوعی

- مقاله آموزشی(۵)
- کتاب آموزشی(۱)
- فیلم آموزشی(۶)
- برنامه ی اختصاصی(۶)
- ارسال بلاگ(۵)
- آموزش فارسی لینوکس(۲)

خلاصه آمار

مجموع نمایش‌ها	۹۲۳۹
مجموع بازدیدکننده‌ها	۳۵۰۵
بازدیدکننده‌های امروز	۴۱
نمایش‌های دیروز	۷۴
مجموع مطالب	۲۲
مجموع نظرات	۳۵
عمر سایت	۹۰ روز
حاضرین در سایت	۶

بایگانی

- تیر ۱۳۹۳(۴)
- خرداد ۱۳۹۳(۲)
- اردیبهشت ۱۳۹۳(۵)
- فروردین ۱۳۹۳(۱۱)

آخرین مطالب

- آموزش فارسی لینوکس : قسمت چهارم
- آموزش فارسی لینوکس : قسمت سوم
- ارسال بلاگ : WPA / WPA2 Rainbow Tables
- فیلم آموزشی : استخراج فایل های صوتی و تصویری از پکت های کپچر شده
- برنامه ی اختصاصی : Mini Wireless IDS
- آپدیت شماره 3 : برنامه ای برای باگ CVE-2014-0160
- آپدیت شماره 2 : خون ریزی قلبی
- آپدیت شماره 1 : برنامه ی Rioter
- برنامه ی اختصاصی : معرفی برنامه ی Rioter
- فیلم آموزشی : 13 فیلم از مباحث مختلف

پیوندها

- پلیس فضای تولید و تبادل اطلاعات نیروی انتظامی جمهوری اسلامی ایران
- وبلاگ تخصصی تیم امنیتی سرزمین امن
- آریا دیتا سنتر
- تیم امنیتی هکران کلاه سیاه ایران - آموزش امنیت و راه های مقابله با هک
- TBH | Forums | Turk Black Hat
- وب سایت شخصی مسلم حقیقیان
- اشتراک هک و امنیت

```
$cat file.ln
```

Inode : در لینوکس هر فایل و دایرکتوری یک عدد منحصر به فرد و یا unique code دارد که مشخصه ی سایز , سطح دسترسی (پرمیشن) , سازنده , گروه سازنده , تاریخ تغییر و دسترسی به فایل و ... است پس می توان گفت که Inode یک ساختمان داده است .

تمامی این اعداد در inode table ذخیره می شود این جدول در زمانی که فایل سیستم پارتیشن را مشخص کردیم ساخته می شود . نکته ی حایز اهمیت این جاست که این جدول متناهی است یعنی تعداد محدودی جای خالی برای این اعداد در نظر گرفته شده است مثلا 513176 و طبیعتا اگر تعداد فایل هایی که بر روی یک پارتیشن داریم برابر با تعداد خانه های این جدول باشد , عمل پر شدن پارتیشن رخ می دهد ! با این که ممکن است هنوز فضای خالی درون دیسک وجود داشته باشد .

پس می توان نتیجه گرفت شرط پر شدن یک پارتیشن برابر است با :

الف) پر شدن ظرفیت پارتیشن

ب) پر شدن ظرفیت جدول Inode

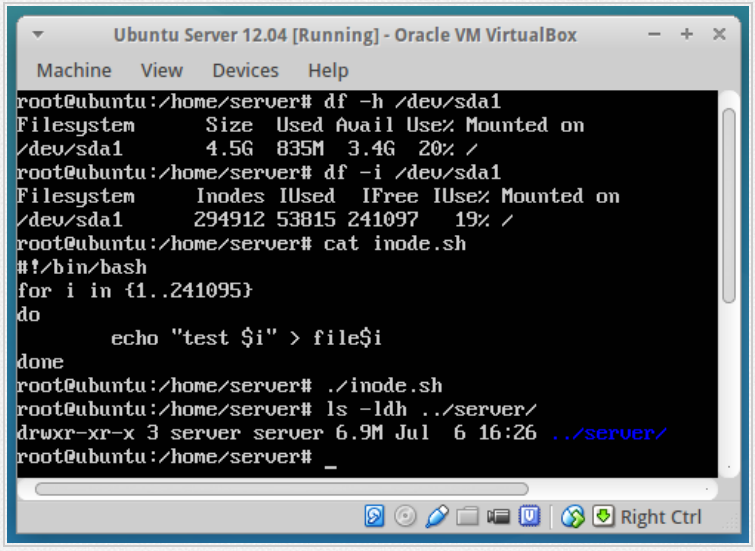
برای مشاهده ی ظرفیت جدول Inode پارتیشن می توان از دستور زیر استفاده کرد :

```
$df -i
```

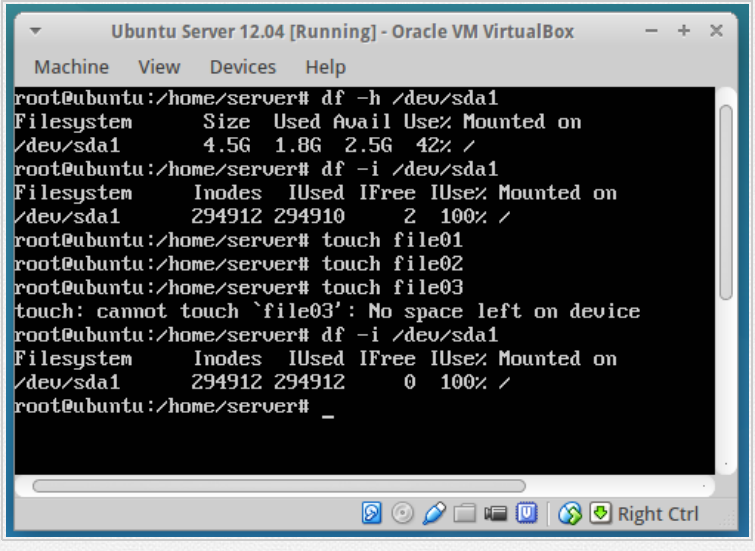
اگر یک اسکریپت کوچیک مثل اسکریپت زیر رو بنویسیم می تونیم پر شدن پارتیشن رو قبل از تکمیل ظرفیتش مشاهده کنیم! مثال :

```
#!/bin/bash
for i in {1..241096}
do
   echo "Welcome $i times" > file$i
done
```

قبل از اجرای Script



بعد از اجرای Script



برگردیم به Symbolic link . اگر به عـدد Inode فایل اصـلی و فایل Symbolic link نگاه کنیـد خواهیـد دید که این دو عدد با هم متفاوت هسـتند . در واقع دلیل این تفاوت این است که این دو فایل کاملا مجزا و جـدای از هم می باشـند , با این تفاوت که در فایل Symbolic link آدرس یک فایل به عنوان مبدا درج شـده است .

مثال :

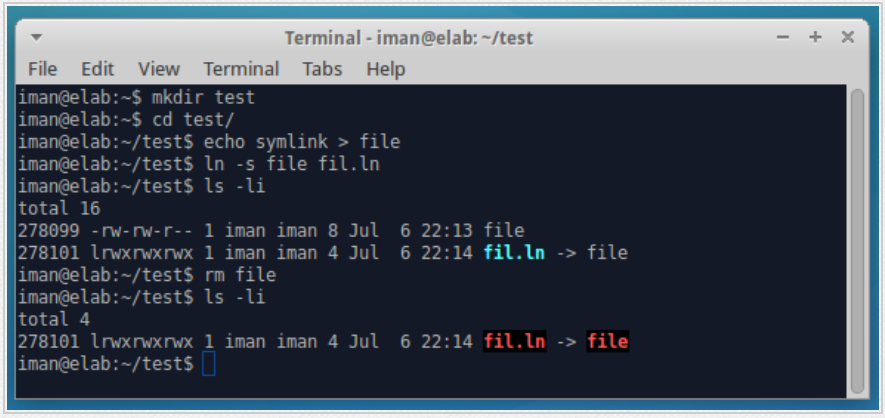
```
$ls -li file file.ln
```

نکته : سوییچ i- در دستور ls عدد Inode فایل (ها) را نیز نشان می دهد .

حال فرض کنید فایل اصلی پاک شده اما فایل Symbolic link همچنان در دایرکتوری وجود داشته باشد . به عنوان مثال :

```
$rm file
$ls file.ln
file.ln
```

نتیجه این خواهـد شد که فایل Symbolic link همچنان باقی می مانـد اما از آن جایی که آدرس درون آن به یک آدرسـی که ماهیتی نـدارد اشاره می کند , فاقد ارزش و اعتبار می باشد (اصطلاحا Broken link می گویند) و در ترمینال با یک رنگ خاص (برای مثال قرمز) نشان داده می شوند .



نکته : برای شناسایی Symbolic link ها می توان از حرف l در ابتدای پرمیشن اسـتفاده کرد . هم چنین از علامت -> می توان فهمید که فایل قبل از -> یک Symbolic لینک و فایل بعد از -> فایل مبدا است .

نکتـه : از روی رنـگ هم می تـوان پی به Symbolic لینـک هـا برد اما بایـد توجه کرد که رنگی بودن خروجی دسـتور ls به دلیل سـوییـچ color-- است که به صورت پیش فرض روی yes یا always یا force قرار گرفته است . توضیحات بیش تر در :

```
$man ls
```

تمرین :

```
$echo test symlink > file
$cat file

$ln -s file slfile1
$cat slfile1

$ln -s file slfile2
$cat slfile2

$ls -li file slfile1 slfile2
$stat slfile1
```

Creating hard link

وظیفه ی hard link ها تقریبا شبیه به symlink ها می باشـد با این تفاوت که در نحوه ی پیاده سازی با هم تفاوت هایی دارند . گفتیم که هر فایل در هارد دیسـک یک عـدد Inode منحصـر بـه فرد دارد کـه مشـخصه ی آن فایـل است و این اعـداد Inode درون جـدولی قرار دارد . پس اگر دو فایـل عـدد Inode یکسانی داشته باشند , می توان نتیجه گرفت که هر دو لینک به یک فایل در هارد دیسـک اشاره می کنند .

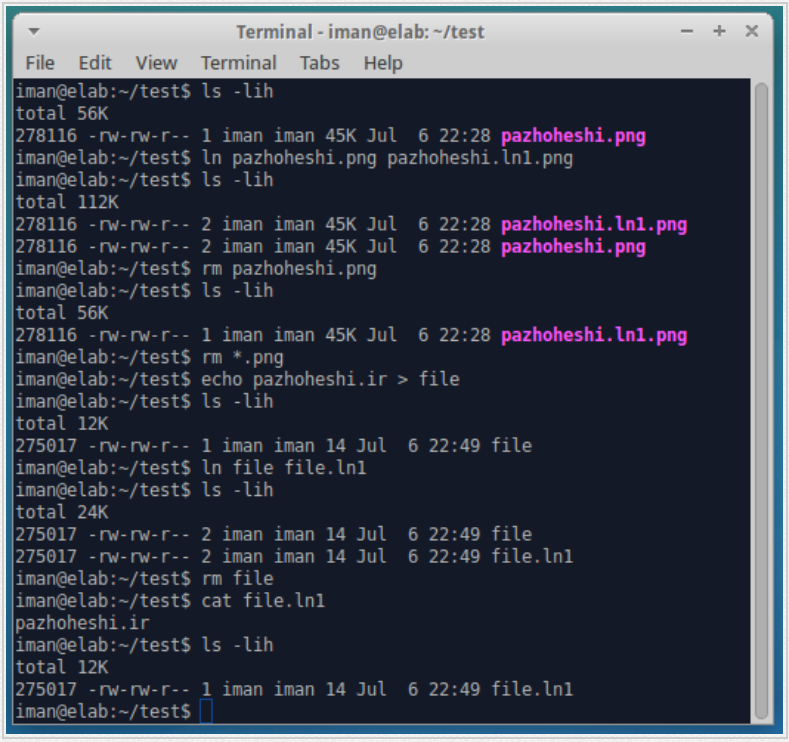
در واقع وقتی فایلی درون هارد دیسـک قرار دارد یک لینک برای ارتباط با آن توسط سیسـتم عامل ایجاد می شود (با توجه به Inode Table) یعنی زمانی که سـعی می شود به فایلی دسترسـی پیدا کرد , سیسـتم عامل ابتدا آن را به عدد Inode تبدیل می کند سپس اطلاعات را در اختیار کاربر قرار می دهد . پس منطقا تعداد زیادی از این لینک ها می توانیم داشته باشیم که همه ی آن ها به یک فایلی اشاره کنند .

به عبـارتی دیگر درون Inode tables می توانیم عدد Inode یکسانی با اسـم های مختلفی داشـته باشـیم که هر کدام از آن ها به یک فایل خاص اشاره می کننـد و لینـک می شونـد . و اگر یکی از آن هـا پاک شود به دلیل این که هنوز یک یا چنـد لینک دیگر (با همان عـدد inode و اسـم دیگر) به فایل وصـل شده است , دسترسـی به اطلاعات از بین نمی رود .

نتیجه ی یکی بودن عـدد Inode این خواهـد شـد که تظیمـاتی از قبیل پرمیشن , سازنـده , گروه سازنـده و ... در همه ی این لینک ها یکسان است چرا که همان طور که گفته شد , همه ی این لینک ها به یک فایل خاص بر روی هارد دیسک اشاره می کنند .

نکته : با توجه به مفاهیم بالا , طبیعتا اگر یکی از این لینک ها (حتی لینک اول) پاک شود , لینک های بعدی همچنان فعال و ارزشـمند هستند چرا که هنوز راهی ارتباطی برای ارتباط با فایل درون هارد وجود دارد . مثال : فرض کنید از صفحه ی نمایش عکس گرفتیم و با نام pazhoheshi.png ذخیره کردیم .

```
$ln pazhoheshi.png pazhoheshi.ln1.png
$ls -lih
```



همـان طور که در عکس بالاـ مشـخص است , دو فایل pazhoheshi.png و pazhoheshi.ln1.png هر دو به یـک فایل عکسـی درون هارد اشاره دارنـد و همـان طور که مشـخص است هر دو دارای شـماره ی Inode برابر هسـتند همراه بـا این که هر دوی این فایـل حجمی برابر بـا 45K دارنـد که باز هم نشان دهنده ی این موضوع است که عکس بر روی هارد حجمی برابر با 45K دارد .

نکته : اگر بر روی دایرکتوری جاری یعنی test سایز دایرکتوری را ببنیم خواهیم دید که حجم دایرکتوری را 45KB نشان می دهد نه 90KB . دلیل این موضوع باز یکی بودن عدد inode دو فایل است که در واقع هر دوی این فایل به اطلاعات خاصی درون هارد اشاره می کنند .

در ادامه (ادامه ی عکس) مشاهـده می کنیـد که همین روند را برای یک فایل متنی دنبال کردیم . و می بینید که با پاک کردن فایل اول , هنوز می توان به فایل اصلی دسترسی داشت .

توجه : این عمـل کپی گرفتن از فایـل نیست ! چرا که کپی از فـایلی در واقع یک فایل جدیـد با ویژگی های جدیـد درون هارد دیسک ایجاد می کنـد اما hard link فقط پلی ارتباطی با اطلاعات درون هارد است (مثلا پلی برای دسترسی به یک فایل)

نکته : همان طور که گفته شد , کپی یک فایل و فایل اصلی , اعداد Inode متفاوتی دارند .

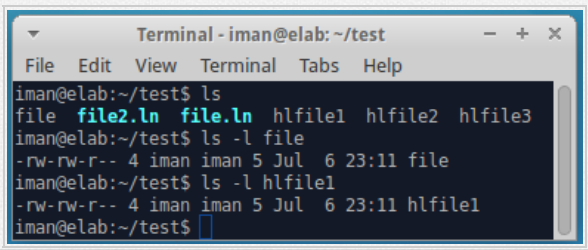
نکته : hard link های یک فایل , پرمیشن , سازنده و دیگر اطلاعات یکسانی دارند و تغییر در هر کدام برابر با تغییر در همه ی link ها می باشد .

نکته : hard link برای دایرکتوری نمی باشد و فقط بر روی فایل ها جوابگو است .

نکته : از hard link می توان به عنوان یک بک آپ استفاده کرد با این مزیت که حجمی تکراری هارد دیسک را اشغال نمی کند .

نکته : hard link فقط درون یک پارتیشن اتفاق میافتد یعنی نمی توان یک hard link از یک فایلی که درون یک پارتیشن دیگر است ایجاد کنیم .

برای تعیین تعداد hard link ها می توان از ستون دوم (از سمت چپ) دستور ls -l استفاده کرد . به عنوان مثال :



همان طور که مشاهده می کنید عدد 4 مشخصه ی تعداد hard link های آن فایل است یعنی : file / hdfile1 / hlfile2 / hlfile3

نکته : ستون دوم (از سمت چپ) دستور ls -l برای دایرکتوری (ها) , نشان دهنده ی تعداد دایرکتوری های موجود درون آن دایرکتوری است که این عدد از 2 شروع می شود چرا که درون هر دایرکتوری دو دایرکتوری زیر وجود دارد :

```
. ..
```

که . مشخصه ی همان دایرکتوری و .. مشخصه ی دایرکتوری قبلی است .

تمرین :

```
$echo test hardlink > file
$cat file

$ln file hlfile1
$cat hlfile1

$ln file hlfile2
$cat hlfile2

$ln file hlfile3
$cat hlfile3

$ls -i file hlfile1 hlfile2 hlfile3
$ls -il file hlfile1 hlfile2 hlfile3
$stat hlfile1

$ls -li file [h]lfile[1-3]
```

104.7 : Find System Files and Place Files in the Correct Location

Description : Candidates should be thoroughly familiar with the Filesystem Hierarchy Standard (FHS), including typical file locations and directory classifications.

Key Knowledge Areas:

- Understand the correct locations of files under the FHS.
- Find files and commands on a Linux system.
- Know the location and purpose of important file and directories as defined in the FHS.

The following is a partial list of the used files, terms and utilities:

- find
- locate
- updatedb
- whereis
- which
- type
- /etc/updatedb.conf

LSB یا Linux Standard Base استانداری است که در سال 1993 برای توزیع های لینوکسی ارایه شد به طور خلاصه می توان گفت برای نظم و ترتیب دادن به توزیع های مختلف مورد استفاده قرار می گیرد . یک بخشی یا زیر شاخه ای از LSB استانداردی به نام FHS یا به طور دقیق تر Filesystem Hierarchy Standard است که در آن استاندارد هایی برای ترتیب فایل ها و دایرکتوری ها گفته شد یا به طور دقیق تر استانداردی برای فایل سیستم در نظر گرفته شد . دلیل این موضوع این است که توزیع های مختلف به دلایل اسمی (اسم دایرکتوری ها) و یا ترتیب فایل ها با هم فرق نداشته باشند مثلا زمانی رو فرض کنید که پشت یک سیستم ردهتی نشسته اید , اگر ردهت از این استاندارد رعایت نمی کرد و برای هر دایرکتوری اسمی دلخواه انتخاب می کرد یا مثلا مسیر کانفیگ برنامه ها را در مسیر دلخواهی قرار می داد , شما 70٪ کارایی خود را از دست می دادید .

این موضوع رو توی دنیای واقعی هم میشه دید مثلا 90٪ وسایل برقی داخل ایران دوشاخه هایی مشابه دارند طبیعتا اگر محصولی با دوشاخه ای متفاوت تولید کنید , فروش و استفاده کننده هاتون هم کمتر می شود . در لینوکس هم همین طور , می تونید استاندارد رو رعایت نکنید اما از طرفی کاربران کمی هم از توزیع شما استفاده می کنند .

البته با وجود این استاندارد بعد از این همه سال , هنوز هم حتی توزیع های بزرگ به طور 100٪ از این استاندارد رعایت نمی کنند . ساده ترین دلیل استفاده نکردن از LSB سلیقه و دلایل شخصی می تواند باشد (یا مثلا دلایل تجاری و سیاسی و ...)

پس FHS سلسله مراتب فایل ها و دایرکتوری های فایل سیستم را مشخص می کند مثلا مسیر کانفیگ برنامه ها در کجا باشد یا Library ها در کدام مسیر باشد و ...

طبق این استاندارد گروهی از دایرکتوری ها essential یا ضروری هستند و گروهی non essential یا غیر ضروری هستند . پس طبق تعریف یک ماشین لینوکسی دایرکتوری های دسته ی اول یعنی essential ها را باید داشته باشید اما non essential ها را می تواند نداشته باشد .

نکته : طبق این استاندارد هر چه دایرکتوری ای به / نزدیک تر باشد , فایل های درون آن نیز از اهمیت بیش تری برخوردار هستند و طبیعتا هر چه از / دور شویم با فایل ها و دایرکتوری های کم اهمیت تری رو به رو خواهیم شد .

essential ها یا ضروری ها :

/bin : برنامه هایی که برای سیستم عامل ضروری بوده و تمامی کاربران می توانند از آن ها استفاده کنند , درون این دایرکتوری قرار می گیرد مثلا برنامه یا دستور ls و یا mkdir و ...

نکته : دستور و برنامه ای داریم به نام firefox . از آن جایی که این برنامه برای سیستم عامل ضروری نیست پس مسیر قرار گیری آن درون bin/ نیست .

sbin/ : گفتیم که برنامه و یا دستور هایی که تمامی کاربران می توانند استفاده کنند درون bin/ قرار می گیرد اما اگر برنامه ای مخصوص یوزر های ادمین باشد و یا برای اجرا پرمیشن root لازم داشته باشد و برای سیستم عامل ضروری باشد , درون این دایرکتوری یعنی sbin/ قرار می گیرد مثلا fdisk که برای پارتیشن بندی است

etc/ : کانفیگ تمامی برنامه ها در این دایرکتوری قرار می گیرد که در اکثر موارد فایل هایی متنی هستند به غیر از چند مورد خاص . طبیعتا اگر از این دایرکتوری یک بک آپ بگیریم می توانیم کانفیگ تمامی برنامه های سیستم را به سیستم جدید منتقل کنیم .

lib/ : تمامی Library و ماژول های کرنل در این دایرکتوری قرار دارد .

mnt/ : دایرکتوری که برای mount کردن در نظر گرفته شده است

مفاهیم Mounting :

در لینوکس یک فایل سیستم اصلی داریم که تمامی فایل های سیستمی و دایرکتوری ها درون آن قرار می گیرند که با / نمایش داده می شود در واقع / تنه ی اصلی سیستم بوده و دیگر دایرکتوری ها حکم شاخه ها را بازی می کنند . حال زمانی را فرض کنید که یک پارتیشن دیگر و یا یک هارد دیسک دیگر متصل به سیستم داریم . در این صورت برای این که سیستم عامل بتواند اطلاعات درون آن پارتیشن و یا هارد دیسک را بخواند باید به یکی از این دایرکتوری های متصل به تنه ی اصلی سیستم (/) متصل شود . به این عمل mount کردن یک پارتیشن یا دیوایس می گویند .

پس لینوکس یک تنه ی اصلی دارد و هر چند پارتیشن و یا هارد دیسک به آن اضافه شود , در حکم یکی از زیر شاخه های تنه ی اصلی به / متصل می شوند .

جایی که پارتیشن mount می شود مهم نیست اما به صورت پیش فرض دایرکتوری برای این منظور در نظر گرفته شده است مثل mnt/ اما می توان در دایرکتوری های دیگر نیز عمل mounting را انجام دهیم برای مثال یک پارتیشن و یا هارد جدا به Home Directory یکی از یوزر ها اختصاص دهیم

مثالی فنی تر : فرض کنید که یک سرور لینوکسی دارید که به صورت تخصصی روی سرویس ایمیل کار می کند و برای ذخیره کردن ایمیل ها و فایل ها تا یک سال دیگر یک هارد دیسک 2 TB ای در نظر گرفته اید . حال فرض کنید هارد دیسک شما زود تر از موید احتمالی پر شده است . در این حال کافیست یک هارد دیسک جدید خریده و آن را در مسیر قبلی mount کنید .

توجه داشته باشید توی این حالت از آپشن خاصی در برنامه ی سرویس ایمیل دهی استفاده نکردید . برای مثال برنامه در مسیری خاص عمل ذخیره سازی را انجام می دهد و تغییر مسیر آن مستلزم راه اندازی و کانفیگ مجدد برنامه می باشد . اگر قابلیت mount کردن در هر مسیر و دایرکتوری ای وجود نداشت می باست سرویس را برای مدت زمان طولانی متوقف کرد سپس تمامی مراحل را از اول تکرار کرد یعنی سرمایه ای از وقت و انرژی گرفته می شد .

این حالت بار ها در ویندوز به چشم خورده است . اما در ویندوز پارتیشن ها با کمیتی به نام Drive letter شناسایی می شود پس هر پارتیشن یک تنه ی اصلی به حساب می آید که در داخل خود زیر شاخه ها ی (دایرکتوری ها) متعددی دارد .

در مثالی که بالا گفته شد . اگر سرور , ویندوزی بود می توانستیم با تغییر Drive letter پارتیشن , آن برنامه را بایپس کنیم (با فرض نداشتن همچین آپشنی در برنامه) اما مشکلی که بار ها در ویندوز به چشم خورده است این است که در تعداد پارتیشن های زیاد , تغییر Drive letter معمولا با اشکالاتی همراه است .

برگردیم که دایرکتوری mnt , پس با توجه به مفاهیم بالا دایرکتوری پیش فرضی که برای mount کردن در نظر گرفته شده است mnt/ می باشد اما می توان جای دیگری نیز mount کرد .

نکته : فرض کنید یک دایرکتوری به نام iman/ همراه با چندین فایل و اطلاعات در داخل آن دارید , سپس یک پارتیشن دیگری را در این دایرکتوری mount می کنید . نتیجه این خواهد شد که پس از mount کردن پارتیشن در دایرکتوری iman/ دیگری به اطلاعاتی که در این دایرکتوری وجود داشت دسترسی نخواهید داشت ! البته نداشتن دسترسی تا زمانی است که پارتیشن جدید در این دایرکتوری mount شده باشد . یعنی پس از این که پارتیشن را unmount کردید باز می توانید اطلاعات قبلی که در دایرکتوری وجود داشت را مشاهده کنید .

dev/ : لینوکس تمامی دیوایس ها را همانند فایل نشان می دهد ولی صرفا به این معنا نیست که هر دیوایس یک فایل است ! مثلا dev/sda/ نمایانگر نخستین درایو ساتا در سیستم است . همچنین این فایل ها هر بار که سیستم روشن می شوند , ساخته شده برای مثال وقتی سیستم خاموش است یک هارد متصل کردید سپس دوباره سیستم رو روشن کردید ...

root/ : همان طور که مشخص است Home Directory کاربر root در این مسیر تعیین شده است .

media/ : از این دایرکتوری معمولا برای mount های موقت استفاده می کنند مثلا cdrom و یا فلش مموری را در این دایرکتوری mount می کنند . اما هارد های اینترنال که قرار هست همیشه به سیستم متصل بماند معمولا در mnt/ مونت می شوند .

non essential ها یا غیر ضروری ها :

home/ : تمامی Home Directory های کاربران در این مسیر قرار می گیرد .

tmp/ : تمامی برنامه ها فایل ها و اطلاعات موقت خود را در این دایرکتوری ذخیره می کنند معمولا با راه اندازی مجدد سیستم تمامی محتویات این دایرکتوری پاک می شود .

var/ : فایل هایی که به مرور زمان سایزشان تغییر می کند در این مسیر قرار می گیرد . به عنوان مثال فایل های Log

opt/ : برنامه هایی که توسط سوم شخص یا به صورت third party نصب شده است در این مسیر قرار می گیرد . به عنوان مثال FireFox

نکته : اگر FireFox را از سایت Mozilla گرفته و نصب شود جزء این حالت حساب می شود .
نکته : نصب فایرفاکس از روی مخازن و یا DVD نصاب جزء این حالت حساب نمی شود .

نکته : زمانی که یک برنامه به صورت third party نصب شود دیگر تمامی فایل های آن درون سیستم پخش نمی شود چرا که هر توزیع ممکن است مسیر و دایرکتوری های متفاوت داشته باشد یعنی دقیقا رعایت نکردن استاندارد FHS در توزیع های مختلف .

/boot : کلیه ی تنظیمات مربوط به boot loader سیستم و کلیه ی فایل هایی که باعث می شود یک سیستم اجرا شود در این مسیر قرار گرفته است .

/proc : فایل سیستم هایی که بر روی ram هستند , در این دایرکتوری قرار دارند . یعنی در هنگام روشن شدن سیستم ساخته می شوند و در ram قرار می گیرند . اصطلاحا گفته می شود که از این طریق می توان با کرنل در ارتباط بود . به عنوان مثال :

```
/proc/sys/net/ipv4/ip_forward
```

مقدار پیش فرض و دیفالت این فایل 0 است که اگر به 1 تغییر کند , سیستم تبدیل به یک روتر یا مسیر یاب می شود .

/usr : تقریبا مشابه / می باشد یعنی دوباره با دایرکتوری هایی مثل bin , sbin , lib و ... رو به رو خواهیم شد فقط باید به این نکته توجه کرد که هر چه از / دور تر بشویم , از اهمیت فایل ها و دایرکتوری ها کاسته می شود .

/usr/bin : برنامه هایی در این دایرکتوری قرار می گیرد که برای کاربران معمولی بوده با این تفاوت که کم اهمیت برای سیستم عامل می باشند .

/usr/sbin : برنامه هایی که دارای اهمیت کمتر (به دلیل دور بودن از /) و برای ادمین ها (s) است , در این دایرکتوری قرار می گیرد .

/usr/share : فایل هایی در این دایرکتوری قرار می گیرد که 3 ویژگی زیر را داشته باشند :

1. اسم فایل ها
2. سایز و حجم ثابتی داشته باشند یعنی به مرور زمان حجمشان تغییر نمی کند
3. فایل ها به صورت Architecture-independent باشد . یعنی مستقل از معماری (32 یا 64 بیت) باشند . مثل آیکون ها و ... که هیچ ارتباطی با نوع پردازنده ندارند .

3. The /usr/share hierarchy is for all read-only **architecture independent** data files

مثال : فونت های سیستم و یا آیکون ها و ...

/usr/include/ : معمولا هدر فایل ها در این دایرکتوری قرار می گیرد .

/usr/lib/ : در این دایرکتوری Library برنامه های کم اهمیت (برای سیستم) قرار می گیرد مثل فایرفاکس

/usr/local/ : تقریبا همان اتفاقی که / میافتید در این جا نیز رخ می دهد . باز 2 نکته را باید در نظر گرفت , اول این که چون از / دور شدیم پس اهمیت فایل های درون این دایرکتوری کم است دوم این که فایل ها و یا برنامه هایی درون این دایرکتوری قرار می گیرد که به صورت local در سیستم ساخته شده است . مثلا زمانی را فرض کنید که ادمین سرور یک اسکریپت برای دسته ای از کار ها نوشته است . در این صورت می توان این برنامه را در این مسیر قرار داد . حال اگر برنامه فقط برای یوزر ادمین باشد باید آن را به sbin و اگر برای تمامی یوزر ها باشد باید آن را به bin انتقال داد .

نکته : فقط یوزر root می تواند به این دایرکتوری دسترسی کامل داشته باشد و بقیه ی یوزر ها تنها مجوز استفاده (اجرا) دارند .

Locating Binaries' Command

دستور which :

از این دستور برای پیدا کردن فایل هایی که قابلیت execute دارند استفاده می شود . برای مثال دستور ls :

```
$which ls
/bin/ls
```

در خروجی دستور بالا , مسیر فایل اجرایی برنامه ی ls را می توان مشاهده کرد . مثالی دیگر :

```
$which passwd
$which fdisk
```

دستور whereis :

تقریبا شبیه به دستور which می ماند با این تفاوت که مسیر man page را نیز نمایش می دهد .

نکته : از سویچ b- می توان فقط به دنبال binaries فایل مورد نظر بود .

نکته : از سویچ m- می توان فقط به دنبال man page فایل مورد نظر بود .

دستور type :

تقریبا برابر با دستور which می باشد با این تفاوت که فایل های shell builtin را تشخیص می دهد . به عنوان مثال :

```
$which set
$type set

$which echo
$type echo
```

```
$type ls
$type if
$type for
```

نکته : دستور echo هم به صورت shell builti و هم در مسیر /bin/echo قرار دارد . اما در هنگام اجرای دستور echo , echo به صورت shell builti اجرا می شود . به طور دقیق تر shell builti اولویت اول دارد .

مفهوم PATH :

طبیعتا وقتی به دنبال فایل binaries می گردیم باید یک مسیر و یا یک بازه ی مشخص برای جست و جو در نظر بگیریم که به این بازه اصطلاحا PATH گفته می شود .

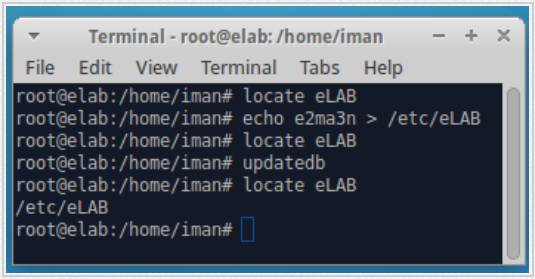
برای مشاهده ی PATH یا این بازه می توانید به صورت زیر عمل کنید :

```
$echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
```

Locating Files' Command

دستور locate :

با استفاده از این دستور نیز می توان در کل فایل سیستم به دنبال فایلی گشت . این دستور برای خود یک دیتابیس دارد که مسیر تمامی فایل ها و دایرکتوری ها را در خود دارد , طبیعتا اگر فایلی یا دایرکتوری ای به فایل سیستم اضافه شود دیگر با استفاده از این دستور پیدا نمی شود چرا که آدرس آن درون دیتابیس وجود ندارد . به عنوان مثال :



دستور find :

از این دستور می توان برای سرچی دقیق تر استفاده کرد . به این صورت که واقعا تک تک دایرکتوری ها را به صورت آنلاین جست و جو می کند . به Syntax این دستور نگاه کنید :

```
find where-to-look criteria what-to-do
```

به عنوان مثال :

```
$find ~/Desktop -iname "*pazhoheshi*"
```

توجه داشته باشید که از سویچ iname- برای مشخص کردن اسم فایل استفاده می کنیم . مثال :

```
$find ~ -inum 1723181
```

از سویچ inum- برای سرچ عدد Inode ای خاص در تمامی فایل ها و دایرکتوری ها استفاده می کنیم . مثال :

```
$find /usr -perm 4755
```

از سویچ perm- برای پیدا کردن تمامی فایل ها و دایرکتوری ها با پرمیشنی مشخص استفاده می کنیم . مثال :

```
$find /usr -perm -4000
```

توجه داشته باشید که علامت - قبل از عدد 4000 به مفهوم پیدا کردن تمامی فایل ها و دایرکتوری هایی است که پرمیشنی بیش تر از 4000 دارد .

```
$find ~/src -iname "*.c" -exec chmod g+w {} \;
```

در مثال بالا ابتدا با استفاده از سویچ -iname به دنبال تمامی فایل هایی با پسوند c می گردیم سپس به Group همه ی آن ها پرمیشن نوشتن یا همان write را اضافه می کنیم .

```
#find /etc -iname "*.conf" -exec cp {} ~/config \;
```

در مثال بالا ابتدا با استفاده از سویچ -iname تمامی فایل ها با پسوند conf در مسیر /etc را پیدا کرده سپس تمام آن ها را در مسیر ~/config کپی می کنیم .

103.2 Process text streams using filters

Description Candidates should should be able to apply filters to text streams.

Key Knowledge Areas:

Send text files and output streams through text utility filters to modify the output using standard UNIX commands found in the GNU textutils package.

The following is a partial list of the used files, terms and utilities:

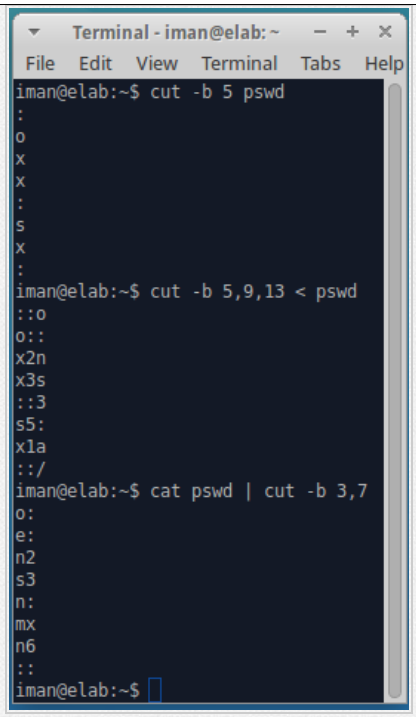
cat
cut
expand
fmt
head
od
join
nl
paste
pr
sed
sort
split
tail
tr
unexpand
uniq
wc

دستور cut :

```
$cp /etc/passwd pswd
$cut -b 5 pswd
```

از سویچ -b برای مشخص کردن بایت پنجم هر خط استفاده می کنیم .

```
$cut -b 5,9,13 < pswd
$cat pswd | cut -b 3,7
```




```
$cat pswd | cut -d: -f1 > usernames
```

از سویچ d- برای مشخص کردن ستون ها (در این جا : جدا کننده ی هر ستون است) و از سویچ f- برای مشخص کردن ستون اول استفاده کردیم و در نهایت Standard Output را در فایل usernames می ریزیم .

```
$cat pswd | cut -d: -f1,3
```

با استفاده از دستور بالا ستون 1 و 3 را که معرف نام کاربری و user ID را جدا کرده ایم .

```
$cat pswd | cut -d: -f1-4
```

با استفاده از دستور بالا ستون 1 تا 4 را جدا کرده ایم .

دستور head :

از این دستور برای خواندن head فایل استفاده می کنیم . مثال :

```
$cat pswd | cut -d: -f1 | head -n3 > 3users
```

با استفاده از سویچ n- سه خط اول را جدا کرده و در فایل 3users ریخته ایم .

نکته : به صورت پیش فرض عدد سویچ n- ده می باشد .

مثال :

```
$cat pswd | cut -d: -f1 | head -c5
```

با استفاده از سویچ c- پنج بایت اول را جدا کردیم . توجه داشته باشید که \n و ... هم یک بایت حساب می شوند .

دستور tail :

از این دستور برای خواندن آخر یک فایل استفاده می کنیم . مثال :

```
$cat pswd | cut -d: -f1 | tail -n5 > 5Lastusers
$cat pswd | cut -d: -f1 | tail -c5
```

دو سویچ n و c استفاده شده در tail دقیقا مثل heat عمل می کند . مثال :

```
$tail -f pswd
```

با استفاده از سویچ f- آخر فایل pswd خوانده می شود اما به صورتی که فایل بسته نشده و هر تغییری که در فایل به صورت همزمان اعمال شود را می خواند و نمایش می دهد . مثلا می توان برای خواندن فایل های log استفاده کرد چرا که در فایل های log مرتبا اطلاعات append می شود .

دستور expand and unexpand :

از دستور expand برای تبدیل تمامی tab های استفاده شده در هر خط به Space استفاده می شود . مثال :

```
$expand file1 > file2
$ls -l | unexpand
```

دستور sort :

برای مرتب کردن خروجی می توان از این دستور استفاده کرد . به عنوان مثال برای مرتب کردن یک لیست بر حسب حروف الفبایی و ...

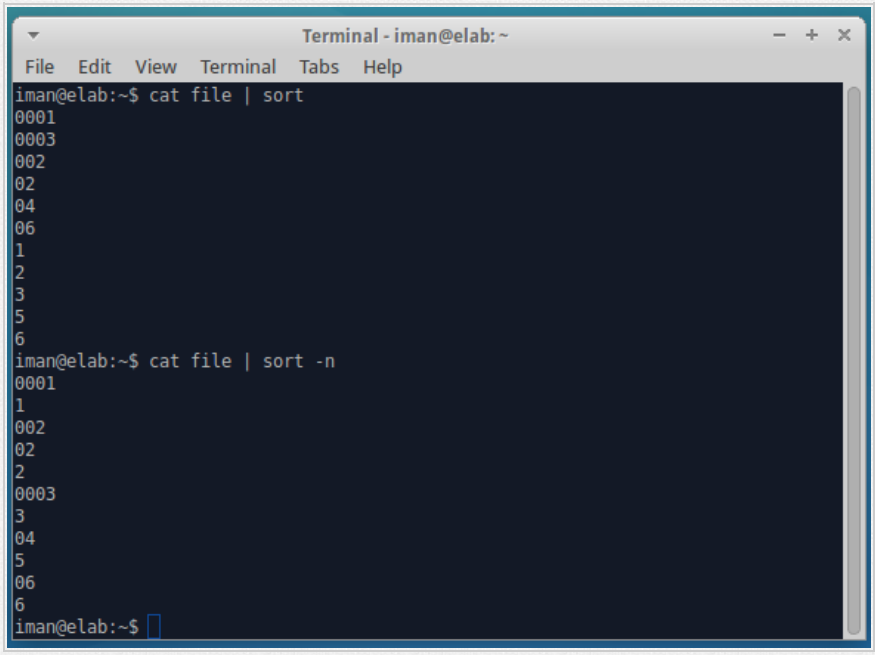
مثال :

```
$cat /etc/passwd | cut -d: -f1 | sort > sortedUsers
```

مثال :

```
$cat pswd | cut -d: -f3 | sort
$cat pswd | cut -d: -f3 | sort -n
```

از سویچ n- برای مرتب کردن اعداد از کوچیک به بزرگ استفاده می کنیم .



نکته : از سویچ r- برای برعکس مرتب کردن استفاده می کنیم . یعنی مثلا برای اعداد , از بزرگ به کوچیک مرتب می شود . مثال :

```
$cat pswd | cut -d: -f1 | sort -n -r
```

مثال :

```
$cat pswd | cut -d: -f1,3 | sort -t: -k2
```

در دستور بالا با استفاده از سویچ t- ستون دوم را مشخص کرده و با استفاده از سویچ k- شماره ی ستونی که قرار است بر حسب آن مرتب شود را مشخص کردیم . مثال :

```
$cat pswd | cut -d: -f1,3 | sort -t: -k2 -n | cut -d: -f1
```

با استفاده از دستور بالا , تمامی یوزر ها که بر حسب یوزر آی دی از کوچک به بزرگ مرتب شده است , نمایش داده می شود .

دستور nl :

برای شماره گذاری هر سطر از این دستور استفاده می کنیم . مثال :

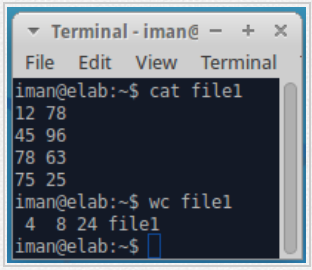
```
$cat pswd | cut -d: -f1 | nl
```

دستور wc :

اگر man page این دستور را ببینید خواهید دید که :

Print newline, word, and byte counts for each FILE

یعنی تعداد خط ها و کلمه ها و بایت ها نمایش داده می شود . به عنوان مثال :



از سمت چپ , عدد 4 یعنی این که فایل file1 دارای 4 خط است , عدد 8 یعنی فایل file1 دارای 8 کلمه است (12 و 78 و 45 و ... هر کدام یک کلمه حساب می شود) , عدد 24 یعنی فایل file1 بیست و چهار بایت دارد (هر کدام از کارکتر ها یعنی 1 و 2 و Space و 7 و \n یک کارکتر حساب می شود)

نکته : از سویچ l- می توان فقط برای نمایش تعداد خطوط استفاده کرد . مثال :

```
$cat /etc/passwd | wc -l
```

دستور tac :

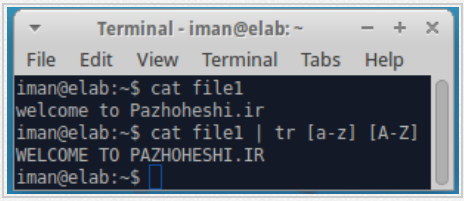
این دستور دقیقا برعکس cat عمل می کند . اگر دقت کنید می بینید که عبارت tac هم بر عکس شده ی عبارت cat است . مثال :

```
$tac pswd
```

دستور tr :

به طور خلاصه از این دستور برای ترجمه ی کارکتر ها استفاده می شود یعنی تعویض کارکتر ها با یک کارکتر دیگر . مثال :

```
$cat file1 | tr a-z A-Z
```

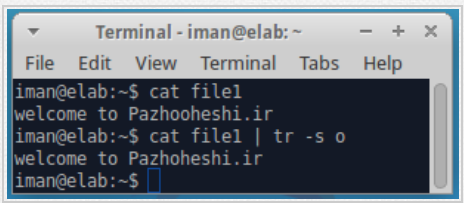


مثال :

```
$cat file1 | tr aio @!*
$ls -l | tr rwx ---
$
$cat file1 | tr -s o
$ls -l | tr rwx --- | tr -s -
ls | tr -d 'rwx'
```

نکته : سویچ s- تعداد کارکترهای تکراری ای که در کنار هم هستند را تبدیل به یک کارکتر می کند .

نکته : سویچ d- کارکتر های آرگمان را پاک می کند .



ادامه ی 103.2 در جلسه ی بعد

منبع : پژوهشی دات آی آر | Pazhoheshi.ir

نویسنده : e2ma3n

دانلود PDF این مقاله

هر گونه نظر / ایراد / اشکال و ... را از طریق ایمیل e2ma3n@Gmail.com در میون بزارید . با تشکر

E2MA3N ۹۳/۰۴/۱۵ ۰ ۰

Symbolic link | hard link | lpic1 | lpic1 101 | symlink | آموزش 104.6 | آموزش lpic1 101 104.6 | آموزش فارسی lpic

آموزش فارسی 1 lpic | آموزش فارسی لینوکس

نظرات (۱)

sc4recr0w ۱۷ تیر ۹۳ ، ۱۵:۳۸

Wow Khaste nabashi :X
alie

ارسال نظر

ارسال نظر آزاد است، اما اگر قبلا در بیان ثبت نام کرده اید می توانید ابتدا وارد شوید.

نام *

پست الکترونیک

سایت یا وبلاگ

پیام *

کد امنیتی *

نظر بصورت خصوصی ارسال شود

پست الکترونیک برای عموم قابل مشاهده باشد

ارسال