

part 1

Shell Scripting

www.esecurity.ir

collector : saeed hosseini

shell scripting

**In
Linux**

Collector

Saeed hosseyni

www.esecurity.ir

part 1

فصل یک

تاریخچه و مفاهیم ----- صفحه ۴

فصل دوم

اصول اولیه و تعریف برخی مفاهیم شل ----- صفحه ۶

فصل سوم

فایل ها ----- صفحه ۱۶

فصل چهارم

دایرکتوری ها ----- صفحه ۲۷

فصل پنجم

پروسس ها ----- صفحه ۳۳

فصل ششم

متغیرها ----- صفحه ۴۱

این راهنما به صورت آزاد و تحت مجوز GFDL منتشر می‌شود بنابراین شما با توجه به مفاد این گواهی می‌توانید آن را صورت آزاد دریافت کنید، و به طور کاملاً قانونی به دیگران بدهید و بر اساس نیازهای روز و خودتان محتویاتش را تغییر داده و حتی منتشر کنید. همانطور که در روند توسعه نرم‌افزارهای آزاد تمام کاربران یک برنامه به عنوان همکار در نظر گرفته می‌شوند در اینجا نیز این شمایید که با کمک خود می‌توانید این راهنما را به‌روز و زنده نگه دارید؛



فصل اول - تاریخچه و مفاهیم

تاریخچه

گنو

در دهه ۱۹۷۰ و اوایل دهه ۱۹۸۰، برنامه‌نویسان کامپیوتر تمامی آنچه را می‌نوشتند با دیگران به اشتراک می‌گذاشتند. هر شخصی متن برنامه خود را در اختیار دیگر برنامه‌نویسان می‌گذاشت. اشتراک گذاری نرم‌افزار امری عادی بود.

در سال ۱۹۷۱ ریچارد استالمن کار خود را در دانشگاه MIT آغاز کرد، و در گروهی که منحصر از نرم‌افزار آزاد استفاده می‌کردند به کار پرداخت. شرکت‌های کامپیوتری نیز اغلب نرم‌افزار آزاد توزیع می‌کردند. برنامه‌نویسان در همکاری با یکدیگر آزاد بودند و اغلب نیز همین کار را انجام می‌دادند.

در اوایل دهه ۱۹۸۰ همه چیز به آرامی دگرگون شد. شرکت‌های نرم‌افزاری دیگر متن برنامه‌های خود را در اختیار دیگران قرار نمی‌دادند. برنامه‌نویسان نمی‌توانستند برنامه‌های دیگران را تغییر داده و آن را بهبود بخشند. از این پس به اشتراک گذاری نرم‌افزار جرم محسوب می‌شد.

اولین مرحله برای داشتن یک کامپیوتر آزاد، وجود یک سیستم عامل آزاد بود. بنابراین ریچارد استالمن زمانی که دانشجوی دکتری بود از شغل و تحصیل خود در MIT استعفا داد و در اوایل سال ۱۹۸۳ پروژه گنو را آغاز و اطلاعیه اولیه آن را در سپتامبر ۱۹۸۳ منتشر کرد. در طی سال‌های ۱۹۸۴ تا ۱۹۸۵ او کامپایلر جی‌سی‌سی و ویرایشگر ایمکس را نوشته و به صورت آزاد منتشر کرد. به این ترتیب ابزارهای اولیه مورد نیاز برای طراحی و ساخت یک سیستم عامل فراهم شد. ابزاری مبهوت کننده برای برنامه نویسان مستقل. وی با جادوگری افسانه‌ای خود به تنهایی ابزاری را ایجاد نمود که برتر از تمام ابزارهایی که تمام گروه‌های برنامه نویسان تجاری ایجاد کرده بودند قرار گرفت GCC. یکی از کارآمدترین و قویترین کامپایلرهای است که تا کنون ایجاد شده اند.

استالمن در سال ۱۹۸۵ بنیاد نرم‌افزارهای آزاد (free software foundation) را پایه گذاری کرد، مفهوم کپی‌لفت را تعریف و اجازه نامه جامع و عمومی گنو (gnu public license) برای حمایت نرم‌افزارهای آزاد و تضمین آزادی کاربران نوشت و در اواخر همان سال نسخه کامل‌تر آن به نام اعلامیه گنو را منتشر کرد که در حال حاضر به ۲۲ زبان ترجمه شده است.

نام «گنو» به این علت انتخاب شده است که تعدادی از نیازها را برطرف می‌کند؛ نخست، یک مخفف بازگشتی برای "GNU's Not Unix" است، دوم، یک کلمه واقعی است، سوم، آهنگ گفتن (یا خواندن) آن جالب است. کلمه «آزاد» در «نرم‌افزار آزاد» به آزادی اشاره می‌کند، نه قیمت. شما برای به دست آوردن نرم‌افزار آزاد ممکن است مبلغی بپردازید یا نپردازید. در هر صورت، وقتی نرم‌افزار را در اختیار داشته باشید، سه آزادی ویژه برای استفاده از آن خواهید داشت. یک سیستم عامل شبه یونیکس خیلی بیشتر از یک هسته است؛ و شامل کامپایلرها، ویرایشگرها، برنامه‌های قالب‌بندی متن، نرم‌افزارهای پستی و خیلی چیزهای دیگر می‌باشد. بنابراین نوشتن یک سیستم عامل کامل کار بسیار بزرگی است. ما در ژانویه ۱۹۸۴ شروع به کار کردیم. سالها به طول انجامید. بنیاد نرم‌افزار آزاد در اکتبر ۱۹۸۵ بیشتر برای جذب سرمایه جهت کمک به توسعه گنو تاسیس شد.

آغاز داستان در سال ۱۹۹۱ است. زمانی که با وجود قدرت سخت افزارهای جدید، محدودیت های کامپیوترها رو به پایان می رفت. ولی هنوز چیزی کم بود... و این چیزی نبود جز فقدان عمیق در حیطه سیستم های عامل. داس، امپراطوری کامپیوترهای شخصی را در دست داشت. سیستم عامل بی استخوانی که با قیمت ۵۵۰۰۰ دلار از یک هکر سیاتلی توسط بیل گیتز (Bill Gates) خریداری شده بود و با یک استراتژی تجاری هوشمند، به تمام گوشه های جهان رخنه کرده بود. کاربران PC انتخاب دیگری نداشتند. کامپیوترهای اپل مکینتاش بهتر بودند. ولی قیمت های نجومی، آنها را از دسترس کثر افراد خارج می ساخت. خیمه گاه دیگر دنیای کامپیوترها، دنیای یونیکس بود. یونیکس به خودی خود بسیار گرانقیمت بود. آنقدر گرانقیمت که کاربران کامپیوترهای شخصی جرات نزدیک شدن به آنرا نداشتند. کد منبع یونیکس که توسط آزمایشگاه های بل بین دانشگاهها توزیع شده بود، محتاطانه محافظت میشد تا برای عموم فاش نشود. برای حل شدن این مسأله، هیچیک از تولید کنندگان نرم افزار راه حلی ارایه ندادند. بنظر میرسید این راه حل به صورت سیستم عامل MINIX ارائه شد.

این سیستم عامل، که از ابتدا توسط اندرو اس. تاننباوم (Andrew S. Tanenbaum) پروفیسور هلندی، نوشته شده بود به منظور تدریس عملیات داخلی یک سیستم عامل واقعی بود. این سیستم عامل برای اجرا روی پردازنده های ۸۰۸۶ اینتل طراحی شده بود و بزودی بازار را اشباع کرد. بعنوان یک سیستم عامل، MINIX خیلی خوب نبود. ولی مزیت اصلی آن، در دسترس بودن کد منبع آن بود. هرکس که کتاب سیستم عامل تاننباوم را تهیه می کرد، به ۱۲۰۰۰ خط کد نوشته شده به زبان C و اسمبلی نیز دسترسی پیدا می کرد. برای نخستین بار، یک برنامه نویس یا هکر مشتاق می توانست کد منبع سیستم عامل را مطالعه کند. چیزی که سازندگان نرم افزارها آنرا محدود کرده بودند. دانشجویان کامپیوتر در سرتاسر دنیا با خواندن کتاب و کدهای منبع، سیستمی را که در کامپیوترشان در حال اجرا بود، درک کردند. و یکی از آنها لینوس توروالدز (Linus Torvalds) نام داشت.

لینوس بندیکت توروالدز (Linus Benedict Torvalds) دانشجوی سال دوم علوم کامپیوتر دانشگاه هلسینکی فنلاند و یک هکر خود آموخته بود. این فنلاندی ۲۱ ساله، عاشق وصله پینه کردن محدودیت هایی بود که سیستم را تحت فشار قرار می دادند. ولی مهمترین چیزی که وجود نداشت یک سیستم عامل بود که بتواند نیازهای حرفه ای ها را برآورده نماید. MINIX خوب بود ولی فقط یک سیستم عامل مخصوص دانش آموزان بود و بیشتر به عنوان یک ابزار آموزشی بود تا ابزاری قدرتمند برای بکارگیری در امور جدی. در این زمان برنامه نویسان سرتاسر دنیا توسط پروژه گنو (GNU) که توسط ریچارد استالمن (Richard Stallman) آغاز شده بود، تحریک شده بودند.

تا سال ۱۹۹۱ پروژه GNU تعداد زیادی ابزار ایجاد کرده بود ولی هنوز سیستم عامل رایگانی وجود نداشت. کار بر روی هسته سیستم عامل گنو موسوم به HURD ادامه داشت ولی به نظر نمی رسید که تا چند سال آینده قابل استفاده باشد. این زمان برای توروالدز بیش از حد طولانی بود... در ۲۵ آگوست ۱۹۹۱، این نامه تاریخی به گروه خبری MINIX از طرف توروالدز ارسال شد:

- از: لینوس بندیکت توروالدز
- به: گروه خبری MINIX
- موضوع: بیشتر چه چیزی را می خواهید در MINIX ببینید؟

سلام به تمام استفاده کنندگان از MINIX من در حال تهیه یک سیستم عامل رایگان فقط به عنوان سرگرمی و نه به بزرگی و حرفه ای GNU برای دستگاههای ۳۸۶ و ۴۸۶ هستم. از ماه آوریل کار را آغاز کرده ام و هم اکنون این سیستم عامل آماده است و کار می کند. دوست دارم از دیدگاه دیگران در مورد سیستم عاملم با خبر شوم. چه آنان که مینیکس را دوست دارند و چه آنان که دوست ندارند. چرا که سیستم عامل من تا حدی شبیه به مینیکس است. در حال حاضر ۱,۰۰۸ bash و ۱,۴۰ gcc را بر روی آن دارم و چیزهای دیگری که به نظر می رسد همه درست کار می کنند. این بدان معناست که طی چند ماه آینده چیز بهر دبخوری فراهم خواهد کرد و دوست دارم بدانم مردم بیشتر چه امکاناتی لازم دارند. به هر پیشنهاد و نظری خوش آمد می گویم اما قول نمی دهم که آن را انجام دهم! پ.ن: بله این نرم افزار آزاد است. البته قابل انتقال بر روی انواع دیگر رایانه نیست) چرا که دستورات AT ۳۸۶ را به کار می برد (و ممکن است غیر از دیسک سخت AT چیز دیگری را پشتیبانی نکنند. این همه چیزی است که من دارم!

همانطور که در این نامه پیداست، خود توروالدز هم باور نمی کرد که مخلوقش آنقدر بزرگ شود که چنین تحولی در دنیا ایجاد کند. لینوکس نسخه 0.01 در اواسط سپتامبر ۱۹۹۱ منتشر شد و روی اینترنت قرار گرفت. شور و اشتیاقی فراوان حول مخلوق توروالدز شکل گرفت. کدها داندل شده، آزمایش شدند و پس از بهینه سازی به توروالدز بازگردانده شدند.

لینوکس نسخه 0.02 در پنجم کتبر به همراه اعلامیه معروف توروالدز آماده شد:

آیا شما از روزهای زیبای MINIX ۱/۱ محروم شده اید؟ هنگامی که مردها مرد بودند و راه اندازهای دستگاه خود را خودشان می نوشتند؟ آیا شما فاقد یک پروژه زیبا هستید و می میرید تا سیستم عاملی داشته باشید تا بتوانید آنرا مطابق با نیازهای خود در آورید؟ اگر اینگونه است، این نامه برای شما نوشته شده است....

لینوکس نسخه ۰/۰۳ پس از چند هفته آماده شد و تا دسامبر، لینوکس به نسخه ۰/۱۰ رسید. هنوز لینوکس فقط چیزی کمی بیشتر از یک فرم اسکلت بود. این سیستم عامل فقط دیسکهای سخت AT را پشتیبانی می کرد و ورود به سیستم نداشت و مستقیماً به خط فرمان بوت می شد. نسخه ۰/۱۱ خیلی بهتر شد. این نسخه از صفحه کلیدهای چند زبانه پشتیبانی می کرد، دیسکهای فلاپی و کارتهای گرافیکی VGA، EGA، هرکولس و... نیز پشتیبانی می شدند. شماره نسخه ها از ۰/۱۲ به ۰/۹۵ و ۰/۹۶ افزایش پیدا کرد و ادامه یافت. به زودی کد آن بوسیله سرویس دهنده های FTP در فنلاند و مناطق دیگر، در سرتاسر جهان منتشر شد. به زودی صدها نفر به اردوگاه لینوکس پیوستند. سپس هزاران نفر و سپس صدها هزار نفر.

لینوکس دیگر اسباب بازی هکرها نبود. با پشتیبانی نرم افزارهای پروژه GNU، لینوکس آماده یک نمایش واقعی بود. لینوکس تحت مجوز GPL قرار داده شد. با این مجوز همه می توانستند کدهای منبع لینوکس را به رایگان داشته باشند، بر روی آنها مطالعه کرده و آنها را تغییر دهند. دانشجویان و برنامه نویسان آنرا قاپیدند. و خیلی زود تولید کنندگان تجاری وارد شدند. لینوکس به خودی خود رایگان بود و هست. کاری که این تولیدکنندگان انجام دادند، کامپایل کردن بخش ها و نرم افزارهای مختلف و ارایه آن بصورت یک فرمت قابل توزیع همانند سایر سیستم عامل ها بود، تا مردم عادی نیز بتوانند از آن استفاده کنند. همچنین اتفاقات جالبی با لینوکس رخ می دهد. در کنار PC، لینوکس به روی اکثر پلتفرمها منتقل شده است. لینوکس تغییر داده شد تا کامپیوتر دستی شرکت ۳Com یعنی PalmPilot را اجرا نماید. تکنولوژی کلاستر کردن این امکان را بوجود آورد تا بتوان تعداد زیادی از ماشینهای لینوکس را به یک مجموعه واحد پردازشی تبدیل نمود. یک کامپیوتر موازی. در آوریل ۱۹۹۶ محققین آزمایشگاه های ملی لوس آلاموس از ۶۸ کامپیوتر مبتنی بر لینوکس برای پردازش موازی و شبیه سازی موج انفجار اتمی استفاده کردند. ولی بر خلاف ابر کامپیوترهای دیگر، هزینه آنها بسیار ارزان تمام شد. ابرکامپیوتر خود ساخته آنها با تمام تجهیزات و سخت افزارها ۱۵۲۰۰۰ دلار هزینه در بر داشت و این یک دهم هزینه یک ابرکامپیوتر تجاری است. این ابرکامپیوتر به سرعت ۱۶ بیلیون محاسبه در ثانیه دست یافت و به رتبه ۳۱۵ این ابرکامپیوتر جهان دست پیدا کرد و صد البته یکی از پایدارترین آنها بود. پس از سه ماه از آغاز فعالیت، هنوز بوت نشده بود. بهترین موردی که امروزه برای لینوکس وجود دارد، طرفداران متعصب آن هستند. هنگامی که یک قطعه سخت افزاری جدید ارائه می شود، هسته لینوکس برای استفاده از آن تغییر داده می شود. برای مثال هنگام ارائه پردازنده ۶۴ بیتی شرکت توروالدز، هنوز یک انسان ساده است. بر خلاف بیل گیتس او یک میلیارد نیست.



فصل دوم - اصول اولیه و تعریف برخی مفاهیم شل

آشنایی با اصطلاحات

!echo "Hello World" - نوشتن یک برنامه ساده بش اسکریپت لینوکس و اجرای آن در شل یا ترمینال

Putting it in your path - چگونه مسیر برنامه ها را در لینوکس ذخیره کنیم تا در موقع نیاز فقط لازم باشد نام آن ها در ترمینال بنویسیم

commands - آموزش دستورات و برنامه های پیش فرض بش در لینوکس

Aliases - چگونه دستورات خواص و پیش فرض بش را کوتاهتر کنیم

shell functions - فانکشن های شل

Here Scripts - مدیریت رشته های بلند در بش اسکریپت

Variables - متغیر ها در بش اسکریپت

How to create a variable - چگونه در بش اسکریپت متغیر تعریف کنیم

Environment Variable - متغیر های محیطی در بش اسکریپت

Substitutions - جایگذاری ها در بش اسکریپت

Quoting مدیریت رشته توسط ' و " در بش اسکریپت

Shell Functions - فانکشن های شل

Constants - ثوابت در شل اسکریپت

some real works کمی کار عملی در بش اسکریپت

Flow Control - کنترل جریان در بش اسکریپت

Stay Out Of Trouble - عادت ها خوب برنامه نویسی در بش اسکریپت

Keyboard Input - مدیریت ورودی صفحه کلید در بش اسکریپت

Arithmetic - عملیات ریاضی در بش اسکریپت

Positional Parameters - پارمترها ترتیبی در بش اسکریپت

Errors and Signals and Traps - مدیریت خطا ها و علائم در بش اسکریپت

دستور

در یونیکس یک دستور برنامه‌ای است که شما قادر به اجرای آن هستید. برای اجرای یک دستور باید نام آن را در محیط شل تایپ کنید

شل یک محیط واسطه رو برای ارتباط شما با سیستم فراهم می‌کند. شل از شما مقداری رو می‌گیرد و محاسبات رو بر اساس اون مقدار برای شما انجام می‌دهد. وقتی اجرای برنامه تمام میشه خروجی برای شما نمایش داده میشه.

شل محیطی ست که میتونه دستورات برنامه‌ها و اسکریپت‌های شما رو اجرا کنه. همونطور که سیستم عامل‌های مختلفی هست شل‌های مختلفی هم وجود داره.

```

saeed@localhost:/home/saeed/Desktop
File Edit View Search Terminal Help
[root@localhost Desktop]# date
Sun Apr  5 00:37:19 PDT 2015
[root@localhost Desktop]# who
saeed    tty1          2015-03-31 04:29 (:0)
saeed    pts/0          2015-03-31 05:01 (:0.0)
[root@localhost Desktop]#

```

در اینجا دستور date اجرا شده‌است. این دستور تاریخ و زمان جاری را نمایش می‌دهد. بعد از اینکه اجرای دستور پایان یافت و خروجی آن نمایش داده شد

ترکیب چند دستور

در شل امکان ترکیب چندین دستور در یک خط وجود دارد. برای این منظور میتوان دستورات را با استفاده از ; از هم جدا کرد. در این حالت خروجی هر دستور به ترتیب بر روی صفحه‌ی نمایش نشان داده میشود. در حالت کلی به صورت زیر میتوان چندین دستور را پشت‌سر تایپ و اجرا کرد:

```

saeed@localhost:/home/saeed/Desktop
File Edit View Search Terminal Help
[root@localhost Desktop]# pwd ; ls ; date
/home/saeed/Desktop
firestarter-1.0.3-1.i386.rpm  Firestarter v.1
Sun Apr  5 00:45:40 PDT 2015
[root@localhost Desktop]#

```

در این مثال ابتدا دستور `pwd` اجرا و خروجی آن نمایش داده می شود و سپس دستورات `ls` , `date` اجرا می شود

شل چیست

شل محیطی از لینوکس می باشد که می توان بوسیله آن دستورات را به سیستم فهماند.

توجه کنید که شل قسمتی از کرنل لینوکس نمی باشد و تنها راهی برای اجرای دستورات و ایجاد فایلها می باشد. شل برنامه است که دستورات کاربر را دریافت و آن را اجرا می کند. برنامه های مختلفی وجود دارند که به عنوان شل استفاده می شوند.

برای فهمیدن مفهوم شل می توان به شکل زیر توجه کرد

انواع شل در لینوکس در زیر آمده است :

Remark	Where	Developed by	Shell Name
Most common shell in Linux. It's Freeware shell.	Free Software Foundation	Brian Fox and Chet Ramey	BASH (Bourne-Again SHell)
The C shell's syntax and usage are very similar to the C programming language.	University of California (For BSD)	Bill Joy	CSH (C SHell)
--	AT & T Bell Labs	David Korn	KSH (Korn SHell)
TCSH is an enhanced but completely compatible version of the Berkeley UNIX C shell (CSH).	--	See the man page. Type \$ man tcsh	TCSH

برای آگاهی از شلهای موجود بر لینوکس می توان دستور زیر را تایپ کرد:

\$ cat /etc/shells

در صورتی که شما از bourne-shell استفاده کنید شکل prompt به صورت \$ و در صورتی که از C-shell استفاده کنید به صورت % خواهد بود. در این ویکی، bash از خانواده ی bourne بررسی شده است.

انواع مختلف دستورات

اسامی مستعار (alias)

نام های مستعار یک دستور بلند را که مکرر اجرا می کنید را در قالب یک دستور کوچک خلاصه می کند. فرض کنید شما باید هر چند لحظه یکبار کارایی فرایند های سیستم را با دستور `ps -aux` چک کنید اما شاید نخواهد این دستور را تکرار کنید پس بهتر است از نام های مستعار یا Alias ها استفاده کنید.

از دستور alias برای تعریف Alias ها استفاده می شود. فرمت کلی Alias ها بصورت زیر هستند.

```
alias alias_name=cmd
```

بجای alias_name نام دلخواه معادل دستور را قرار دهید و بجای cmd دستور دلخواه را قرار دهید. خاصیت alias ها ماندگار نیستند و با بستن پنجره خط فرمان یا حتی با گشودن یک برگه یا Tab جدید در همان پنجره خط فرمان از بین می روند برای آنکه این تاثیر ماندگار باشد بهتر است آنها را در فایل bashrc ذخیره کنیم. فایل bashrc با هر بار ورود به سیستم خوانده می شود پس بهترین مکان برای ذخیره سازی نام های مستعار همین فایل است. این فایل در دو جا ذخیره می شود: یکی در دایرکتوری خانگی همه کاربران که اثر آن فقط بر همان کاربر است و یکی هم در دایرکتوری /etc/ که اثرش بر تمام کاربران و کل سیستم است. پس اگر می خواهید alias تعریف شده فقط برای خودتان اثر داشته باشد آنرا در فایل bashrc دایرکتوری خانگی خودتان ذخیره کنید. (این فایل پنهان است و اول آن یک علامت نقطه وجود دارد. خط اول از دو خط زیر) ولی اگر می خواهید بر کل کاربران اثر بگذارد در فایل /etc/ تعریف کنید (خط دوم از خطوط زیر). با هر ویرایشگر دلخواهی می توانید این فایل را ویرایش کنید.

```
vi /etc/bashrc
```

با هر یک از ویرایشگر ها دستورات زیر را در فایل bashrc وارد کنید:

```

saeed@localhost:/home/saeed
File Edit View Search Terminal Help
GNU nano 2.0.9 File: /etc/bashrc Modified
fi
# vim:ts=4:sw=4
alias wtf='watch -n 1 w -hs'
alias wth='ps -uxa | more'
alias attrib=chmod
alias chdir=cd
alias copy=cp
alias cp=cp -i
alias d=dir
alias del=rm
alias deltree=rm -r
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
saeed@localhost:/hom...

```

مانند دو دستور Bold اول هر وقت از کارکتر های خاص مانند dash و فضای خالی بعد از علامت = استفاده می کنیم باید آنها را سمت راست را بین ' ' و یا بین " " قرار دهیم.

نکات:

بین علامت مساوی و عبارت های سمت چپ و راست هیچ فاصله ای وجود ندارد.

نام های مستعار مانند دستور ها به بزرگی و کوچکی حساس هستند پس بین move و Move و ... mOvE در تفاوت است.

برای لیست کردن نام های مستعار درون سیستم کافیت دستور alias را به تنهایی در خط فرمان اجرا کنید و یا اینکه از روش گفته شده در [این پست](#) استفاده کنید. (دستور copmgen)

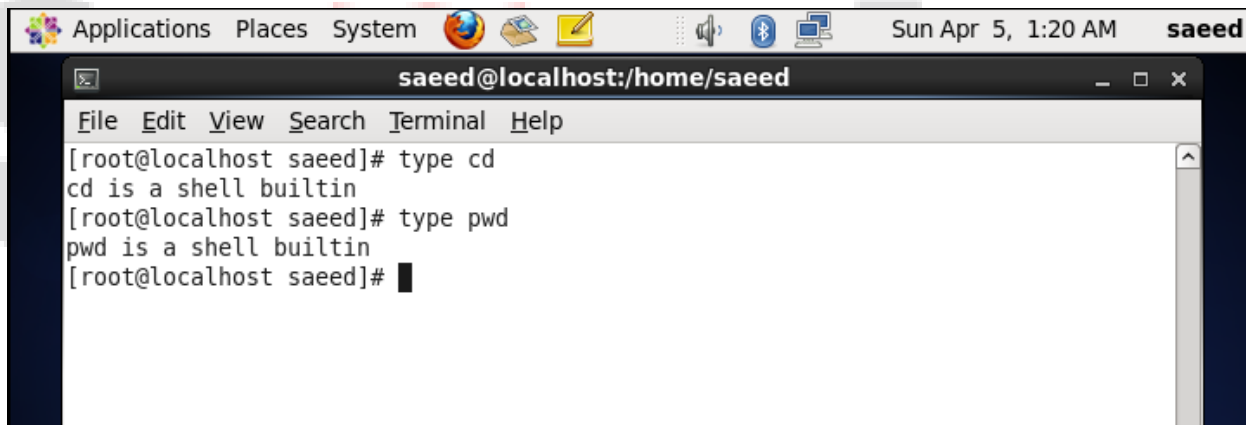
از بین بردن نام های مستعار:

با دستور unalias به شکل زیر می توانید یک alias را از سیستم حذف کنید بدون اینکه نیاز به ویرایش فایل bashrc را داشته باشید.

```
“ unalias alias_name
```

دستورات درونی (builtin)

دستوراتی که توسط خود شل اجرا می شوند و به برنامه دیگری وابسته نیست



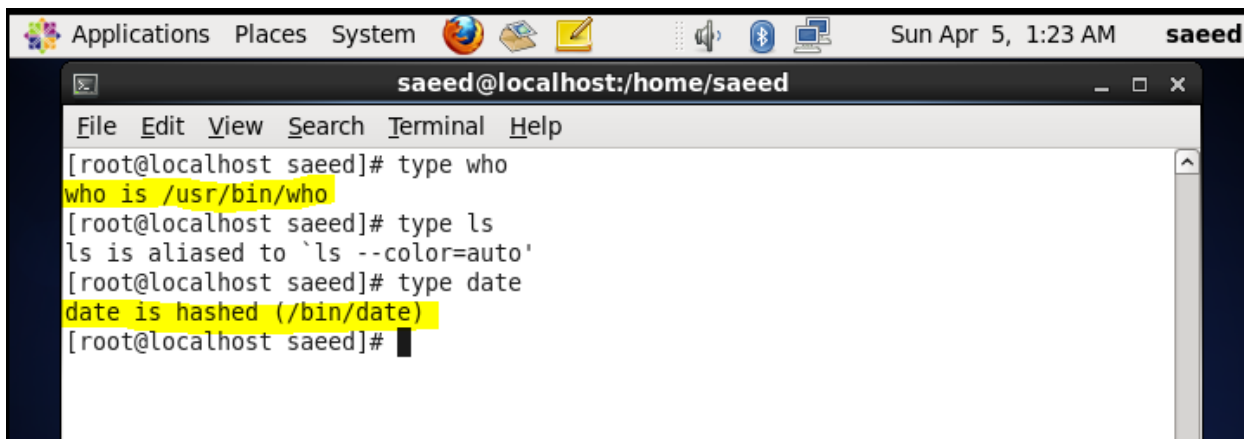
```

saeed@localhost: /home/saeed
File Edit View Search Terminal Help
[root@localhost saeed]# type cd
cd is a shell builtin
[root@localhost saeed]# type pwd
pwd is a shell builtin
[root@localhost saeed]#

```

دستورات خارجی

که اغلب دستورات را تشکیل می دهند، متناظر با یک فایل اجرایی برای شل (باینری یا اسکریپت) هستند که در پوشه های خاصی قرار دارند که بعداً معرفی خواهد شد.



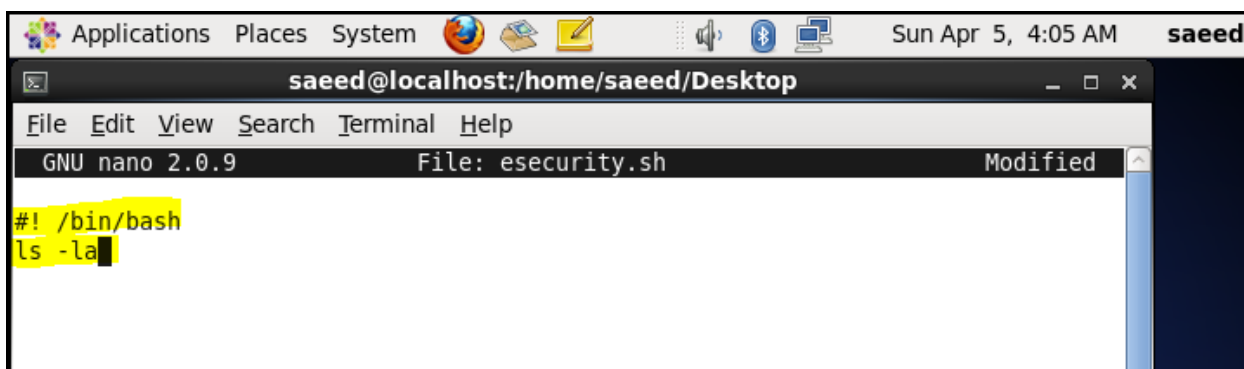
A terminal window titled 'saeed@localhost:/home/saeed' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following commands and outputs:

```
[root@localhost saeed]# type who
who is /usr/bin/who
[root@localhost saeed]# type ls
ls is aliased to `ls --color=auto'
[root@localhost saeed]# type date
date is hashed (/bin/date)
[root@localhost saeed]#
```



نوشتن و اجرای یک اسکریپت

یکی از مهم‌ترین عملیاتی پس از نوشتن یک اسکریپت باید انجام شود قابل اجرا کردن آن است. به صورتی که اسکریپت مجوز اجرا داشته باشد. برای نوشتن و اجرای یک اسکریپت ابتدا یک فایل متنی با نام first را باز کرده و سپس دو خط زیر را درون آن بنویسید و سپس آن را با فرمت sh ذخیره کنید

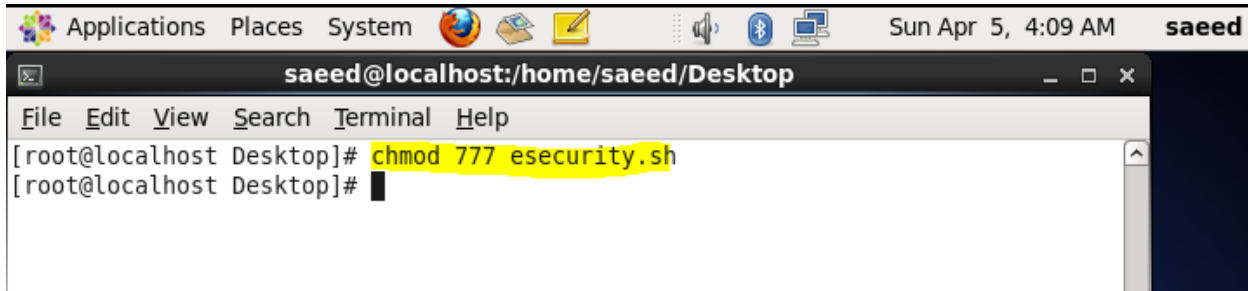


A terminal window titled 'saeed@localhost:/home/saeed/Desktop' with a menu bar (File, Edit, View, Search, Terminal, Help). The window shows the GNU nano 2.0.9 editor editing a file named 'esecurity.sh'. The content of the file is as follows:

```
#!/bin/bash
ls -la
```

خط اول این اسکریپت که معروف به خط جادویی نیز است موجب ایجاد یک نمونه‌ی جدید از شل برای اجرای دستورات می‌شود. در صورتی که خط اول در ابتدای اسکریپت نوشته نشود، از شل جاری برای اجرای اسکریپت استفاده می‌شود. به عنوان مثال در صورتی که این خط نوشته نشود اسکریپت‌هایی که با استفاده از tcsh یا csh نوشته شده‌اند توسط bourne قابل اجرا نیست.

برای اجرایی کردن فایل security.sh دستور زیر را اجرا کنید

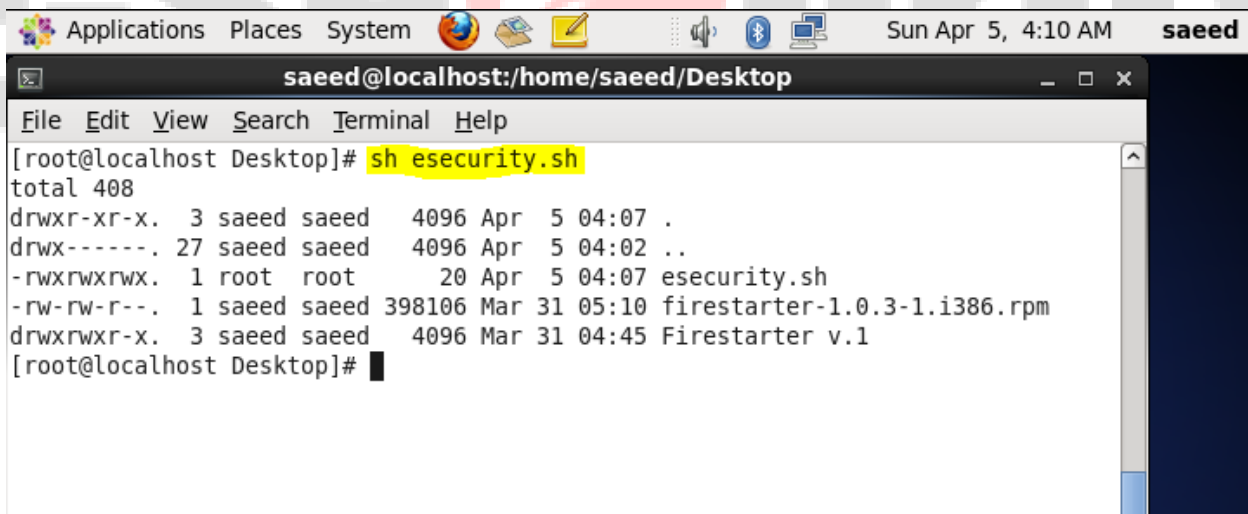


```

saeed@localhost:/home/saeed/Desktop
File Edit View Search Terminal Help
[root@localhost Desktop]# chmod 777 esecurity.sh
[root@localhost Desktop]#

```

حال برای اجرای این اسکریپت میتوان از نام آن استفاده کرد:



```

saeed@localhost:/home/saeed/Desktop
File Edit View Search Terminal Help
[root@localhost Desktop]# sh esecurity.sh
total 408
drwxr-xr-x. 3 saeed saeed 4096 Apr 5 04:07 .
drwx----- 27 saeed saeed 4096 Apr 5 04:02 ..
-rwxrwxrwx. 1 root root 20 Apr 5 04:07 esecurity.sh
-rw-rw-r-- 1 saeed saeed 398106 Mar 31 05:10 firestarter-1.0.3-1.i386.rpm
drwxrwxr-x. 3 saeed saeed 4096 Mar 31 04:45 Firestarter v.1
[root@localhost Desktop]#

```

نکته: دستور `/bin/bash #!` حتماً باید در سطر اول اسکریپت قرار داشته باشد. در غیر اینصورت شل آن را نادیده می‌گیرد.

نکته: پسوند `sh` برای فایل اسکریپت لازم نیست، و خط اول فایل برای شل کافی است تا بفهمد باید آن را با `bash` اجرا کند.

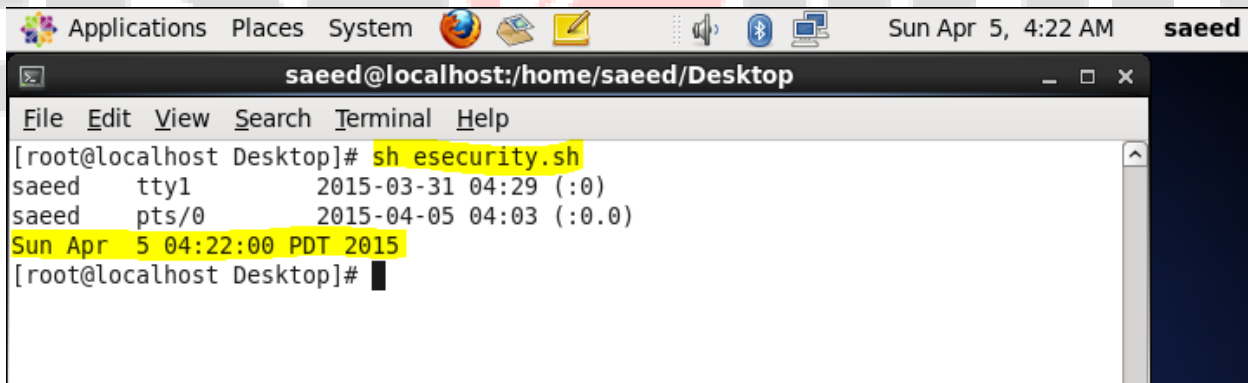
توضیحات

توضیحات خط‌هایی در اسکریپت هستند که اجرا نمیشود و فقط برای مستند سازی دستورات به کار می‌روند. در شل توضیحات با کاراکتر # آغاز می‌شوند و هر چیزی که بعد از آن نوشته شود برای اجرا نادیده گرفته می‌شود. همچنین اکثر برنامه نویسان اضافه کردن توضیحات به برنامه‌ها را جهت خوانایی آن ضروری میدانند.

مثال: اسکریپت بخش قبل را در نظر بگیرید. میتوان توضیحاتی به صورت زیر به آن اضافه نمود :

```
#!/bin/bash
# print out the date and who's logged on
date ; who ;
```

خروجی این اسکریپت با خروجی اسکریپت بخش قبل تفاوتی ندارد :



```
saeed@localhost:/home/saeed/Desktop
File Edit View Search Terminal Help
[root@localhost Desktop]# sh esecurity.sh
saeed    tty1      2015-03-31 04:29 (:0)
saeed    pts/0     2015-04-05 04:03 (:0.0)
Sun Apr 5 04:22:00 PDT 2015
[root@localhost Desktop]#
```

همچنین میتوان توضیحات را به صورت زیر نیز اضافه نمود :

```
#!/bin/bash
# print out the date and who's logged on
date ; who ; # execute the date and who commands
```

که همچنان تفاوتی در خروجی ایجاد نمی‌شود .

E-Security

فصل سوم - فایل‌ها

در سیستم‌های یونیکسی همه چیز به صورت فایل است. در این فصل و فصل بعد به برخی از ویژگی‌های انواع فایل در سیستم‌های یونیکسی میپردازیم

مشاهده لیست و مشخصات فایل‌ها

برای مشاهده لیست فایل‌های یک دایرکتوری از دستور ls استفاده می‌شود. به طور خلاصه شکل استفاده‌ی عمومی تر از این دستور به شکل زیر است:

مشاهده لیست و مشخصات فایل‌ها

برای مشاهده لیست فایل‌های یک دایرکتوری از دستور ls استفاده می‌شود. به طور خلاصه شکل استفاده‌ی عمومی تر از این دستور به شکل زیر است :


```

saeed@localhost:/home/saeed/Desktop
File Edit View Search Terminal Help
[root@localhost Desktop]# ls
esecurity.sh firestarter-1.0.3-1.i386.rpm Firestarter v.1
[root@localhost Desktop]#

```

که این دستور لیست فایل‌های موجود در دایرکتوری جاری را نمایش می‌دهد.

نکته: در خروجی این دستور، احتمالاً رنگ انواع مختلف فایل (فایل عادی، دایرکتوری، لینک و فایل‌های خاص) با هم تفاوت دارد.

بررسی چند آپشن کاربردی

آپشن F

این آپشن همانند دستور بالا عمل میکند با این تفاوت که در خروجی فایل‌ها، دایرکتوری‌ها و فایل‌های خاص را با درج یک کاراکتر در انتهای اسم آنها هنگام نمایش از هم جدا میکند. این کاراکتر برای دایرکتوری‌ها اسلش /، و برای فایل‌های خاص بسته به نوع ورژن ls ممکن است !، @، # باشد و برای فایل‌های معمولی هم چیزی درج نمیشود. برای اطلاعات دقیق از نوع کاراکتری که شل شما برای مشخص کردن فایل‌های خاص استفاده میکند میتوانید به `man ls` مراجعه کنید.

```

saeed@localhost:/home/saeed/Desktop
File Edit View Search Terminal Help
[root@localhost Desktop]# ls -F
esecurity.sh* firestarter-1.0.3-1.i386.rpm Firestarter v.1/
[root@localhost Desktop]#

```

آپشن l

با استفاده از این آپشن هر فایل را در یک خط نمایش داده می‌شود.

```

saeed@localhost:/home/saeed/Desktop
File Edit View Search Terminal Help
[root@localhost Desktop]# ls -l
total 400
-rwxrwxrwx. 1 root root    23 Apr  5 04:21 esecurity.sh
-rw-rw-r--. 1 saeed saeed 398106 Mar 31 05:10 firestarter-1.0.3-1.i386.rpm
drwxrwxr-x. 3 saeed saeed  4096 Mar 31 04:45 Firestarter v.1
[root@localhost Desktop]#

```

نکته: این آپشن معمولاً برای استفاده در حلقه‌ها در اسکریپت، یا برای فرستادن به دستور دیگر، مفید است. چون نام فایل/دایرکتوری ممکن است شامل فاصله باشد ولی احتمالاً شامل کاراکتر «سر خط» نیست (و اگر هم باشد، ls بجای آن علامت سوال نشان می‌دهد).



اشاره: در دستور ls امکان ادغام آپشن‌ها وجود به طور مثال دستورهای زیر با معال هم هستند:

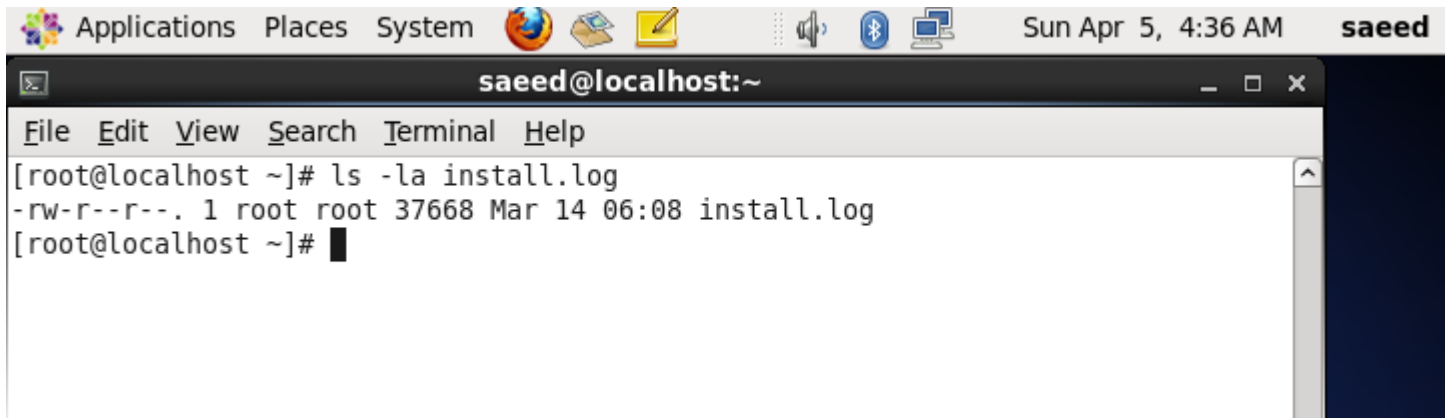
```

saeed@localhost:/home/saeed/Desktop
File Edit View Search Terminal Help
[root@localhost Desktop]# ls -Fal
total 408
drwxr-xr-x. 3 saeed saeed  4096 Apr  5 04:07 ./
drwx----- 27 saeed saeed  4096 Apr  5 04:02 ../
-rwxrwxrwx. 1 root root    23 Apr  5 04:21 esecurity.sh*
-rw-rw-r--. 1 saeed saeed 398106 Mar 31 05:10 firestarter-1.0.3-1.i386.rpm
drwxrwxr-x. 3 saeed saeed  4096 Mar 31 04:45 Firestarter v.1/
[root@localhost Desktop]#

```

بررسی نوع فایل

برای مشخص شدن نوع یک فایل از دستور ls به همراه آپشن l می‌توان استفاده کرد.



```

saeed@localhost:~
File Edit View Search Terminal Help
[root@localhost ~]# ls -la install.log
-rw-r--r--. 1 root root 37668 Mar 14 06:08 install.log
[root@localhost ~]#
  
```

در اینجا شما چندین کاراکتر (-) مشاهده میکنید که شروع با (-) نشان دهندهی فایل معمولی (regular file). برای فایل‌های ویژه کاراکتر اول یکی از موارد جدول زیر است:

- : Regular file
 l : Symbolic link
 c : Character special
 b : Block special
 p : Named pipe
 s : Socket
 d : Directory file

مثال :

```

saeed@localhost:/etc
File Edit View Search Terminal Help
drwxr-xr-x. 2 root root 4096 Mar 14 06:04 gre.d
-rw-r--r--. 1 root root 785 Mar 14 06:08 group
-rw-----. 1 root root 772 Mar 14 06:08 group-
lrwxrwxrwx. 1 root root 22 Mar 14 06:08 grub.conf -> ../boot/grub/grub.conf
-----. 1 root root 650 Mar 14 06:08 gshadow
-rw-----. 1 root root 640 Mar 14 06:08 gshadow-
drwxr-xr-x. 3 root root 4096 Mar 14 06:05 gtk-2.0
drwxr-xr-x. 3 root root 4096 Mar 14 06:03 hal
-rw-r--r--. 1 root root 26 Jan 12 2010 host.conf
-rw-r--r--. 1 root root 81 Apr 5 02:34 hosts
-rw-r--r--. 1 root root 466 Mar 27 14:12 hosts.allow
-rw-r--r--. 1 root root 540 Mar 27 14:59 hosts.deny
drwxr-xr-x. 2 root root 4096 Mar 14 05:58 hp
drwxr-xr-x. 4 root root 4096 Mar 14 06:03 httpd
drwxr-xr-x. 2 root root 4096 Mar 14 06:04 init
lrwxrwxrwx. 1 root root 11 Mar 14 05:59 init.d -> rc.d/init.d
-rw-r--r--. 1 root root 884 Mar 14 02:41 inittab
saeed@localhost:/etc #

```

که حرف d (در ابتدای خروجی) نشان می‌دهد که فایل از نوع دایرکتوری است.

نکته: آپشن d در دستور ls به این معنی است که اگر آرگومان داده شده دایرکتوری بود، جزئیات خود دایرکتوری را نشان دهد، نه محتویات آن را.

فایل‌های معمولی (regular)

متداول ترین نوع فایلی که با آن سروکار دارید، فایل‌های معمولی (regular file) هستند. در این نوع از فایل تقریباً هر نوع داده‌ای از قبیل یک متن، یک برنامه کاربردی، فیلم، فایل‌های باینری یا تصویر را میتوان ذخیره کرد.

یونیکس توانایی درک داده‌های موجود در یک regular file را ندارد. یک فایل معمولی میتواند به هر شکلی یک داده خام را ذخیره کند. زیرا یونیکس عمل تفسیر بر روی داده‌های یک فایل را انجام نمیدهد.

اشاره: معمولاً فایل‌های معمولی میتوانند اطلاعات اندکی از نوع داده‌هایی که در خودشان ذخیره کرده‌اند به شما بگویند. که برای این منظور دستور file بسیار مفید است.

شکل کلی دستور: file:

file filename

مثال:

```

saeed@localhost:/
File Edit View Search Terminal Help
[root@localhost ~]# file /etc
/etc: directory
[root@localhost ~]# file /etc/yum.conf
/etc/yum.conf: ASCII English text
[root@localhost ~]#

```



مجوزها، مالکها و گروههای یک فایل

مشخصه‌های مالک، مجوزهای دسترسی و گروه‌ها برای یک فایل یکی از ویژگی‌های بسیار مهم سیستم‌های یونیکسی است که روش‌های ایمنی را برای ذخیره کردن فایل‌ها مهیا میکند. هر فایل در یونیکس سه مشخصه مهم دارد :

- مجوز مالک آن
- مجوز گروه آن
- مجوز دیگران

مجوز مالک مشخص‌کننده‌ی عملیاتی است که مالک آن میتواند انجام دهد. مجوز گروه مشخص‌کننده‌ی عملیات قابل اجرا در گروهی که کاربر در آن قرار دارد و مجوز دیگران مشخص‌کننده‌ی عملیات قابل انجام برای کاربرانی است که عضو گروه مالک فایل نیستند .

همچنین برای هر فایل میتوان سه عمل را انجام داد :

- خواندن
- نوشتن
- اجرا کردن

که به ترتیب به معنی خواندن محتوای فایل، تغییر محتوای فایل و اجرای فایل به عنوان یک برنامه، می‌باشند .

مشاهده‌ی مجوزها

برای مشاهده‌ی مجوزهای یک فایل از دستور ls با آپشن l استفاده میشود .

نمادهای سه مجوز پایه‌ای که کمی قبل‌تر ذکر شدند به صورت زیر می‌باشند :

۱. خواندن فایل r
۲. نوشتن w
۳. اجرا کردن x

مثال :

```

saeed@localhost:/
File Edit View Search Terminal Help
[root@localhost /]# ls -la
total 98
dr-xr-xr-x. 22 root root 4096 Apr  5 04:03 .
dr-xr-xr-x. 22 root root 4096 Apr  5 04:03 ..
-rw-r--r--.  1 root root    0 Mar 31 04:20 .autofsck
dr-xr-xr-x.  2 root root 4096 Apr  5 03:23 bin
dr-xr-xr-x.  5 root root 1024 Mar 14 06:08 boot
drwxr-xr-x.  2 root root 4096 Nov 11 2010 cgroup
drwxr-xr-x. 17 root root 3740 Apr  5 03:23 dev
drwxr-xr-x. 98 root root 4096 Apr  5 04:03 etc
drwxr-xr-x.  3 root root 4096 Mar 14 06:08 home
dr-xr-xr-x. 17 root root 12288 Apr  5 03:23 lib
drwx-----.  2 root root 16384 Mar 14 05:57 lost+found
drwxr-xr-x.  2 root root 4096 Mar 25 13:37 media
drwxr-xr-x.  3 root root 4096 Mar 14 06:10 mnt
drwxr-xr-x.  2 root root 4096 Mar 14 02:41 opt
dr-xr-xr-x.  2 root root 4096 Mar 31 04:20 proc

```

نکته: برای هر فایل امکان اعطا یا لغو هر یک از این سه مجوز وجود دارد

مجوزهای دایرکتوری

مجوز اجرا (x): امکان دستیابی به یک دایرکتوری را مهیا می‌سازد. در صورتی که این مجوز به یک دایرکتوری داده نشود، مجوزهای خواندن و نوشتن تأثیری ندارند .

مجوز خواندن (r): امکان مشاهده‌ی لیست و صفت فایل‌های موجود در دایرکتوری با استفاده از دستور ls را مهیا می‌سازد .
 مجوز نوشتن (w): امکان اضافه یا حذف فایل در دایرکتوری را مهیا می‌سازد .

نکته: در صورتی که یک دایرکتوری تنها مجوز اجرا کردن داشته باشد، کاربر قادر مشاهده یا تغییر فایل‌های ذخیره شده در آن دایرکتوری نخواهد و تنها میتواند فایل‌های موجود در دایرکتوری را اجرا کند .

مجوزهای SUID و SGID

مکانیسمی برای دادن مجوزهای اضافی به برخی برنامه ها (SUID) set user id یا (SGID) set group id نام دارد. به عنوان مثال زمانی که شما برنامه‌ای را که suid آن فعال است اجرا میکنید، در واقع مجوزهای این برنامه از مجوزهای مالک برنامه ارث‌بری میشود. در واقع اگر فایلی دارای مجوز SUID باشد، کاربرانی که مجوز اجرای آن فایل را داشته باشند همانند مالک فایل در نظر گرفته می‌شوند. همچنین برای sgid هم همین رابطه برقرار است. یعنی برنامه‌هایی که بیت sgid مجوز آنها فعال است، مجوزهای خود را از گروه به ارث می‌برند و همانند گروه فایل در نظر گرفته می‌شوند.

نکته: بیت suid و sgid با حرف s نمایش داده میشود.

مثال :

```

saeed@localhost:~$ ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 27000 Aug 21 2010 /usr/bin/passwd
saeed@localhost:~$

```

که نمایان گر این است که بیت suid فعال است و مالک فایل ریشه است.

مجوز sticky یا sticky bit

این مجوز اصولاً به دایرکتوری‌ها داده می‌شود. فایل‌های درون دایرکتوری‌هایی که دارای این مجوز هستند تنها از طریق مالک آن فایل می‌توانند حذف شوند، حتی اگر آن فایل برای همه کاربران دارای مجوز write باشد. نکته: بیت sticky با حرف t نمایش داده میشود.

تعیین مجوزها برای فایل و دایرکتوری

شما میتوانید با استفاده از دستور chmod مجوزها را برای یک فایل یا دایرکتوری تغییر دهید. شکل کلی این دستور به صورت زیر است :

chmod expression files

که در آن expression میتواند یک عبارت سمبلیک یا یک عدد در مبنای هشت باشد.

روش سمبلیک

شکل کلی یک عبارت سمبلیک به صورت زیر است :

(permissions) (action) (who)

در این عبارت هر یک از بخش‌های who, action و permissions به صورت جداول زیر می‌باشند :

who:

- u : مالک
- g : گروه
- o : دیگر
- a : تمام کاربران

actions:

- + : برای اضافه کردن یک یا چند مجوز
- - : برای حذف مجوز یک یا چند مجوز
- = : مشخص کردن صریح مجوزها

permissions:

- r : خواندن
- w : نوشتن
- s : sgid , suid

t : sticky bit

- x : اجرا

تغییر مالک‌ها و گروه‌ها

دو دستور زیر برای این منظور در دسترس است :

chown
chgrp

که دستور اول (change owner) برای تغییر مالک و دستور دوم (change group) برای تغییر گروه به کار می‌رود .

مشاهده‌ی محتوای فایل‌ها

مشاهده‌ی لیست فایل‌ها قابلیت مفیدی است ولی ما در شل (shell) نیاز به مشاهده‌ی محتوای فایل‌ها هم داریم. برای این منظور یکی از دستوراتی که می‌توان استفاده کرد، دستور cat است .

شکل کلی دستور cat به صورت زیر است .

cat [option] filenames

به طور مثال :


```

Applications Places System Sun Apr 5, 4:58 AM saeed
saeed@localhost:/home/saeed/Desktop
File Edit View Search Terminal Help
[root@localhost Desktop]# cat /etc/yum.conf
[main]
cachedir=/var/cache/yum/$basearch/$releasever
keepcache=0
debuglevel=2
logfile=/var/log/yum.log
exactarch=1
obsoletes=1
gpgcheck=1
plugins=1
installonly_limit=5
bugtracker_url=http://bugs.centos.org/set_project.php?project_id=16&ref=http://b
ugs.centos.org/bug_report_page.php?category=yum
distroverpkg=centos-release

# This is the default, if you make this bigger yum won't see if the metadata
# is newer on the remote and so you'll "gain" the bandwidth of not having to
saeed@localhost:/hom...

```

همچنین میتوان از آپشن n یا b برای نمایش شماره هر خط استفاده کرد با این تفاوت که در آپشن b تنها خطوطی که محتوای آنها خالی نیست شمار گذاری میشود. به مثال توجه کنید :

```

Applications Places System Sun Apr 5, 5:00 AM saeed
saeed@localhost:/home/saeed/Desktop
File Edit View Search Terminal Help
[root@localhost Desktop]# cat -b /etc/yum.conf
1 [main]
2 cachedir=/var/cache/yum/$basearch/$releasever
3 keepcache=0
4 debuglevel=2
5 logfile=/var/log/yum.log
6 exactarch=1
7 obsoletes=1
8 gpgcheck=1
9 plugins=1
10 installonly_limit=5
11 bugtracker_url=http://bugs.centos.org/set_project.php?project_id=16&ref=
http://bugs.centos.org/bug_report_page.php?category=yum
12 distroverpkg=centos-release

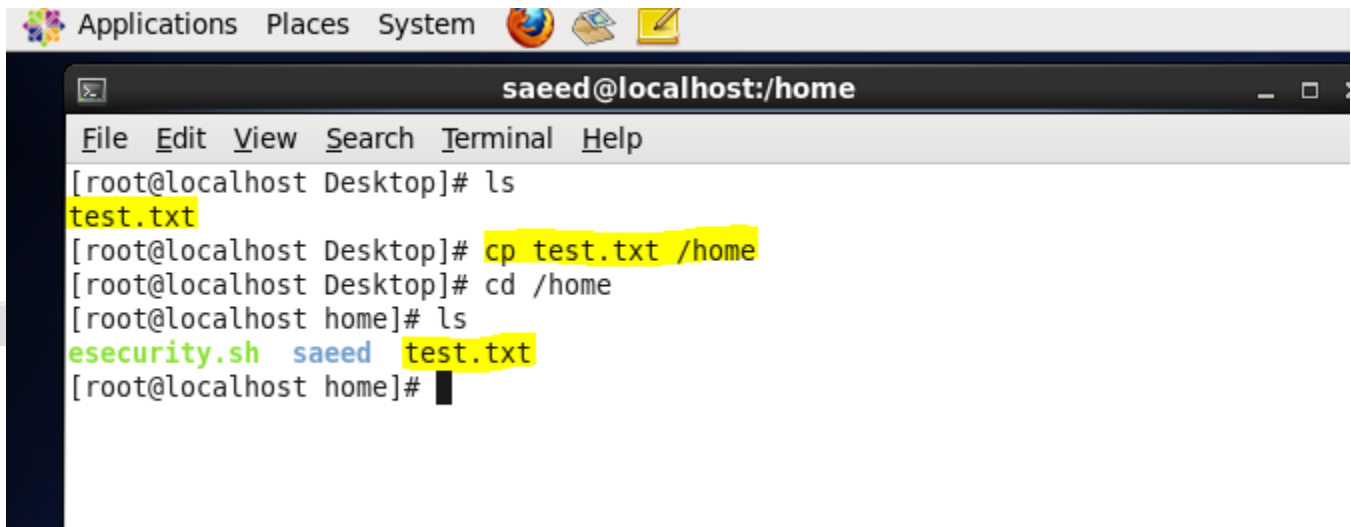
```

دستکاری فایل‌ها

دستکاری فایل‌ها شامل عملیات کپی، انتقال، تغییر نام و حذف می‌باشد که در ادامه دستورات مربوط به هرکدام مختصراً شرح داده شده است.

کپی کردن

شکل کلی دستور کپی (cp)

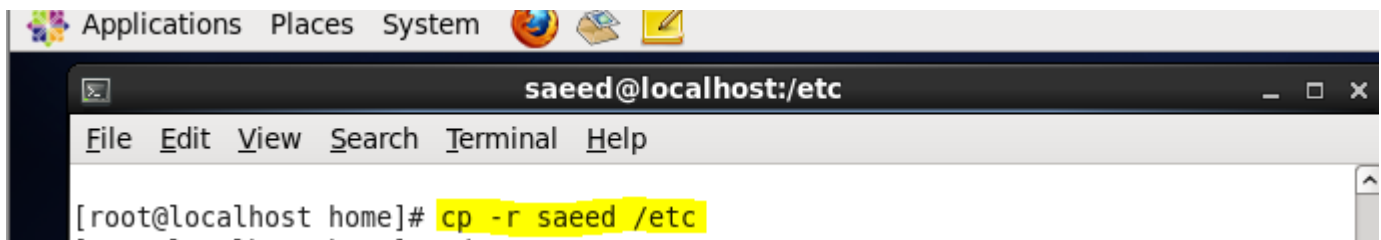


```

Applications Places System
saeed@localhost:/home
File Edit View Search Terminal Help
[root@localhost Desktop]# ls
test.txt
[root@localhost Desktop]# cp test.txt /home
[root@localhost Desktop]# cd /home
[root@localhost home]# ls
esecurity.sh saeed test.txt
[root@localhost home]#
  
```

کپی کردن یک دایرکتوری

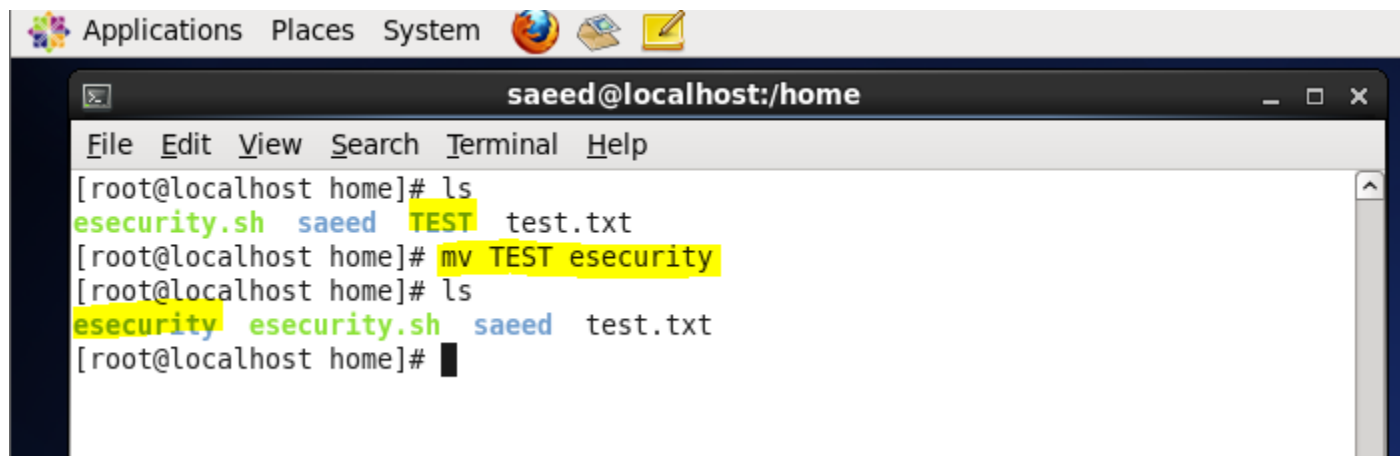
از آپشن `-r` برای این مقصود استفاده می‌شود



```

Applications Places System
saeed@localhost:/etc
File Edit View Search Terminal Help
[root@localhost home]# cp -r saeed /etc
  
```

تغییر نام و جابجا کردن (mv)



The screenshot shows a terminal window titled 'saeed@localhost:/home'. The terminal has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The command history is as follows:

```
[root@localhost home]# ls
esecurity.sh saeed TEST test.txt
[root@localhost home]# mv TEST esecurity
[root@localhost home]# ls
esecurity esecurity.sh saeed test.txt
[root@localhost home]#
```



فصل چهارم - دایرکتوری‌ها

مقدمات

سیستم‌های مبتنی بر یونیکس برای مدیریت دایرکتوری‌ها و فایل‌ها از ساختارهای سلسه‌مراتبی که به صورت یک درخت وارون پیاده‌سازی می‌شود، استفاده می‌کنند که ریشه‌ی (root) این درخت با کاراکتر اسلش (/) مشخص می‌شود. و تمام دایرکتوری‌ها زیر شاخه‌های ریشه هستند. شما می‌توانید از هر زیر دایرکتوری ریشه برای ذخیره اطلاعات خود استفاده کنید. هر فایل در یک دایرکتوری و هر دایرکتوری (به جز دایرکتوری ریشه) در یک دایرکتوری دیگر ذخیره می‌شود.

اشاره: این الگو متفاوت از سیستم‌های مبتنی بر ویندوز یا مک است که چندین دایرکتوری (همانند cd-rom floppy drive یا هارد دیسک) می‌توانند در بالاترین سطح قرار داشته باشند.

تعریف: برای آشنایی با اصطلاحات دایرکتوری والد (parent directory) و دایرکتوری فرزند (subdirectories) یا (child) مثال زیر را در نظر بگیرید.

فرض کنید دو دایرکتوری با نام‌های A و B وجود دارد به طوری که دایرکتوری A محتوی دایرکتوری B نیز هست. در این صورت به A دایرکتوری والد و به B فرزند یا زیردایرکتوری A میگویند.

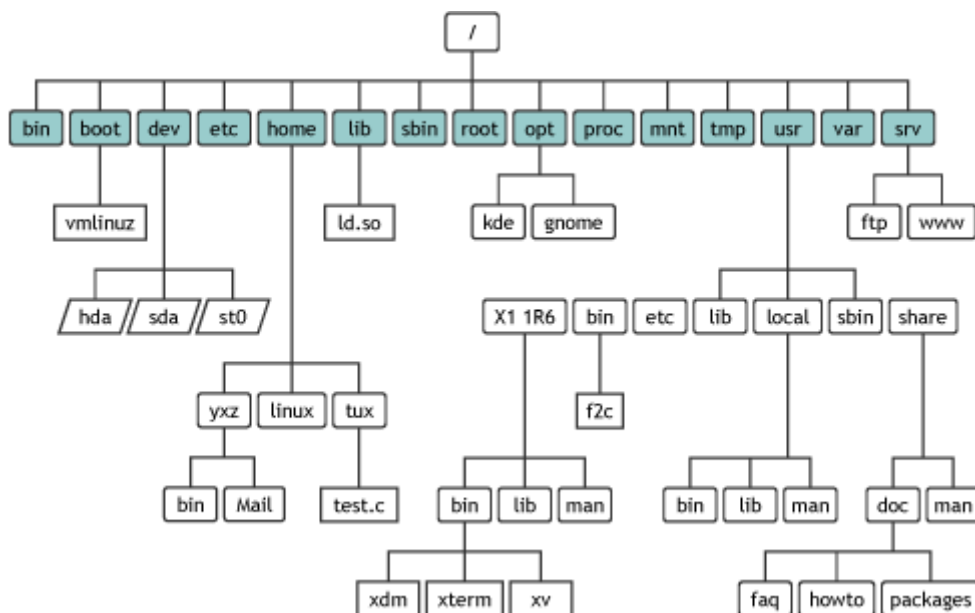
نکته: عمق درخت شامل دایرکتوری‌ها محدود و حداکثر ۱۰۲۴ است.

ساختار سلسله مراتبی دایرکتوری ها در لینوکس

هر کاربر و تمامی مدیران لینوکسی می بایست با این ساختار آشنایی داشته باشند. هر دایرکتوری به منظور خاصی ایجاد و استفاده می شود. دایرکتوری ها می توانند درون یک دایرکتوری دیگری قرار گیرند که “زیر دایرکتوری” گفته می شوند. در تمامی سیستم های عامل ساختار دایرکتوری ها شبیه به یک درخت وارونه است به طوری که ریشه یا root درخت در بالاترین سطح قرار دارد.

میان دایرکتوری و زیر دایرکتوری های درون آن یک رابطه والد و فرزندی وجود دارد که تمامی شاخه ها یا فرزندان ریشه هستند و به همین خاطر است که در پیمایش سیستم فایل در سیستم عامل های شیه یونیکس، مسیر پیمایش همیشه به نماد / که معرف ریشه یا root است آغاز می شود.

در ادامه اساسی ترین دایرکتوری ها در سیستم عامل لینوکس توضیح داده شده اند.



در ادامه اساسی ترین دایرکتوری ها در سیستم عامل لینوکس توضیح داده شده اند.

Root Directory یا دایرکتوری /

- ریشه درخت سیستم فایل لینوکسی و سر آغاز تمامی مسیر های منتهی به یک دایرکتوری یا فایل در پیمایش سیستم فایل.
- تنها کاربر root مجوز نوشتن بر روی این دایرکتوری را دارد. (مانند دستور زیر)
- این دایرکتوری به عنوان دایرکتوری خانگی کاربر root نیست بلکه دایرکتوری /root دایرکتوری خانگی کاربر root است.

`mkdir /directory_under_root_directory`

دایرکتوری /bin

- شامل تمامی فایل های باینری قابل اجرا (فایل های باینری که مجوز x یا executable را دارند).
- دستور های مورد نیاز برای حالت تک کاربره در این دایرکتوری قرار دارند. دستور های زیر این دایرکتوری تحت مالکیت کاربر و گروه کاربری root هستند اما تمامی کاربران قادر به اجرای آنها هستند. به طور مثال دستور `free` ، `touch` یا `pwd`

دایرکتوری /sbin

- همانند دایرکتوری بالا این دایرکتوری نیز شامل فایل های باینری (دستورهای) قابل اجرا است.
- دستور های درون این دایرکتوری به منظور مدیریت سیستم به کار رفته و توسط مدیر یا مدیران سیستم استفاده می شوند. مانند دستور های `fdisk`، `mkfs`

دایرکتوری /root

- این دایرکتوری، دایرکتوری خانگی کاربر root است.

دایرکتوری /home

- دیگر کاربران در زیر دایرکتوری /home یک دایرکتوری همنام با نام کاربری خود دارند.

دایرکتوری /etc

- این دایرکتوری محل ذخیره فایل های پیکربندی سیستم است. به طور مثال فایل پیکربندی سرویس `dhcp` بخش نخست – بخش دوم)
- همچنین در نسخه هایی از لینوکس که از فرایند `init` به منظور کنترل فرایند و سرویس ها استفاده می کنند، اسکریپت های `init` در زیر دایرکتوری `etc/init.d` قرار دارند.

دایرکتوری /proc

- در زیر این دایرکتوری به ازای هر فرایند فعال کنونی در سیستم یک زیر دایرکتوری وجود دارد.
- فایل های وضعیتی سیستم و کرنل لینوکس در زیر این دایرکتوری هستند.

دایرکتوری /boot

- شامل دایرکتوری ها و فایل هایی مرتبط با بوت شدن سیستم عامل مانند : فایل تنظیمات گراب `vmlinux image` – و فایل های دیگر با هسته لینوکس.

دایرکتوری /tmp

- برخی از برنامه های کاربردی مانند اوراکل یا `vmware` نیاز به محلی برای ایجاد فایل های موقتی دارند که این محل به اندازه ای دلخواه به طور پیش فرض دایرکتوری `tmp` است.
- فایل های زیر این دایرکتوری پس از `reboot` شدن سیستم به طور خودکار حذف می شوند.

دایرکتوری /var

- دایرکتوری برای فایل های متغیر یا variable files
- در زیر این دایرکتوری فایل هایی مانند log ها که به سرعت رشد می کنند قرار می گیرند.

دایرکتوری /dev

- محل قرارگیری فایل های مرتبط به هر device. مانند فایل sda که اشاره به اولین دیسک متصل به سیستم می کند.

دایرکتوری /lib

- این دایرکتوری شامل فایل های کتابخانه ای به زبان سی که توسط دستور های زیر دایرکتوری های bin/ و sbin/ استفاده می شوند.

دایرکتوری /mnt

- نقاط اتصال به طور معمول در زیر این دایرکتوری متصل یا mount می شوند.

دایرکتوری /media

- محلی برای mount شدن CD/DVD ROM

دایرکتوری /usr

- دستور های پایه ای لینوکس مانند passwd در زیر دایرکتوری های این دایرکتوری قرار دارند. دو زیر دایرکتوری usr/bin/ و usr/sbin/
- فایل های راهنما و مستندات زیر دایرکتوری /usr/share/doc/

دایرکتوری /srv

- این دایرکتوری برای داده های سرویس های فراهم شده توسط سیستم (Service data) استفاده می شوند.

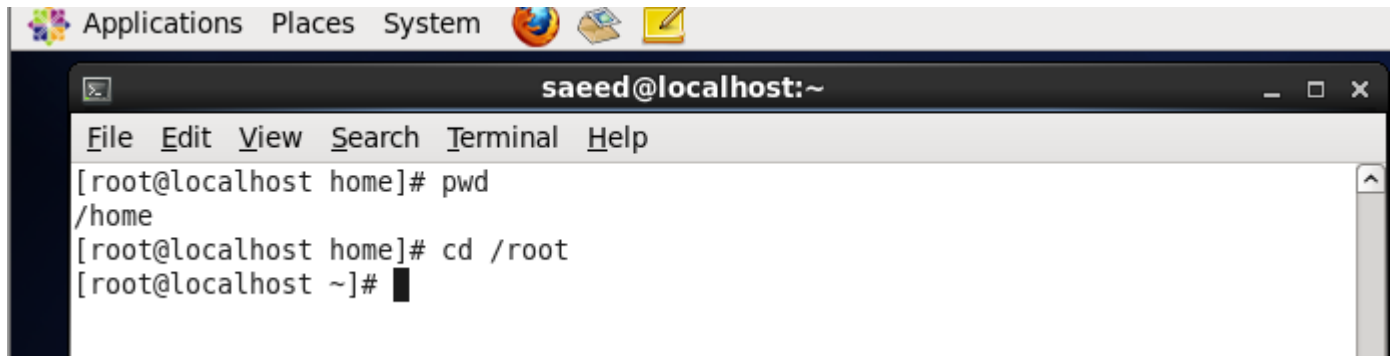
فرض کنید لینوکس را بر روی یک هارد می خواهیم نصب کنیم. در زمان نصب می توانیم دایرکتوری های زیر را به طور پیشفرض و در زمان نصب جدا از هم و بر روی پارتیشن های مجزایی نصب کنیم. به طور معمول لینوکس برای نصب شدن و بدون پارتیشن بندی سفارشی شده از سوی شما سه دایرکتوری / و دایرکتوری /home و swap را بر روی سه پارتیشن مجزا ایجاد می کند). البته swap یک دایرکتوری نیست و تنها یک سیستم فایل است. (همچنین شما می توانید دایرکتوری های زیر را در زمان نصب بر روی پارتیشن های مجزا ایجاد کنید.

- دایرکتوری /
- دایرکتوری /boot
- دایرکتوری /home
- دایرکتوری /tmp
- دایرکتوری /usr
- دایرکتوری /var
- دایرکتوری /opt
- دایرکتوری /usr/local

همچنین هر دایرکتوری که جدا بر روی یک پارتیشن متصل می شود می تواند نوع پارتیشن آن با نوع دیگر پارتیشن ها و همچنین با نوع پارتیشن / متفاوت باشد ولی یک نکته مهم ایت و آن اینکه پارتیشن های مجزا به ازای دایرکتوری ها دلیل بر برهم زدن ساختار نک درختی با شروع از ریشه root یا دایرکتوری / نیست بلکه باز هم همه ی این پارتیشن ها و دایرکتوری ها در پیمایش سر آغاز آنها دایرکتوری / یا دایرکتوری root است.

جابجایی بین دایرکتوری‌ها

ساده‌ترین راه برای مشخص کردن دایرکتوری خانه (home) از استفاده از دستور `cd` و بعد از آن دستور `pwd` است. دستور `cd` شما را به دایرکتوری خانه هدایت میکند و دستور `pwd` مسیر آن را نمایش میدهد.



```

saeed@localhost:~
File Edit View Search Terminal Help
[root@localhost home]# pwd
/home
[root@localhost home]# cd /root
[root@localhost ~]#
  
```

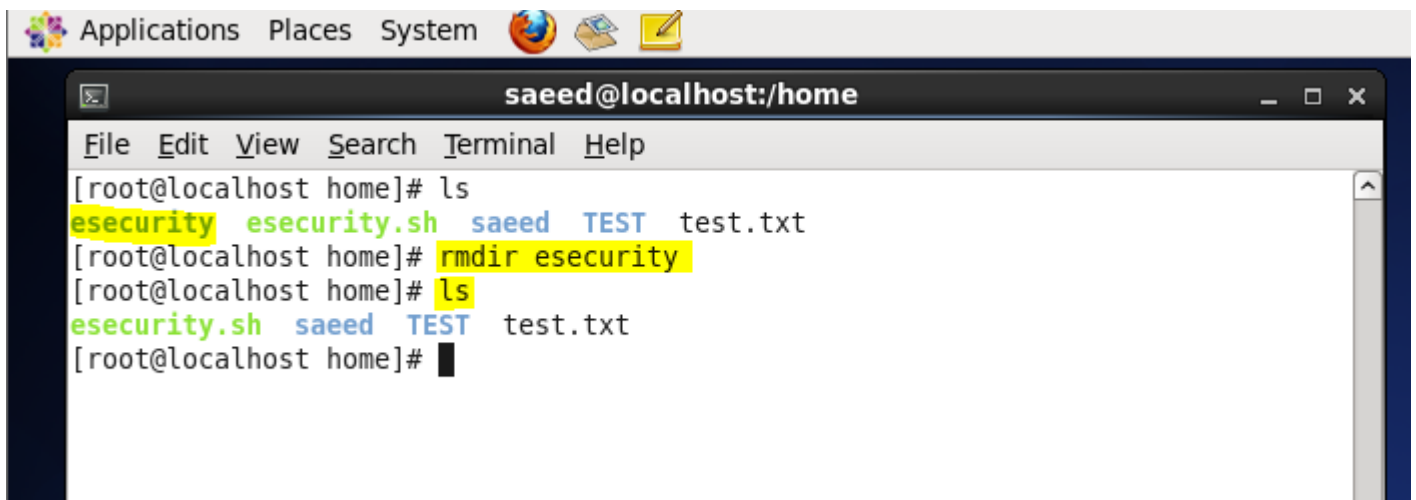
خطاهای متداولی که در استفاده از این دستور رخ می‌دهد :

۱. تلاش برای تغییر مسیر به یک فایل
۲. تلاش برای تغییر مسیر به یک دایرکتوری ناموجود
۳. تلاش برای تغییر مسیر به یک دایرکتوری که مجوز اجرا توسط کاربر را ندارد

حذف یک دایرکتوری

برای این منظور میتوان از دستور زیر استفاده کرد :

دستور اول شکل ایمن‌تر دستور حذف است به این دلیل که تنها دایرکتوری‌هایی که محتوای آنها خالی هست پاک میشود



```

saeed@localhost:/home
File Edit View Search Terminal Help
[root@localhost home]# ls
esecurity esecurity.sh saeed TEST test.txt
[root@localhost home]# rmdir esecurity
[root@localhost home]# ls
esecurity.sh saeed TEST test.txt
[root@localhost home]#
  
```

دستور دوم دایرکتوری را به همراه تمام محتوای آن پاک میکند .

```

saeed@localhost:/home
File Edit View Search Terminal Help
[root@localhost home]# ls
esecurity.sh saeed TEST test.txt
[root@localhost home]# cd TEST/
[root@localhost TEST]# ls
saeed.txt
[root@localhost TEST]# cd ..
[root@localhost home]# rm -r TEST/
rm: descend into directory `TEST'? y
rm: remove regular empty file `TEST/saeed.txt'? y
rm: remove directory `TEST'? y
[root@localhost home]# ls
esecurity.sh saeed test.txt
[root@localhost home]#

```

توجه: در شکل دوم، برگرداندن اطلاعات پاک شده بسیار سخت است.

خطاهای رایجی که در استفاده از دستور rmdir ممکن است رخ دهد:

- تلاش برای حذف یک فایل با استفاده از rmdir
- تلاش برای حذف یک دایرکتوری که خالی نیست
- تلاش برای حذف یک دایرکتوری که مجوز نوشتن توسط کاربر را ندارد

تنها خطای رایجی که در استفاده از دستور rm -r ممکن است رخ دهد وجود نداشتن فایل یا دایرکتوری مورد نظر، یا نداشتن مجوز است.

فصل پنجم - پروسس‌ها

مقدمه

شما در این فصل مفهوم process و job در یک سیستم یونیکسی را خواهید آموخت. در یونیکس هر برنامه‌ای به عنوان یک process اجرا میشود. همچنین در این فصل به تعریف اصطلاحات foreground و background پرداخته شده و چگونگی شروع یک process شرح داده میشود، در ادامه نحوه‌ی kill کردن process ها و در نهایت مفهوم والد و فرزند برای یک process توضیح داده شده است

شروع یک process

شما در فصل‌های قبل با برخی از دستورات یونیکس آشنا شدید. هر زمان که شما یک دستور را اجرا میکنید در واقع در حال شروع کردن یک process هستید. به عنوان مثال وقتی شما دستور ls را اجرا میکنید در واقع یک process را شروع کرده‌اید. سیستم برای هر process یک شناسه پنج رقمی در نظر میگیرد که به آن process id یا pid میگویند. هر process در یونیکس یک شناسه منحصر به فرد دارد. و در هر زمان هر pid تنها به یک process اختصاص پیدا میکند. در واقع pid یک عدد ۱۶ بیتی است که میتواند حداکثر تا ۳۲۷۶۷ باشد.

نمایش پروسس‌ها

شما در با وارد کردن دستور تاپ top می‌توانید کلیه پردازش‌هایی را که در سرور شما فعال می‌باشد را مشاهده نمائید. همچنین با وارد کردن دستور M با حروف بزرگ می‌توانید لیست پروسس‌ها را بر اساس میزان رم مصرفی منظم نمائید. با دستور O با حروف بزرگ نیز کلیه حالات مختلف چینش پروسس‌ها را می‌توانید مشاهده نمائید:

```

saeed@localhost:/home
File Edit View Search Terminal Help
[root@localhost home]# top

top - 05:29:18 up 6:19, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 136 total, 1 running, 135 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.3%us, 0.7%sy, 0.0%ni, 98.7%id, 0.0%wa, 0.0%hi, 0.3%si, 0.0%st
Mem: 1031316k total, 683664k used, 347652k free, 64248k buffers
Swap: 2064376k total, 0k used, 2064376k free, 426332k cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
  622 root        20   0     0     0     0  S   0.3   0.0   0:03.37 flush-8:0
 2220 root        20   0 63820  20m 7776  S   0.3   2.0   0:11.49 Xorg
 6685 root        20   0  2656  1108  860  R   0.3   0.1   0:00.28 top
    1 root        20   0  2828  1400 1192  S   0.0   0.1   0:01.71 init
    2 root        20   0     0     0     0  S   0.0   0.0   0:00.00 kthreadd
    3 root        RT    0     0     0     0  S   0.0   0.0   0:00.00 migration/0
    4 root        20   0     0     0     0  S   0.0   0.0   0:00.04 ksoftirqd/0
    5 root        RT    0     0     0     0  S   0.0   0.0   0:00.00 watchdog/0
    6 root        20   0     0     0     0  S   0.0   0.0   0:00.07 events/0
    7 root        20   0     0     0     0  S   0.0   0.0   0:00.00 cpuset
    8 root        20   0     0     0     0  S   0.0   0.0   0:00.00 khelper

```

دو روش برای شروع هر process وجود دارد. اجرای آن در foreground یا background.

Foreground Processes

به طور پیش فرض تمام دستورات به این طریق اجرا میشوند. در صورت نیاز از ورودی میخوانند یا در خروجی مینویسند. به عنوان مثال پس از اجرای دستور `ls` خروجی دستور بر روی صفحه نمایش هدایت می‌شود. (هدایت ورودی و خروجی‌ها در فصل ۱۳ به طور کامل بررسی میشود). در زمانی که این دستور در حال اجرا است، شما نمیتوانید هیچ دستور دیگری را شروع کنید. البته در این حالت میتوان دستورات را وارد کرد و سیستم عامل دستورات را به عنوان buffer ذخیره میکند تا دستور فعلی پایان یابد، ولی تا زمانی که اجرای دستور حاضر پایان نیابد، اجرای هیچ دستور دیگری شروع نمیشود. خوشبختانه در یونیکس امکان فرستادن دستورات به background یا تعلیق آنها وجود دارد.

Background Processes

در این حالت دستور در پس زمینه اجرا می‌شود و شما قادر به وارد کردن دستورات بعدی هستید. مزیت استفاده از این روش در اینست که شما نیاز به صبر کردن برای اتمام دستور جاری ندارید. ساده ترین راه برای اجرای یک دستور در background استفاده از کاراکتر & در آخر هر دستور است.

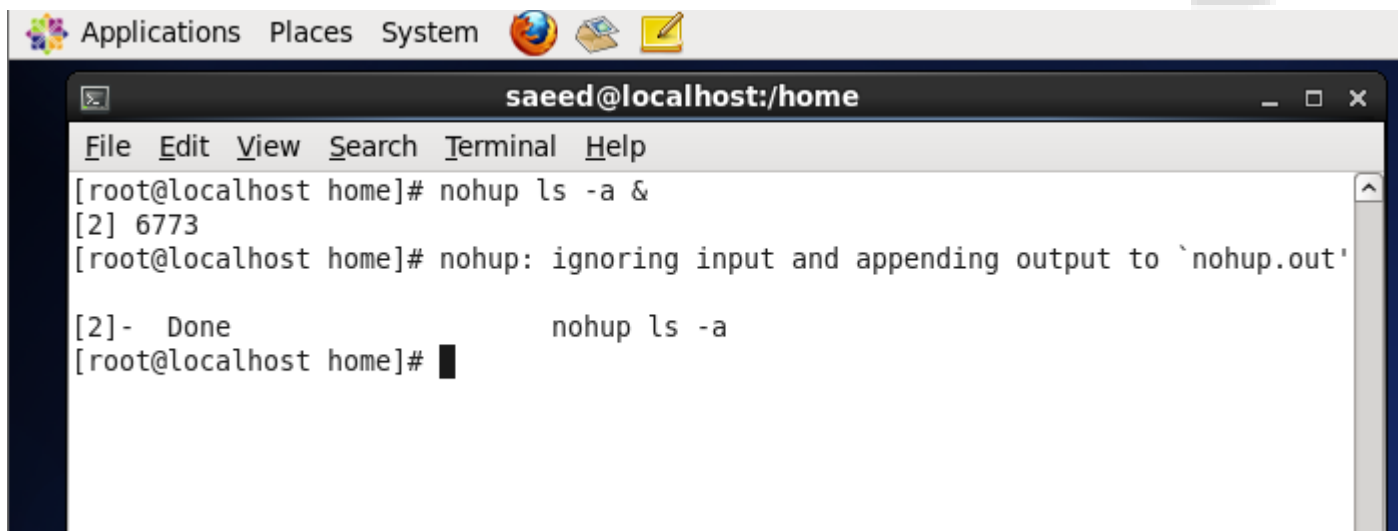
نکته: در صورتی که دستوری که به background فرستاده شده نیاز به ورودی داشته باشد، اجرای آن متوقف شده و تا زمانی که ورودی خود را دریافت کند منتظر میماند.

فرستادن یک دستور از foreground به background

علاوه بر اینکه میتوان دستورات را با استفاده از & در background اجرا کرد، میتوان دستورات foreground را نیز به background فرستاد. برای این منظور میتوان از کلیدهای ترکیبی ctrl+z استفاده نمود که موجب تعلیق و فرستادن process به background میشود. در این حالت دستور در حافظه قرار دارد ولی پردازشی بر روی آن انجام نمیشود.

دستور nohup

دستور nohup یا "no hang up" برای محافظت از اجرای یک دستور زمانی که به هر دلیل در حین اجرای دستور، سیستم شما log off یا disconnect میشود، میتوان استفاده کرد. شکل کلی این دستور به صورت زیر است:



```

Applications Places System
saeed@localhost:/home
File Edit View Search Terminal Help
[root@localhost home]# nohup ls -a &
[2] 6773
[root@localhost home]# nohup: ignoring input and appending output to `nohup.out'

[2]- Done
nohup ls -a
[root@localhost home]#
  
```

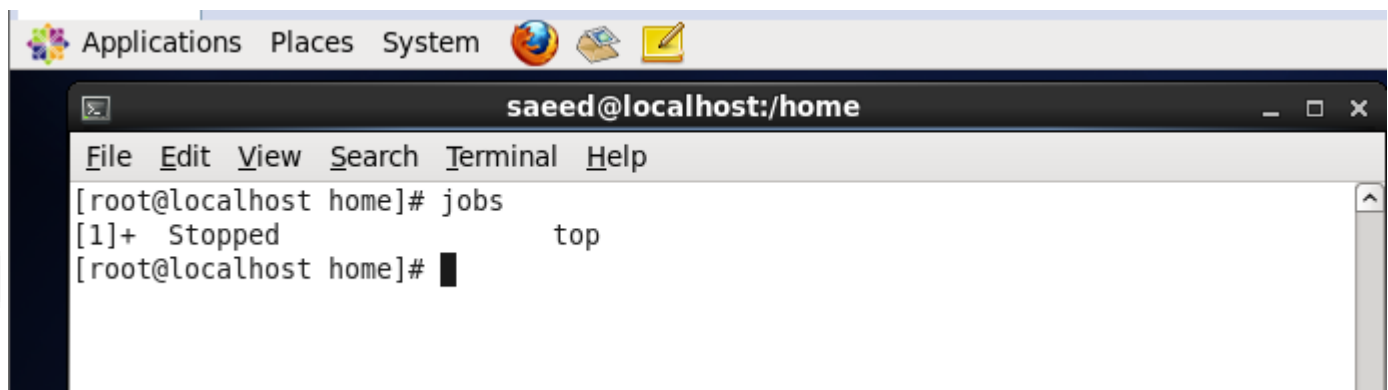
که در این حالت خروجی دستور در فایل nohup.out ذخیره خواهد شد.

لیست process های در حال اجرا

برای این منظور میتوان از دو دستور jobs و ps استفاده کرد .

دستور jobs

از این دستور برای نمایش process های تعلیق شده یا موجود در background استفاده میشود

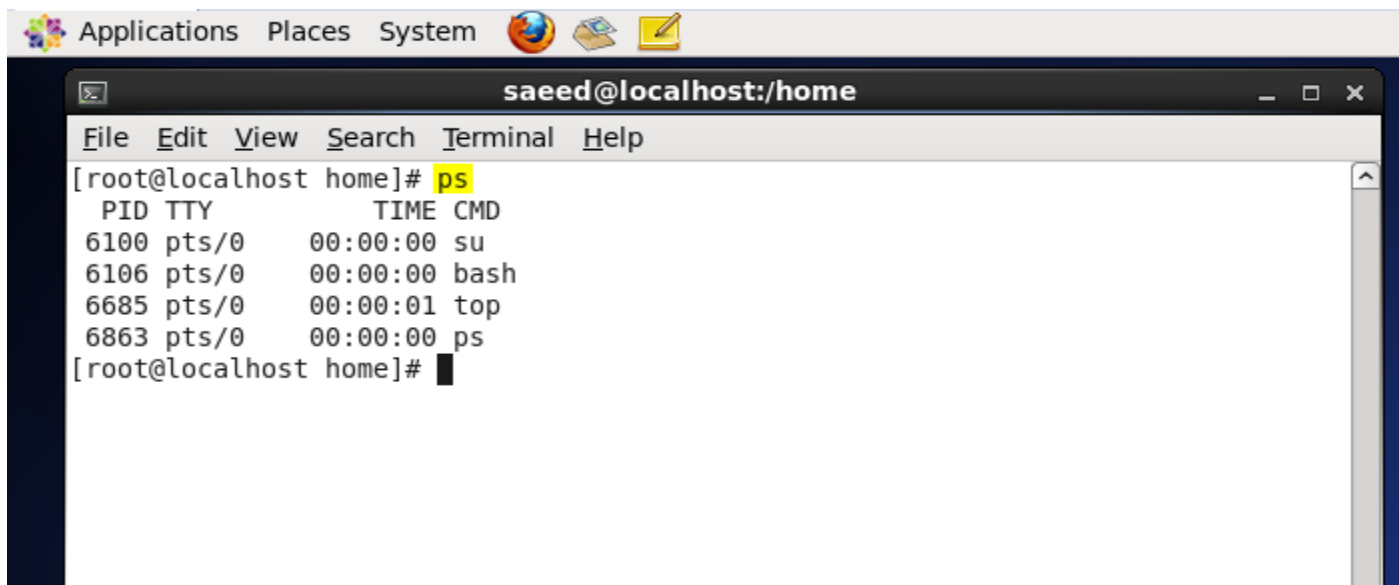


```

saeed@localhost:/home
File Edit View Search Terminal Help
[root@localhost home]# jobs
[1]+  Stopped                  top
[root@localhost home]#
  
```

دستور PS

از این دستور میتوان برای نمایش تمام process های در حال اجرا استفاده کرد .



```

saeed@localhost:/home
File Edit View Search Terminal Help
[root@localhost home]# ps
  PID TTY          TIME CMD
  6100 pts/0        00:00:00 su
  6106 pts/0        00:00:00 bash
  6685 pts/0        00:00:01 top
  6863 pts/0        00:00:00 ps
[root@localhost home]#
  
```

در این حالت برای هر دستور چهار ستون اطلاعات مشخص میشود .
 اولین ستون pid پروسه است .
 در ستون دوم tty به معنی اجرای دستور به وسیلهی ترمینال است.
 در ستون سوم time نشان‌دهندهی زمان پردازش cpu برای انجام این process است .
 و در آخر نام process مشخص شده است .

استفاده از f

این حالت نیز مشابه حالت قبل دستوراتی که توسط شما آغاز شده‌اند را نمایش میدهد با این تفاوت که هشت ستون از اطلاعات را برای هر process نمایش میدهد.

```

saeed@localhost:/home
File Edit View Search Terminal Help
[root@localhost home]# ps -f
UID      PID    PPID  C  STIME TTY      TIME CMD
root     6100   6081  0  04:32 pts/0    00:00:00 su
root     6106   6100  0  04:32 pts/0    00:00:00 bash
root     6685   6106  0  05:28 pts/0    00:00:01 top
root     6889   6106  0  05:48 pts/0    00:00:00 ps -f
[root@localhost home]#
  
```

چهار ستون جدید در این حالت به شرح زیر هستند :

۱. uid : User id
۲. ppid : parent process id در بخش بعد توضیح داده میشود
۳. cpu : درصد استفاده از
۴. stime : زمان شروع پردازش

توجه کنید مقدار عدد ppid برای تمام پردازش‌ها برابر با pid پروسه bash هستند. یا به عبارتی bash والد تمام این process‌ها میباشد .

متوقف کردن یک پروسس
 زمانیکه از دستور top استفاده می‌کنید، می‌توانید با استفاده از دستور k یک پروسس خاص را در صورتیکه از دسترسی لازم برخوردار باشید، حذف نمایید.
 برای این کار باید پس از آن شماره پردازش (PID) مورد نظر را وارد نمایید.

```

saeed@localhost:~/Desktop
File Edit View Search Terminal Help
[saeed@localhost Desktop]$ kill 7253
[saeed@localhost Desktop]$

```

نمایش پردازش های یک کاربر خاص
با دستور top -u saeed شما می توانید پردازش های یک کاربر خاص را مشاهده نمایید.

Top -u saeed

```

saeed@localhost:~/Desktop
File Edit View Search Terminal Help

top - 05:57:55 up 6:47, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 137 total, 1 running, 135 sleeping, 1 stopped, 0 zombie
Cpu(s): 0.3%us, 0.3%sy, 0.0%ni, 99.0%id, 0.0%wa, 0.0%hi, 0.3%si, 0.0%st
Mem: 1031316k total, 682300k used, 349016k free, 65540k buffers
Swap: 2064376k total, 0k used, 2064376k free, 424964k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU  %MEM    TIME+  COMMAND
 6930 saeed     20   0 41616 3024 2480 S   0.0   0.3   0:00.04 bonobo-activati
 7036 saeed     20   0 38740 3400 2940 S   0.0   0.3   0:00.00 gnome-keyring-d
 7045 saeed     20   0 29380 5860 4896 S   0.0   0.6   0:00.06 gnome-session
 7054 saeed     20   0  3568  736  520 S   0.0   0.1   0:00.00 dbus-launch
 7055 saeed     20   0 13572 1464  800 S   0.0   0.1   0:00.08 dbus-daemon
 7073 saeed     20   0  8444 4108 2036 S   0.0   0.4   0:00.15 gconfd-2
 7080 saeed     20   0  120m 8676 7120 S   0.0   0.8   0:00.11 gnome-settings-
 7082 saeed     20   0 28068 7184 5736 S   0.0   0.7   0:00.03 seahorse-daemon
 7089 saeed     20   0  7020 1904 1668 S   0.0   0.2   0:00.00 gvfsd
 7097 saeed     20   0 99428  10m 8548 S   0.0   1.0   0:00.12 metacity
 7110 saeed      9 -11 98.0m 4476 3488 S   0.0   0.4   0:00.28 pulseaudio
 7113 saeed     20   0 44004  12m   9m S   0.0   1.2   0:00.16 gnome-panel
 7114 saeed     20   0  9668 2292 1964 S   0.0   0.2   0:00.00 gconf-helper
 7115 saeed     20   0 83148  19m  13m S   0.0   1.9   0:00.74 nautilus
 7117 saeed     20   0 41616 3188 2556 S   0.0   0.3   0:00.02 bonobo-activati
 7121 saeed     20   0  102m  10m 8400 S   0.0   1.0   0:00.07 gnome-volume-co

```

تنظیمات پنل top

برای نمایش تعداد بیشتر پردازش ممکن است نیاز داشته باشید برخی از اطلاعاتی که در بالای پنل top نمایش داده می شود ، را پنهان نمایید، برای این کار می توانید از دستورات زیر استفاده نمایید، البته امکان افزایش سایز پنجره شل نیز راه دیگری خواهد بود.

- دستور l که اطلاعات میانگین بار را در سطر اول نمایش می دهد، و از این دستور می توانید برای نمایش یا پنهان کردن آن استفاده نمایید.
- دستور t که اطلاعات پردازشگر را در سطر دوم و سوم نمایش می دهد، و از این دستور می توانید برای نمایش یا پنهان کردن آن استفاده نمایید.
- دستور m که اطلاعات حافظه (memory) را در سطر چهارم و پنجم نمایش می دهد، و از این دستور می توانید برای نمایش یا پنهان کردن آن استفاده نمایید.

ذخیره تنظیمات

برای ذخیره تنظیماتی که در نرم افزار top وارد کرده اید می توانید از دستور w استفاده کنید و از دفعه بعد با همان تنظیمات قبلی از top استفاده نمایید. این اطلاعات در مسیر ~/toprc ذخیره می گردند.

برای خروج از برنامه بدون ذخیره تغییرات نیز می توانید از دستور q استفاده کن



فصل ششم - متغیرها

متغیرها محل هایی از حافظه اند که مقداری را در خود نگه می دارند. دارای نام، مقدار و نوع هستند. با نام متغیر به محل حافظه ای که مقدار را در خود یا هر پوسته دیگر یونیکسی نیازی به تعیین نوع ندارند، بلکه با اختصاص مقدار به متغیر، نوع آن معلوم BASH نگه داشته مراجعه می کنیم. متغیرهای یا هر پوسته دیگری دو BASH می شود. متغیرها می توانند مقادیر عددی، رشته ای، آرایه ای و یا حتی خروجی یک دستور را در خود نگه دارند. در متغیرهای محیطی دارای کاربرد خاص هستند و برای منظورهای خاصی وجود. نوع متغیر (۱) محلی و (۲) قابل تعریف توسط کاربر، وجود دارد که مسیر دایرکتوری های برنامه ها و اسکریپت ها را دارد. هر کاربر متغیرها را با مقادیر مربوط به خود دارد. بطور مثال PATH دارند. مانند متغیر با دیگر کاربران متفاوت است، ولی هر کاربر متغیر خودش را دارد که root برای تمامی کاربران عادی یکسان است و برای کاربر PATH مقدار متغیر که مسیر دایرکتوری خانگی کاربر فعلی (کاربری که در حال حاضر در سیستم HOME می تواند که می تواند مقدار آنرا تغییر دهد. مثال دیگر متغیر است) را نگه می دارد و مقدار این متغیر برای هر کاربری با کاربر دیگر متفاوت است LOGIN.

می کنید مقدار دهی می شوند و توسط شما یا login می توان دید. این نتغیرها زمانی که به سیستم printenv,env متغیرهای محلی را با دستورهای وجود PATH برای اینکه ایا مسیر منتهی به دستور میان مسیرهای متغیر PATH خود سیستم عامل استفاده می شوند. بطور مثال سیستم عامل از متغیر است UserID که مخفف UID یا شما می توانید در اسکریپت نویسی از این متغیرها استفاده کنید مانند متغیر (اطلاعات بیشتر). دارد یا نه استفاده می کند نبود مجاز به اجرای root است یا نه؟ اگر root برای تعیین کاربر استفاده کنید. بطور مثال می توانید برای اجرای اسکریپت کاربر را چک کنید که ایا اسکریپت نباشد

```

1  #!/bin/bash
2
3  # baresi mikonad aya moteghayer $UID karbar mosavi 0 hast ya na
4  # agar mosavi 0 nabood yek peygham داده mishavad va ba dastoor
5  # exit 1 az edameh ejraye script jologiri mikonad
6
7  if [ $UID != 0 ]
8  then
9      echo "Shoma root nistid va mojaz be ejraye script nistid!"
10     exit 1
11 fi

```


نوع دیگر متغیر هایی هستند که خود شما ایجاد می کنید. شکل کلی تعریف یک متغیر بصورت زیر است و نباید میان علامت = و عبارت های سمت چپ و راست فاصله ای باشد. مقدار یا value می تواند هر مقدار عددی، رشته ای و حتی خروجی یک دستور دیگر باشد. در تعریف متغیر ها باید دقت کنید که بزرگی و کوچکی نام متغیر ها مهم است و بطور مثال myvar با MyVar متفاوت است ولی در نامگذاری آنها نباید اعداد در ابتدای نام متغیر بیایند و همچنین در نامگذاری آنها از کاراکتر های خاص مانند @%^& استفاده نکنید. کاراکتر \$ تعیین کننده متغیر است یعنی برای استفاده و چاپ متغیر باید علامت \$ در ابتدای نام متغیر بیاید) مانند \$UID در شکل بالا (ولی در هنگام تعریف متغیر علامت \$ آورده نمی شود. اط خط های زیر، خط اول شکل تعریف یک متغیر و خط های بعد مثال هایی هستند:

```
variable_name=value
```

```
myvar1=10
```

```
myvar2=25.3
```

```
“myvar3=”my name is Saeed
```

```
`myvar4=`ls -l
```

شکل دیگر تعریف یک متغیر که مقدارش خروجی یک دستور است بصورت زیر است:

```
(myvar4=$(ls -l
```

معرفی از متغیر های محلی .

متغیر myvar1 مقدار صحیح، متغیر myvar2 مقدار اعشاری، متغیر myvar3 مقدار رشته ای و متغیر myvar4 خروجی دستور ls -l را در خود نگه می دارد. در مورد مقدار دهی متغیر ها نکات زیر مهم هستند:

- برای اختصاص مقادیر رشته ای به متغیر باید رشته میان ” یا “ قرار گیرند ولی زمانی مهم است که بین کاراکتر های رشته فاصله ای وجود داشته باشد مانند my name is Saeed ولی اگر فاصله ای نباشد مانند mynameisSaeed دیگر نیاز به ” و یا “ نیست.
- برای ذخیره خروجی یک دستور در یک متغیر از ` استفاده کنید.
- نوع متغیر های BASH یا هر پوسته دیگری بصورت پویا است، یعنی می توانید یک متغیر را در چندین خط از اسکریپت با مقادیر نوع متفاوت استفاده کنید.

تفاوت متغیر های محلی و تعریف شده توسط شما در حوزه شناسایی آنهاست به این معنی که متغیر های محلی در تمامی ترمینال ها و پنجره ها قابل استفاده و چاپ هستند ولی متغیر های تعریف شده توسط شما فقط در همان ترمینال یا پنجره قابل استفاده و چاپ هستند یعنی اگر پنجره یا tab جدیدی باز کنید دیگر متغیر اعتبار ندارد.

چاپ کردن متغیر ها

با استفاده از دستور های printenv و echo می توانید مقدار متغیر های محیطی را چاپ کنید ولی با دستور printenv نمی توانید مقدار یک متغیر تعریف شده را چاپ کنید و فقط با دستور echo می توانید مقدار آنها را چاپ کنید. فرقی که بین این دو دستور وجود دارد در این است که برای چاپ یک متغیر با echo باید علامت \$ ابتدای نام متغیر بیاید.

```
printenv variable_name
```

```
printnv HOME
```

```
printenv PATH
```

```
echo $variable_name
```

```
echo $HOME
```

```
echo $PATH
```

```
echo $myvar1
```

```
echo $myvar2
```

```
echo $myvar3
```

```
echo $myvar4
```

نکته ای که در رابطه با دستور `echo` وجود دارد در استفاده از مقدار یک متغیر و یک رشته برای چاپ است. دو دستور زیر شاید یک خروجی را بدهند ولی اینطور نیست بلکه اولین دستور عبارت `HOME$` را یک رشته در نظر نمی گیرد بلکه آنرا نام یک متغیر و مقدارش را چاپ می کند ولی دومین دستور عبارت `HOME$` را یک رشته در نظر می گیرد و در خروجی به همراه دیگر کارکتر ها چاپ می کند. پس برای چاپ مقدار یک متغیر به همراه دیگر کارکتر ها باید مانند خط اول زیر میان ” ” قرار گردد. برای درک بهتر دو دستور را اجرا کنید تا تفاوت را بفهمید.

```
“echo=”My Home Directory is $HOME
```

```
‘echo=’My Home Directory is $HOME
```

اختصاص یک متغیر به متغیر دیگر به فرمت زیر است. دقت کنید که نام متغیر سمت راست عبارت حتمن و حتمن باید با علامت `$` شروع شود در غیر اینصورت `BASH` نمی تواند تشخیص دهد که عبارت سمت راست نام یک متغیر است بلکه آنرا یک رشته در نظر می گیرد.

```
new_variable_name=$old_variable_name
```

مثال کلی

```
1  #!/bin/bash
2  NOW=$(date)
3  echo "$NOW"
4  SERVERNAME=`hostname`
5  echo "Running command @ $SERVERNAME...."
6  CWD=$(pwd)
7  cd /path/some/where/else
8  echo "Current dir $(pwd) and now going back to old dir .."
9  cd $CWD
```

متغیر های درونی پوسته

در تمامی سیستم عامل ها یک سری متغیر ها وجود دارند که اطلاعات خاصی را به ازای هر کاربر یا بصورت سراسری برای تمامی کاربران نگه داری می کنند. اینگونه متغیر ها را in-build variable گویند. در این پست اصلی ترین متغیر هایی درونی سیستم عامل لینوکس را معرفی می کنم. این متغیر ها د بیشتر سیستم عامل های یونیکسی مانند BSD ها (مثل FreeBSD) و سیستم عامل سولاریس و همچنین سیستم عامل مکینتاش نیز وجود دارند. می توانید در [این پست](#) چگونگی تعریف متغیر را بخوانید. این متغیر ها در Shell Scripting بسیار کاربردی هستند. بطور مثال برای اینکه بررسی کنیم که آیا کاربر Login شده root است یا نه از متغیر \$UID استفاده می کنیم. توجه کنید که به ازای هر کاربری که Login می کند مقدار این متغیر ها ممکن است متفاوت باشد بطور مثال مقدار این متغیر های برای کاربر root نسبت به دیگر کاربران متفاوت است. برای فهرست کردن تمامی متغیر های محیطی (in-build) از دستور env استفاده کنید.

```
env | less
```

- \$SHELL نام پوسته ای که در هنگام ایجاد یک کاربر و با استفاده از سوییچ s- از دستور useradd به کاربر دادیم را نشان می دهد. لیستی پوسته های نصب شده در سیستم عامل لینوکسی (یونیکسی) در فایل /etc/shell قرار دارد.

```
cat /etc/shells
```

اگر می خواهید بدانید که نام پوسته (Shell) پیش فرضتان چیست دستور را را اجرا کنید.

```
echo $SHELL
```

- \$HISTSIZE در دایرکتوری خانگی هر کاربر فایلی به نام bash_history وجود دارد که لیستی از دستور های اجرا شده آن کاربر را نگه می دارد. متغیر \$HISTSIZE تعداد دستور هایی که در این فایل نگه داری خواهد شد را تعیین می کند. مقدار پیش فرض آن ۱۰۰۰ است.
- \$HISTFILE محل ذخیره فایل را نشان می دهد. که بصورت پیش فرض در دایرکتوری خانگی هر کاربر قرار دارد.
- \$USER نام کاربریتان را ذخیره دارد.
- \$UID شناسه کاربر را نشان می دهد. شناسه کاربر root برابر عدد ۰ و شناسه کاربران سیستمی مانند mysql اعدادی از ۱ تا ۱۰۰ هستند. کاربران سیستمی امکان Login به سیستم را ندارند و تنها توسط سرویس هایشان برای کنترل و دسترسی و نوشتن در فایل هایشان استفاده می شوند. به این خاطر نمی توانند وارد (Login) به سیستم شوند، چون دارای nologin Shell هستند. کاربران معمولی نیز عددی مانند ۵۰۰ به بالا دارند. بطور مثال برای اینکه اجبار کنیم تا اسکریپتی توسط کاربر root انجام شود می توان از این متغیر استفاده کرد.
- \$GROUPS بدست آوردن اطلاعات گروه
- \$PWD حاوی دایرکتوری جاری است. دایرکتوری جاری دایرکتوری است که کاربر در حال حاضر در آن قرار دارد. دستور pwd از مقدار این متغیر استفاده می کند.
- \$HOSTNAME نام ماشین یا hostname را نشان می دهد.

- **HOMES\$**: نام دایرکتوری خانگی کاربر را نشان می دهد.
- **HOSTTYPE\$**: معماری ماشین یعنی ۳۲ بیتی یا ۶۴ بیتی بودن را نشان می دهد.
- **OSTYPE\$**: نوع سیستم عامل را نشان می دهد. بطور مثال اگر از لینوکس استفاده می کنید مقدارش Gnu Linux و یا اگر از سولاریس استفاده می کنید مقدارش Sun Solaris خواهد بود.
- **TERM\$**: نام Terminal را نشان می دهد.
- **TMP\$**: یا **TMPDIR\$** مسیر دایرکتوری Template را نشان می دهد
- **PATH\$**: مسیر دایرکتوری هایی که حاوی فایل های باینری برای کاربر را نشان می دهد. برای اطلاع بیشتر [این پست](#) را بخوانید.
- **PIPESTATUS\$**: عدد وضعیت اجرای یک Pipe را نشان می دهد Pipe. به معنی ارسال خروجی یک دستور به ورودی دستور دیگر است. بطور مثال دستور زیر ابتدا ls -l انجام می شود و خروجی آن به دستور wc -l که تعداد سطر های خروجی دستور ls -l را می شمارد و نشان می دهد. چون اجرای دستور زیر درست است، پس وضعیت خروجی آن باید عدد ۰ باشد.

```
ls -l | wc -l
echo $PIPESTATUS
0
```

اما اگر بجای دستور بالا از دستور زیر استفاده کنیم، چون از دستوری اشتباه استفاده کردیم (یک اشتباه در استفاده نادرست از سوئیچ ها در دستور wc) ، پس عددی به جز صفر (۰) نشان داده می شود.

```
ls -l | wc -l
echo $PIPESTATUS
141
```

- **PPID\$**: شناسه والد فرایند را نشان می دهد.
- متغیر های PS1 و PS2 و PS3 و PS4 نیز برای سفارشی کردن ترمینال به کار می روند. (توجه کنید که اول نامشان \$ است)
- پارامتر های زیر در اسکریپت نویسی بسیار مفید هستند.
- **RANDOM\$**: از این متغیر برای ایجاد یک عدد تصادفی استفاده می کنند. بطور مثال می توان در اسکریپتی که برای ایجاد پسورد های پیچیده استفاده نمود.
- **SECONDS\$**: تعداد ثانیه هایی که یک اسکریپت در حال اجرا شدن است.

- \$0 نام اسکریپت را چاپ می کند. برای مثال در کد های زیر که در فایل fcleanup.sh ذخیره شده اند خط \$0 echo با عث نمایش نام فایل اسکریپت یعنی fcleanup.sh می شود.

```

saeed@localhost:/home/saeed/Desktop
File Edit View Search Terminal Help
GNU nano 2.0.9 File: esecurit.sh Modified

#!/bin/bash
#very sample script for clean up a file
#www.esecurity.ir
echo $0
read -p "pls Enter a filename or path to filename : " FILENAME
cat /dev/null >$FILENAME

echo "File ${FILENAME} clean up"

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

```

- \$# تعداد آرگومان های ورودی یک شل را نشان می دهد.
- \$* تمامی آرگومان های ورودی را نشان می دهد.
- @\$ تمامی آرگومان های ورودی را با یک فاصله نشان می دهد.
- \$NUMBER در اصل این آرگومان ها بصورت \$1 و \$2 و \$3 ... هستند معرف اولین آرگومان ورودی و دومین آرگومان ورودی و سومین آرگومان ورودی و ... هستند. بطور مثال برای چاپ دومین و چهارمین آرگومان ورودی از دستور زیر استفاده کنید.

“echo “Dovomin argument hast: \$2 va Chaharomin argument hast: \$4

- \$? شماره وضعیت خروجی (Exit Status) یک دستور را نشان می دهد.
- \$\$ شناسه فرایند فعلی را نشان می دهد.

- \$: شماره فرایند Background را نشان می دهد.
- \$: آخرین آرگومان شل را نشان می دهد. (اولین آرگومان شل \$۱ بود)

ادامه دارد

منابع و مآخذ

فارسی

<http://wiki.linuxreview.ir/Shell-scripting>

<http://falearn.ir>

<http://bashlinux.persianguig.com>

<http://linux.itpro.ir/articles/9367/>

<http://ashiyane.org/forums/showthread.php?17402>

<http://opensource-mag.ir/index.php/maghalat-manba-baz/maghalat-amniat-hack/18-shell-scripting.html>

<http://barnamenevis.org/showthread.php?84688>

انگلیسی

<http://conqueringthecommandline.com/>

<http://www.tldp.org/LDP/abs/html/>

<http://tille.garrels.be/training/bash/>

<http://learnvimscriptthehardway.stevelosh.com/>

