part 1

Shell Scripting

www.esecurity.ir

collector : saeed hosseini

shell scripting

In

Linux

Collector

Saeed hosseyni

www.esecurity.ir

part 1

فصل یک

فصل دوم

فصل سوم

فصل چهارم

فصل پنجم

فصل ششم

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 کاملا 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 شمائید که با کمک خود می‌توانید این راهنما را به‌روز و زنده نگه دارید؛

فصل اول - تاریخچه و مفاهیم

تاریخچه

گنو

در دهۀ ۱۹۷۰ و اوایل دهۀ ۱۹۸۰، برنامه‌نویسان کامپیوتر تمامی آنچه را می‌نوشتند با دیگران به اشتراک می‌گذاشتند. هر شخصی متن برنامۀ 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 منحصرا 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 دهۀ 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 گذاریِ نرم‌افزار جرم محسوب می‌شد .

اولین مرحله برای داشتن یک کامپیوترِ آزاد، وجود یک سیستم‌عامل آزاد بود .بنابراین ریچارد استالمن زمانی که دانشجوی دکتری بود از شغل و تحصیل خود در MITاستعفا داد و در اوایل سال ۱۹۸۳پروژۀ گنو را آغاز و اطلاعیه اولیه آن‌را 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 اولیۀ مورد نیاز برای طراحی و ساخت یک سیستم‌عامل فراهم شد. ابزاری مبهوت کننده برای برنامه نویسان مستقل. وی با جادوگری افسانه ای 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 کارآمد ترین و قویترین کامپایلرهایی است که تا کنون ایجاد شده اند .

استالمن در سال ۱۹۸۵بنیاد نرم‌افزارهای آزاد (free software foundation) را پایه گذاری کرد، مفهوم کپی‌لفت را تعریف و اجازه‌نامۀ جامع و عمومی گنو (gnu public license)برای حمایتِ نرم‌افزارهای آزاد و تضمین آزادی کابران 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.

لینوکس

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 فقدانی 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 PCانتخاب دیگری نداشتند. کامپیوترهای اپل مکینتاش بهتر بودند. ولی قیمتهای 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 آزمایشگاههای 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 MINIXارائه شد .

این سیستم عامل، که از ابتدا توسط ا ندرو اس. تانبام (Andrew S. Tanenbaum) پروفسور 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، MINIXخیلی خوب نبود. ولی مزیت اصلی آن، در دسترس بودن کد منبع آن بود .هرکس که کتاب سیستم عامل تانبام 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 براورده نمایدMINIX .خوب بود ولی فقط یک سیستم عامل مخصوص دانش آموزان بود و بیشتر به عنوان یک ابزار آموزشی بود تا ابزاری قدرتمند برای بکار گیری 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 HURDادامه 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 MINIXمن 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 کرده‌ام و هم‌اکنون این سیستم‌عامل آماده‌است و کار می‌کند. دوست دارم از دیدگاه دیگران در مورد سیستم‌عاملم با خبر شوم. چه آنان که مینیکس را دوست دارند و چه آنان که دوست ندارند. چرا که سیستم‌عامل من تا حدی شبیه به مینیکس است .در حال حاضر bash ۱،۰۸ و gcc ۱،۴۰ را بر روی آن دارم و چیزهای دیگری که به نظر می‌رسد همه درست کار می‌کنند. این بدان معناست که طی چند ماه آینده چیز به‌دردبخوری فراهم خواهم کرد و دوست دارم بدانم مردم بیشتر چه امکاناتی لازم دارند. به هر پیشنهاد و نظری خوش‌آمد می‌گویم اما قول نمی‌دهم که آن را انجام دهم !پ.ن: بله این نرم‌افزار آزاد است. البته قابل انتقال بر روی انواع دیگر رایانه نیست) چرا که دستوراتAT ۳۸۶ را به کار می‌برد (و ممکن است غیر از دیسک سخت AT چیز دیگری را پشتیبانی نکند. 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 0.01در اواسط سپتامبر ۱۹۹۱ منتشر شد و روی اینترنت قرار گرفت. شور و اشتیاقی فراوان حول مخلوق توروالدز شکل گرفت. کدها دانلود شده، آزمایش شدند و پس از بهینه سازی به توروالدز بازگردانده شدند .

لینوکس نسخه 0.02در 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؟ ایا شما فاقد یک پروزه 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 مستقیما 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 گرافیکیVGA ، 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 FTPدر 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 پروژهGNU ، لینوکس آماده یک نمایش واقعی بود. لینوکس تحت مجوز GPLقرار داده شد. با این مجوز همه می‌توانستند 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 کنارPC ، لینوکس به روی اکثر پلاتفورمها منتقل شده است. لینوکس تغییر داده شد تا کامپیوتر دستی شرکت ۳ Comیعنی 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 میلیاردر 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

Enviroment 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

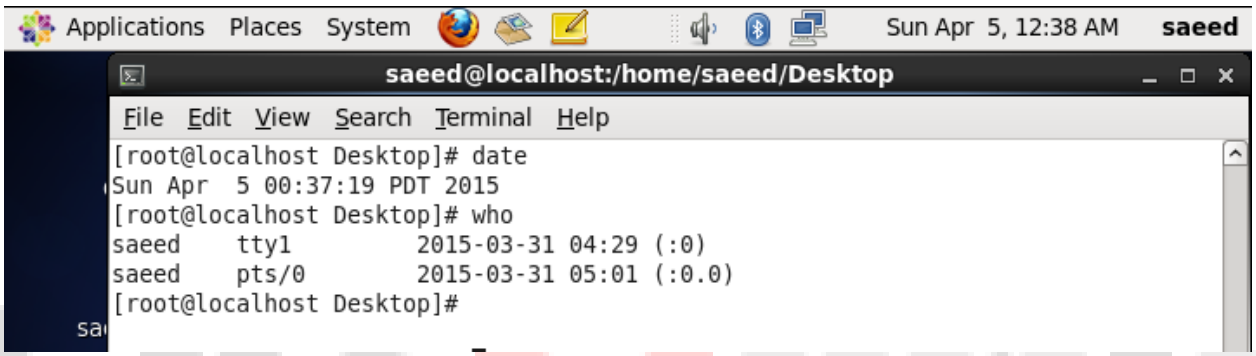
Errors and Signals and Traps- مدیریت خطا ها و علائم در بش اسکریپت

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 میکنه. شل از شما مقداری رو میگیره و محاسبات رو بر اساس اون مقدار برای شما انجام میده. وقتی اجرای برنامه تمام میشه خروجی برای شما نمایش داده میشه.

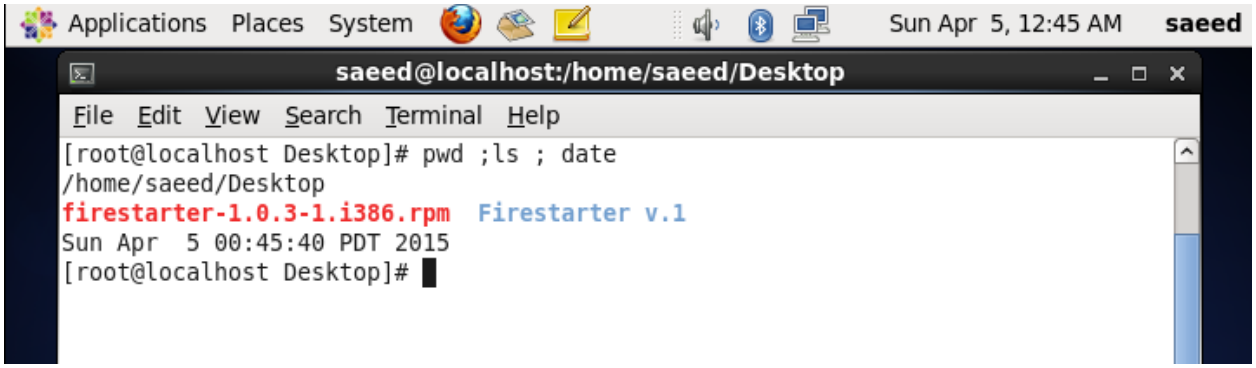
شل محیطی ست که میتونه دستورات برنامه ها و اسکریپت های شما رو اجرا کنه. همونطور که سیستم عامل های مختلفی هست شل های مختلفی هم وجود داره.



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:



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

برا ی فهمیدن مفهوم شل می توان به شکل زیر توجه کرد

انواع شل در لینوکس در زیر آمده است :

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

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، bashاز خانوادهی bourne بررسی شدهاست.

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 Aliasها استفاده کنید.

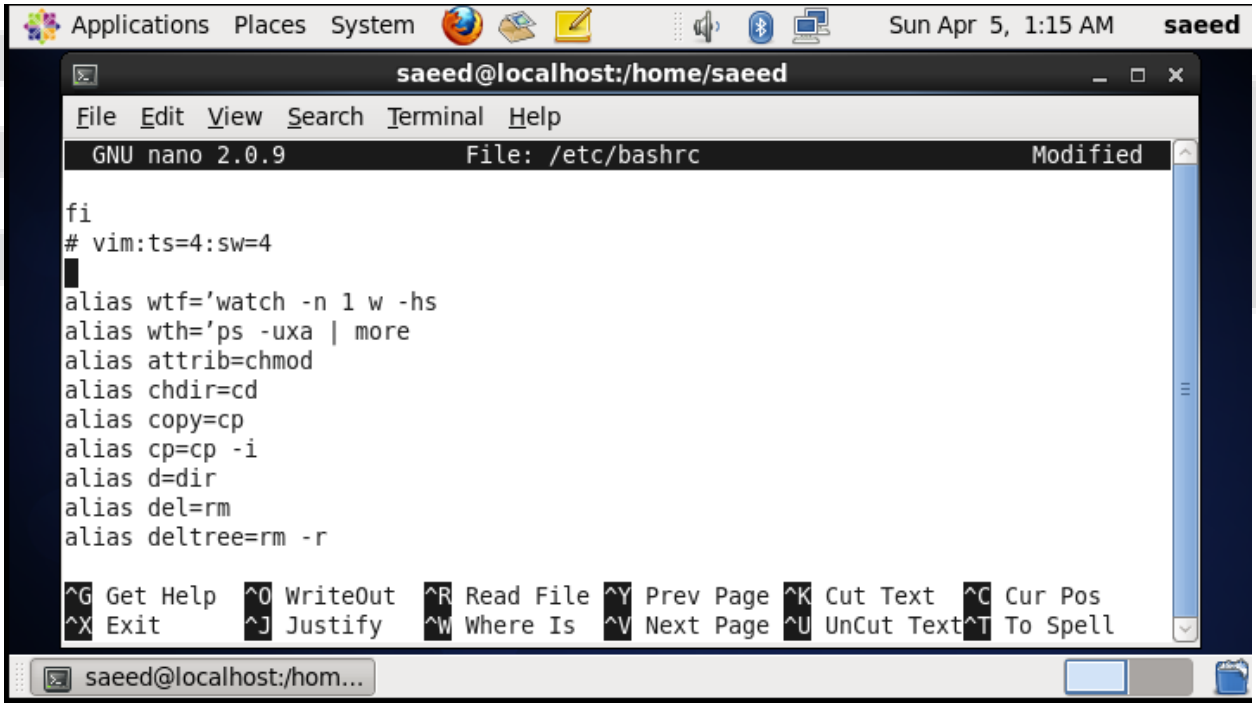
از دستور alias برای تعریف Alias ها استفاده می شود. فرمت کلی Alias ها بصورت زیر هستند.

```
alias alias_name=cmd
```

بجای alias_name نام دلخواه معادل دستور زا 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 bashrcبا هربار 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 دایزکتوری /etc که اثرش برتمام کاربرن و کل سیستم است.پس اگر می خواهید aliasتعریف 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 کاربرن 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:



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 ... mOvEدر تفاوت است.

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

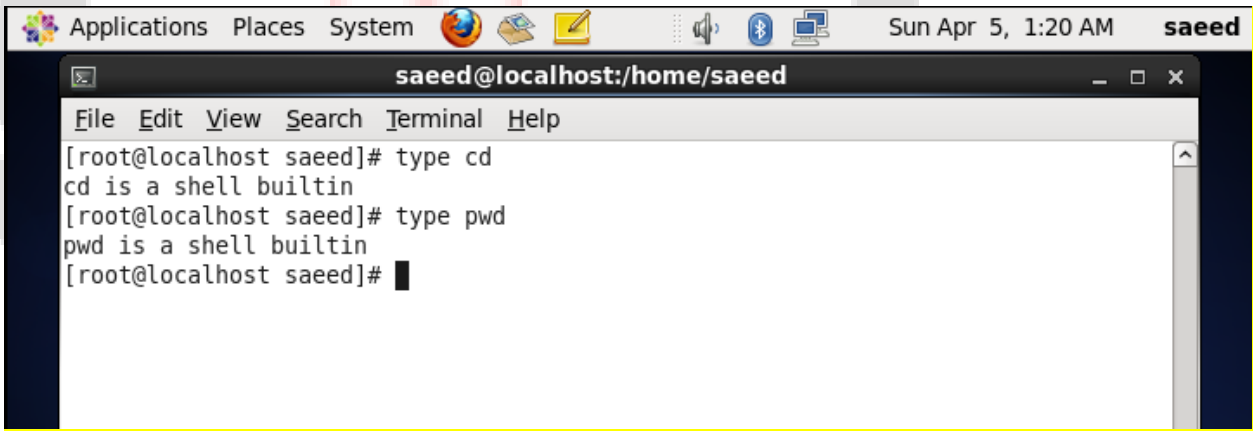
از بین بردن نام های مستعار:

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

```
unalias alias_name
```

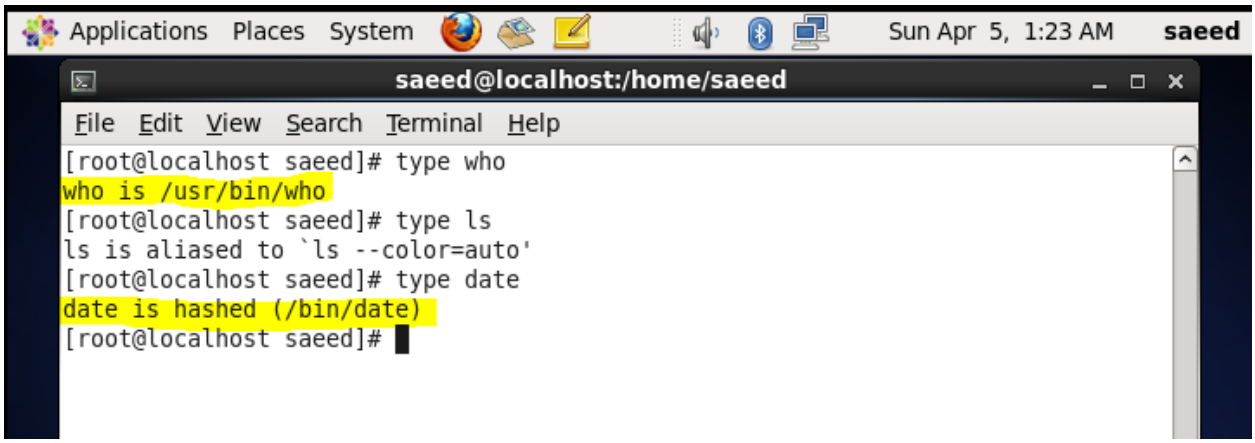
دستورات درونی(builtin)

دستوراتی که توسط خود شل اجرا می‌شوند و به برنامهٔ دیگری وابسته نیست



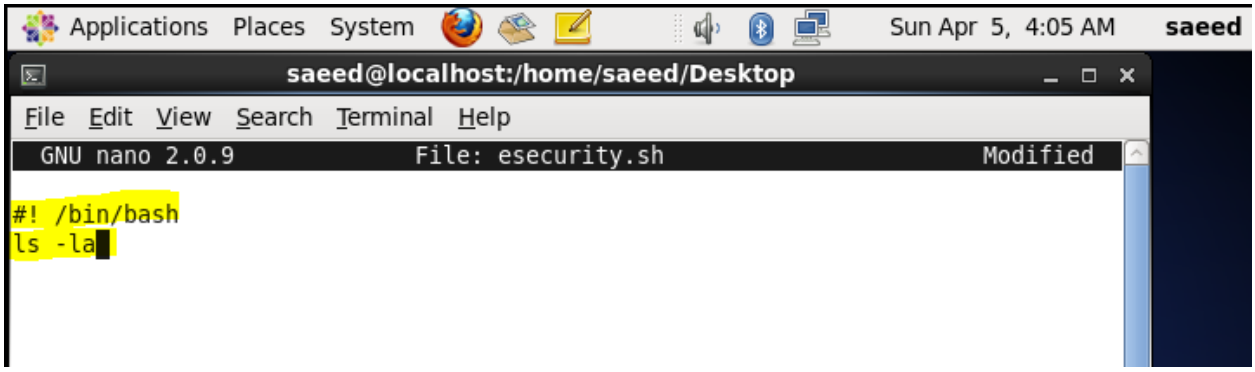
دستورات خارجی

که اغلب دستورات را تشکیل می‌دهند، متناظر با یک فایل اجرایی برای شل (باینری یا اسکریپت) هستند که در پوشه‌های خاصی قرار دارند که بعدا معرفی خواهد شد.



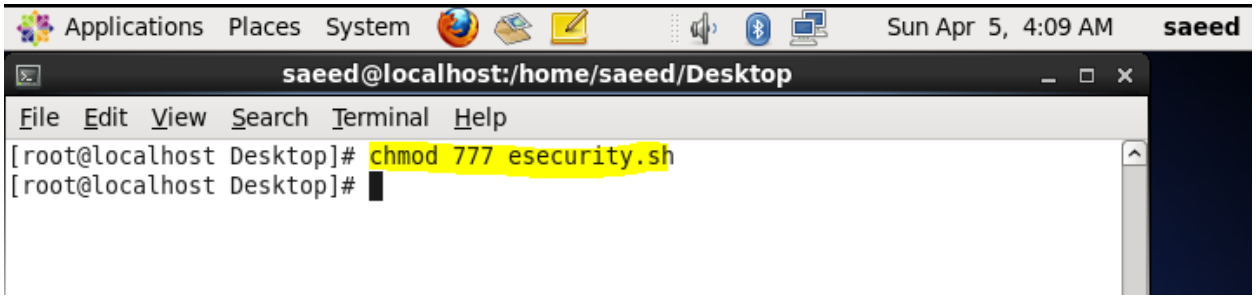
نوشتن و اجرای یک اسکریپت

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

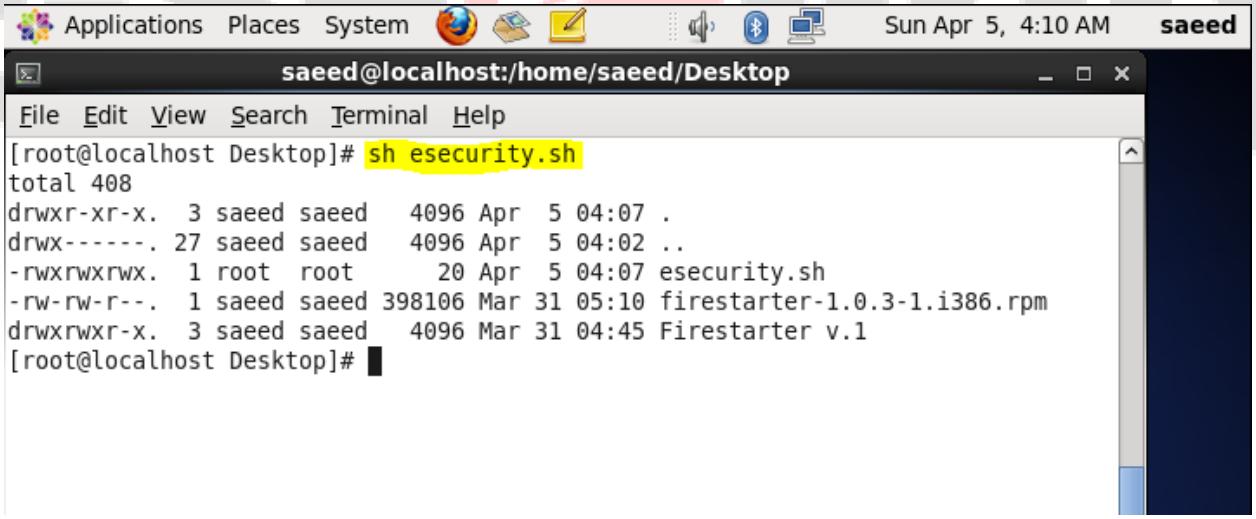


خط اول این اسکریپت که معروف به خط جادویی نیز است موجب ایجاد یک نمونه‌ی جدید از شل برای اجرای دستورات می‌شود. در صورتی که خط اول در ابتدای اسکریپت نوشته نشود، از شل جاری برای اجرای اسکریپت استفاده می‌شود .به عنوان مثال در صورتی که این خط نوشته نشود اسکریپت‌هایی که با استفاده از tcsh یا cshنوشته شده‌اند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

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



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



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 .

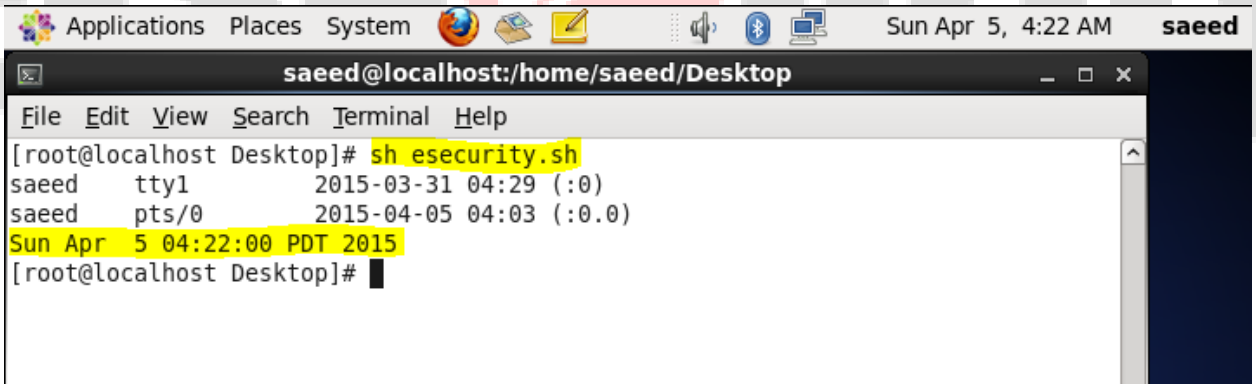
توضیحات

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 :



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 .

فصل سوم - فایل‌ها

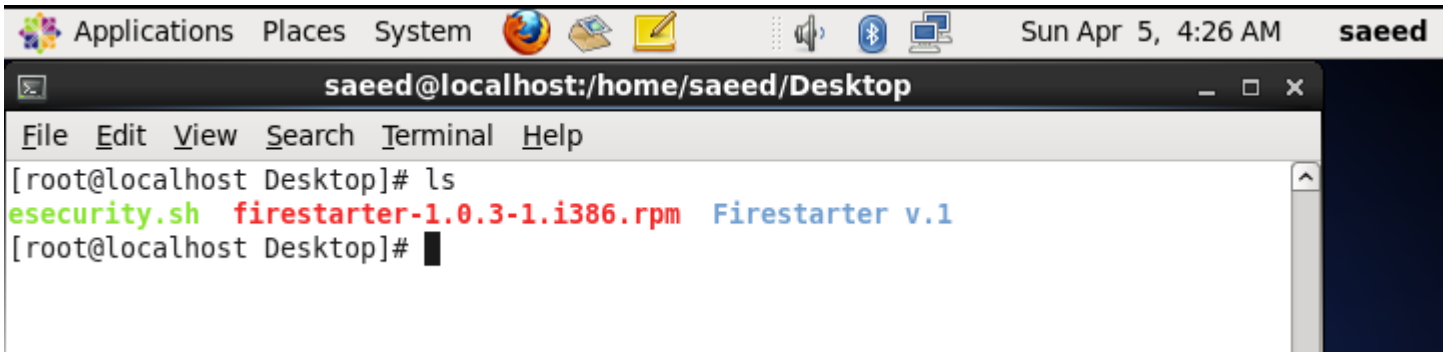
در سیستم‌های یونیکسی همه چیز به صورت فایل است. در این فصل و فصل بعد به برخی ار ویژگی‌های انواع فایل در سیستم‌های یونیکسی میپردازیم

مشاهدهٔ 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:

مشاهدهٔ 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 :



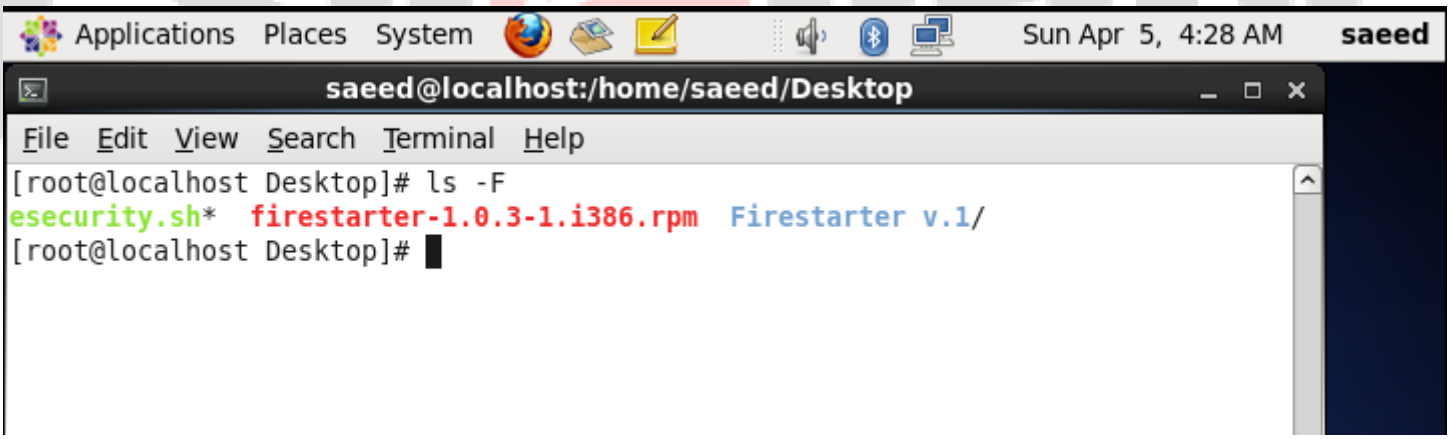
که این دستور لیست فایل‌های موجود در دایرکتوری جاری را نمایش می‌دهد .

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

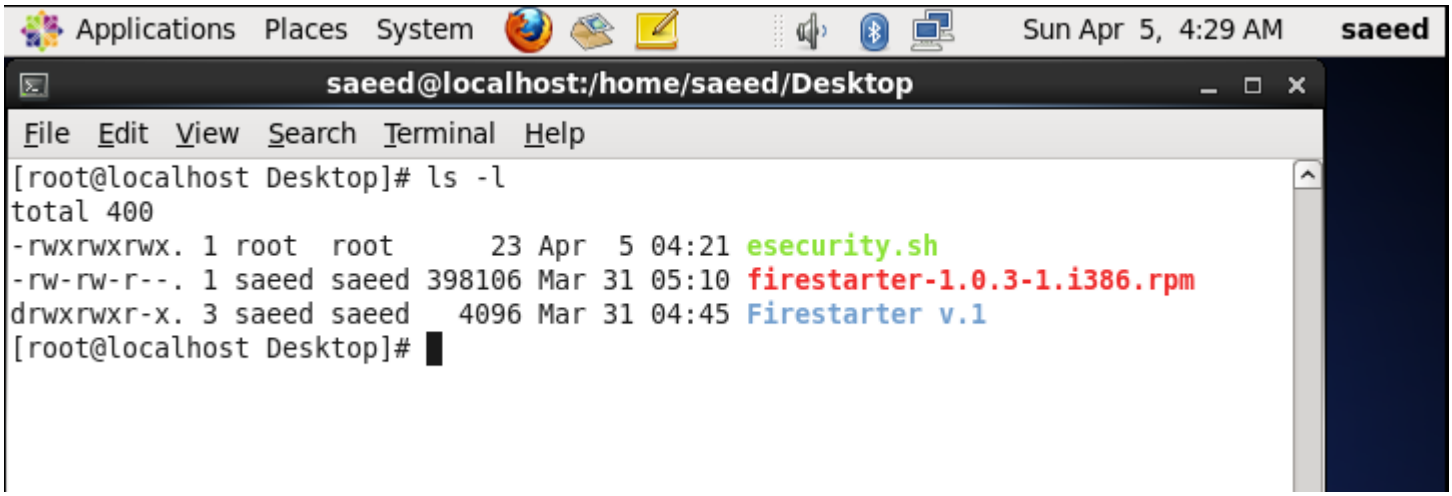
آپشن F

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 .



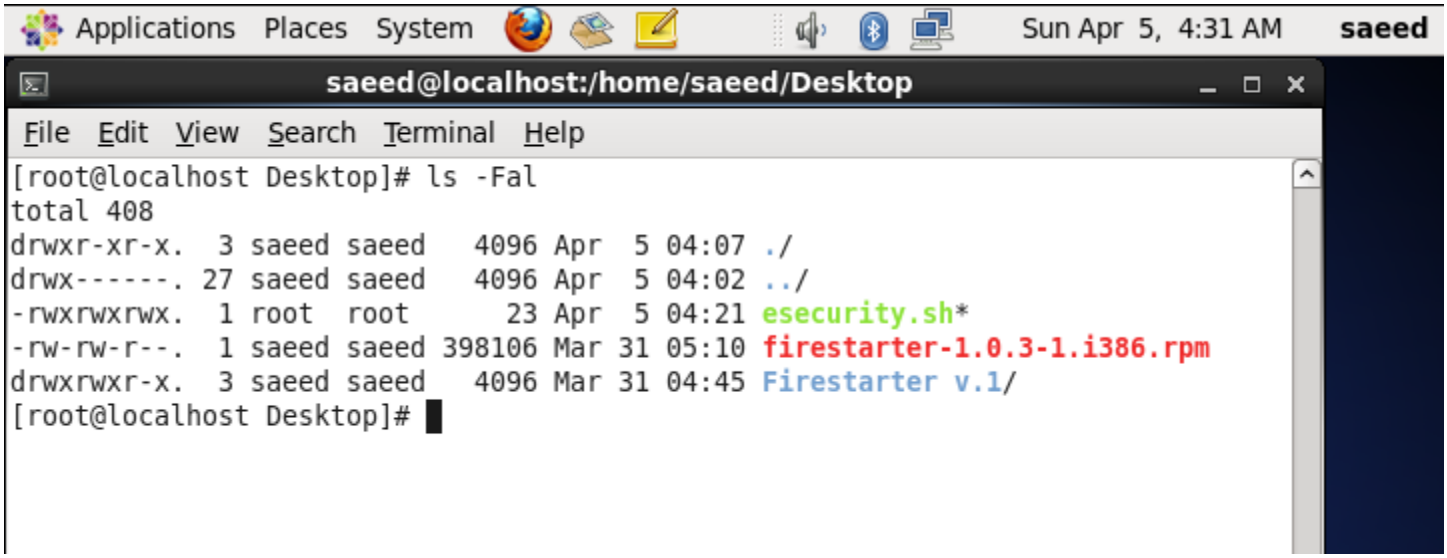
آپشن ۱

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



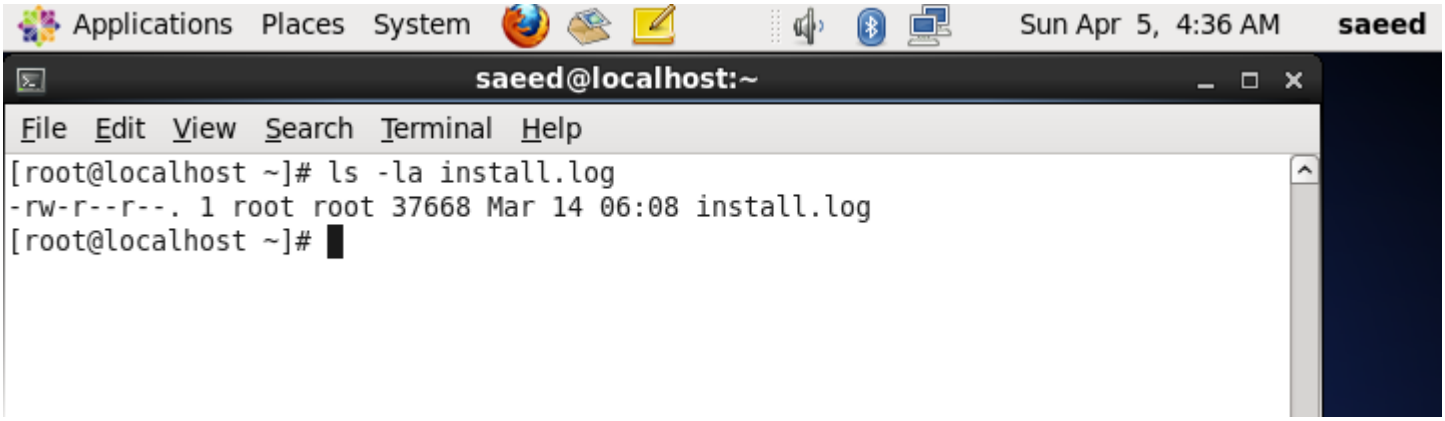
نکته: این آپشن معمولاً برای استفاده در حلقه‌ها در اسکریپت، یا برای فرستادن به دستور دیگر، مفید است. چون نام فایل/دایرکتوری ممکن است شامل فاصله باشد ولی احتمالاً شامل کاراکتر «سر خط» نیست (و اگر هم باشد، lsبجای آن علامت سوال نشان می‌دهد .)

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



بررسی نوع فایل

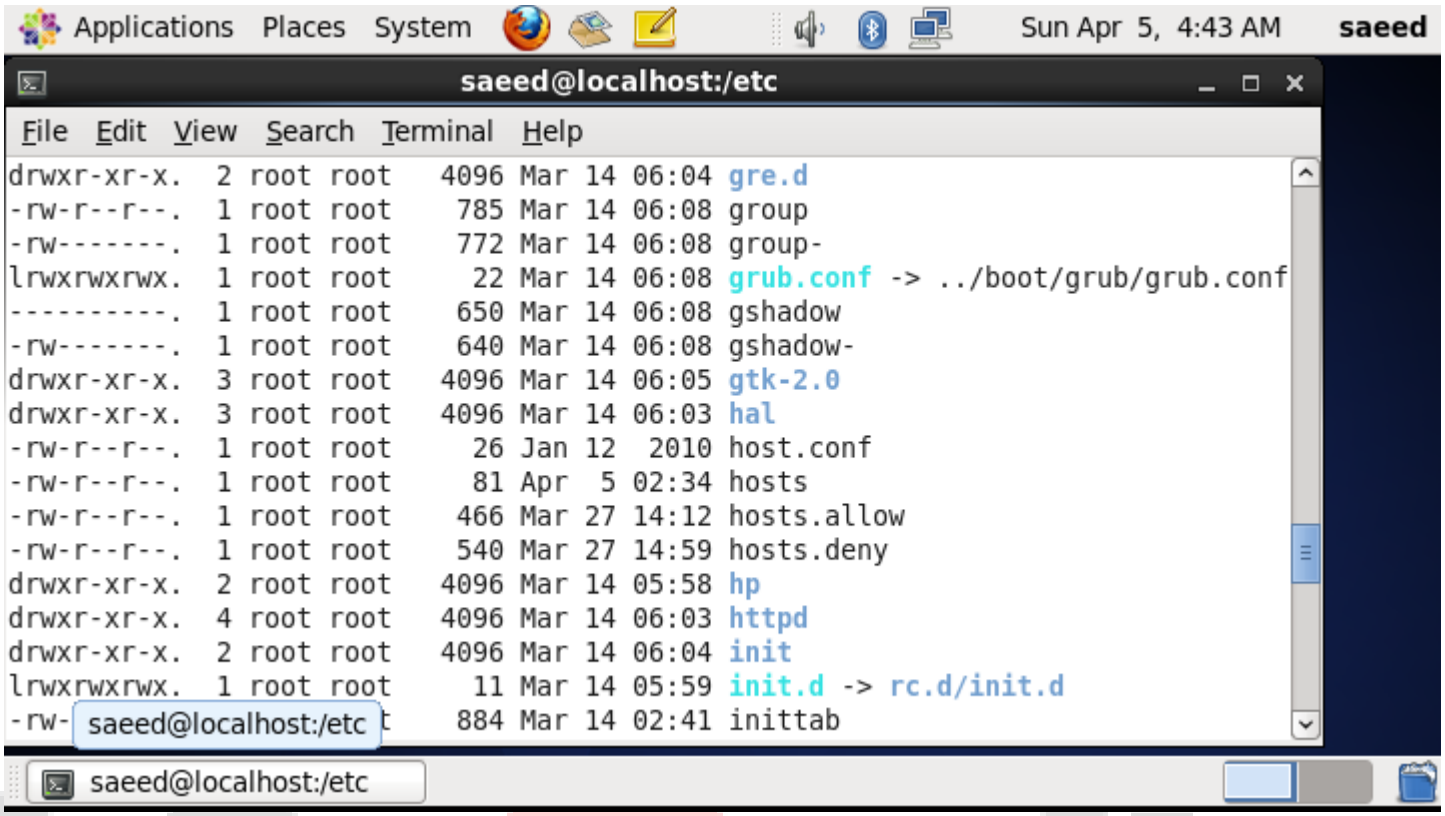
برای مشخص شدن نوع یک فایل از دستور ls به همراه آپشن l می‌توان استفاده کرد .



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 :



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)

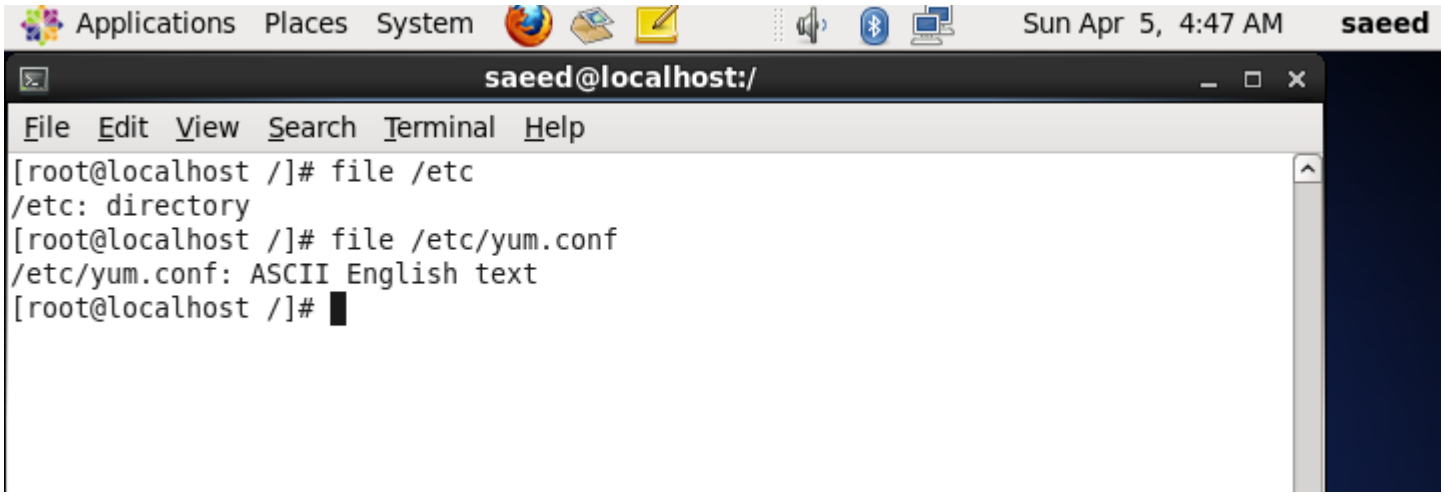
متداول ترین نوع فایلی که با آن سروکار دارید، فایل‌های معمولی (regular file)هستند. در این نوع از فایل تقریبا هر نوع داده‌ای از قبیل یک متن، یک برنامه کاربردی، فیلم، فایل‌های باینری یا تصویر را میتوان ذخیره کرد .
یونیکس توانایی درک داده‌های موجود در یک regular file را ندارد. یک فایل معمولی میتواند به هر شکلی یک داده خام را ذخیره کند. زیرا یونیکس عمل تفسیر بر روی داده‌های یک فایل را انجام نمیدهد .

اشاره: معمولا 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 :



.

.

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 مشخص کننده‌ي عملیاتی است که مالک آن میتواند انجام دهد. مجوز گروه مشخص کننده‌ی عملیات قابل اجرا در گروهی که کاربر در آن قرار دارد و مجوز دیگران مشخص کننده‌ی 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 .

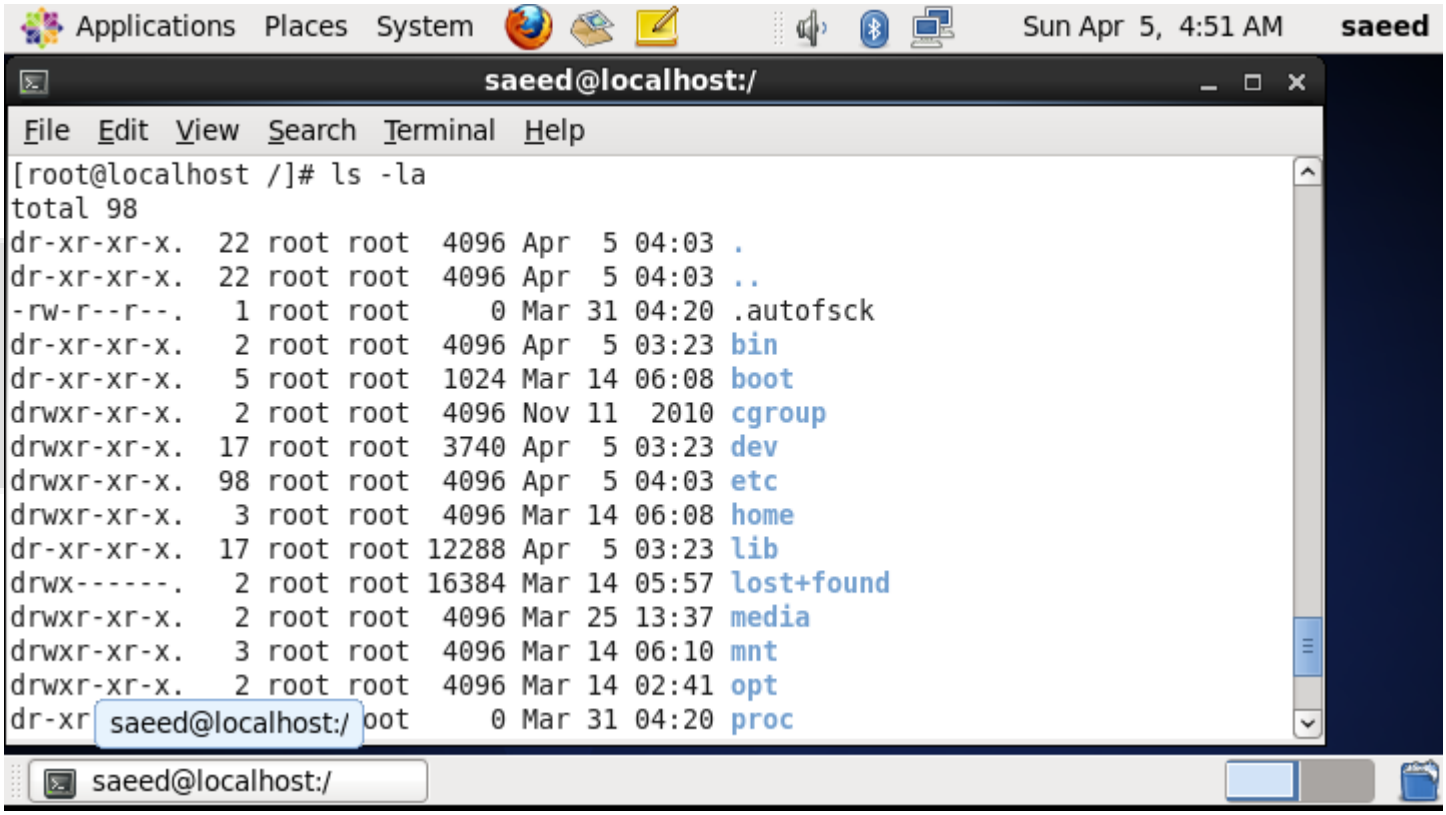
مشاهده‌ی مجوزها

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. rخواندن فایل
۲. wنوشتن
۳. xاجرا کردن

مثال :



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 تاثیری 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 .

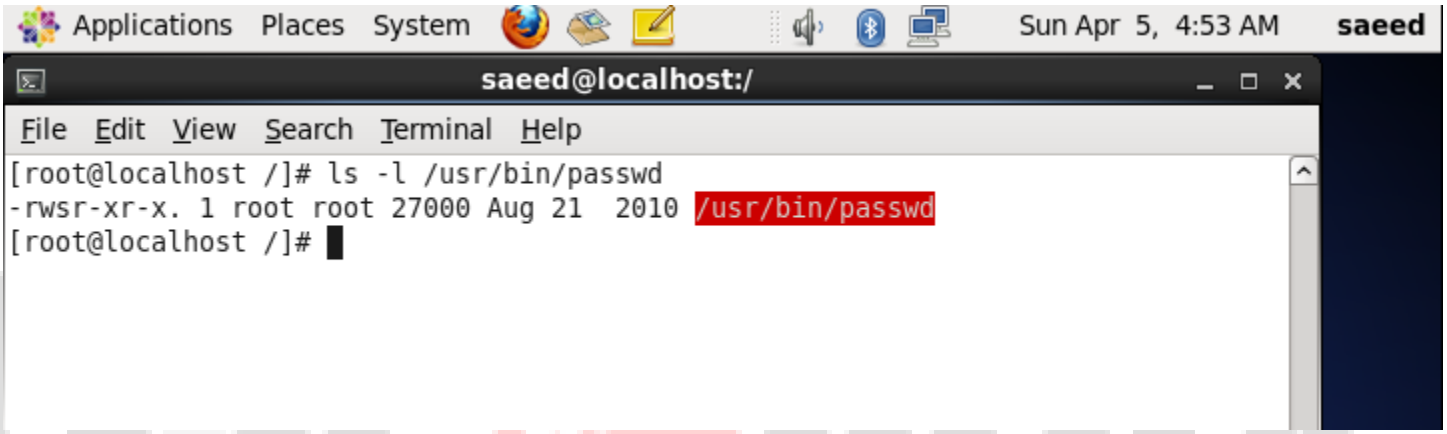
نکته: در صورتی که یک دایرکتوری تنها مجوز اجرا کردن داشته باشد، کاربر قادر مشاهده یا تغییر فایل‌های ذخیره شده در آن دایرکتوری نخواهد و تنها میتواند فایل‌های موجود در دایرکتوری را اجرا کند .

مجوزهای SUID وSGID

مکانیسمی برای دادن مجوزهای اضافی به برخی برنامه ها (SUID) set user idیا (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 SUIDباشد، کاربرانی که مجوز اجرای آن فایل را داشته باشند همانند مالک فایل در نظر گرفته می شوند. 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 :



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

مجوز sticky یاsticky bit

این مجوز اصولا به دایرکتوری‌ها داده می شود. فایل‌های درون دایرکتوری‌هایی که دارای این مجوز هستند تنها از طریق مالک آن فایل می توانند 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 میتوانید یک عبارت سمبلیک یا یک عدد در مبنای هشت باشد .

روش سمبلیک

شکل کلی یک عبارت سمبلیک به صورت زیر است :

```
(who)(action)(permissions)
```

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

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 .

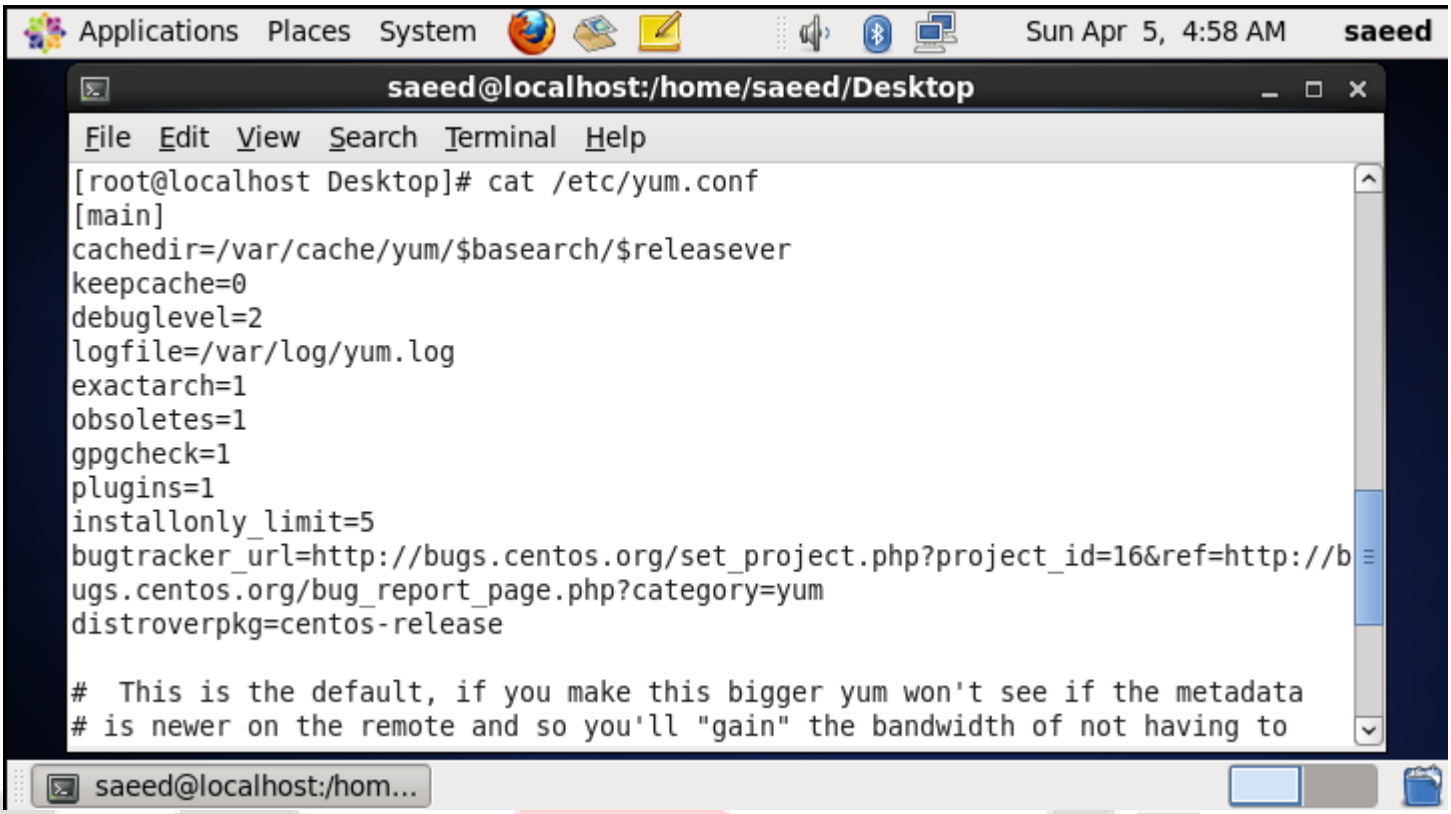
مشاهده‌ی محتوای فایل‌ها

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 .

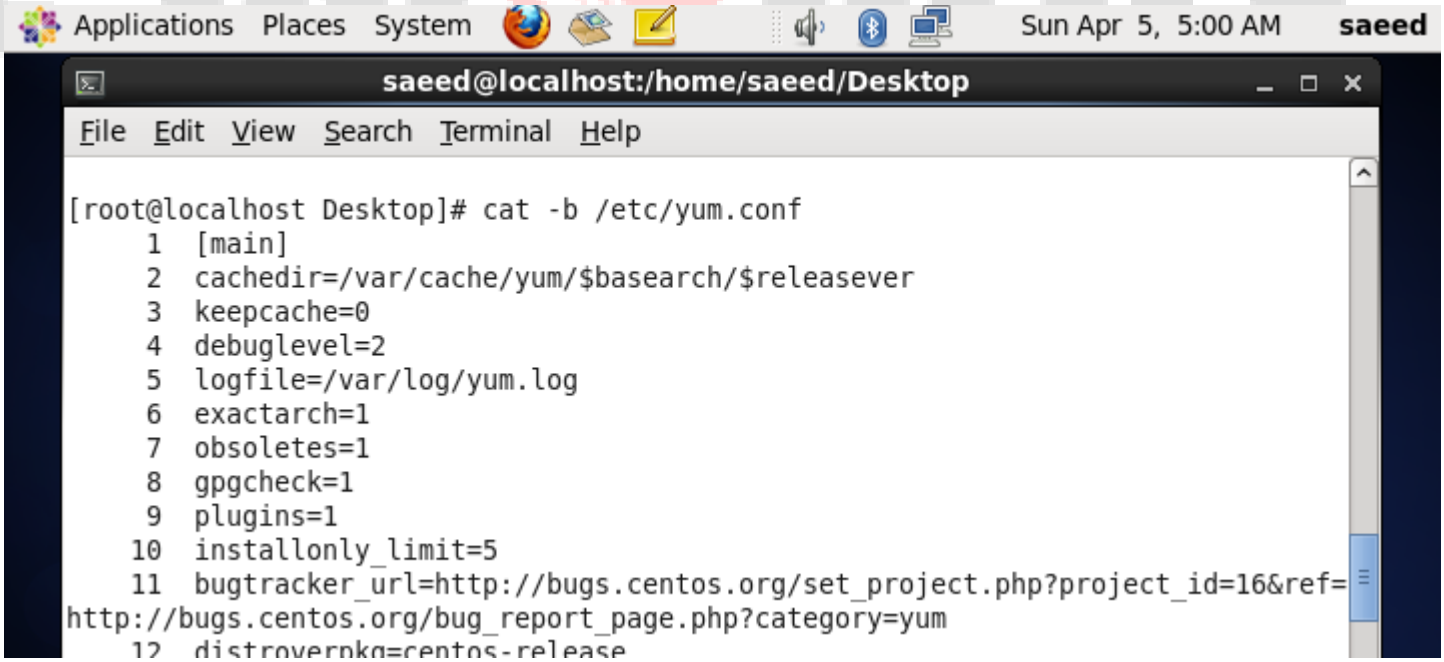
شکل کلی دستور cat به صورت زیر است .

```
cat [option] filenames
```

به طور مثال :



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 :

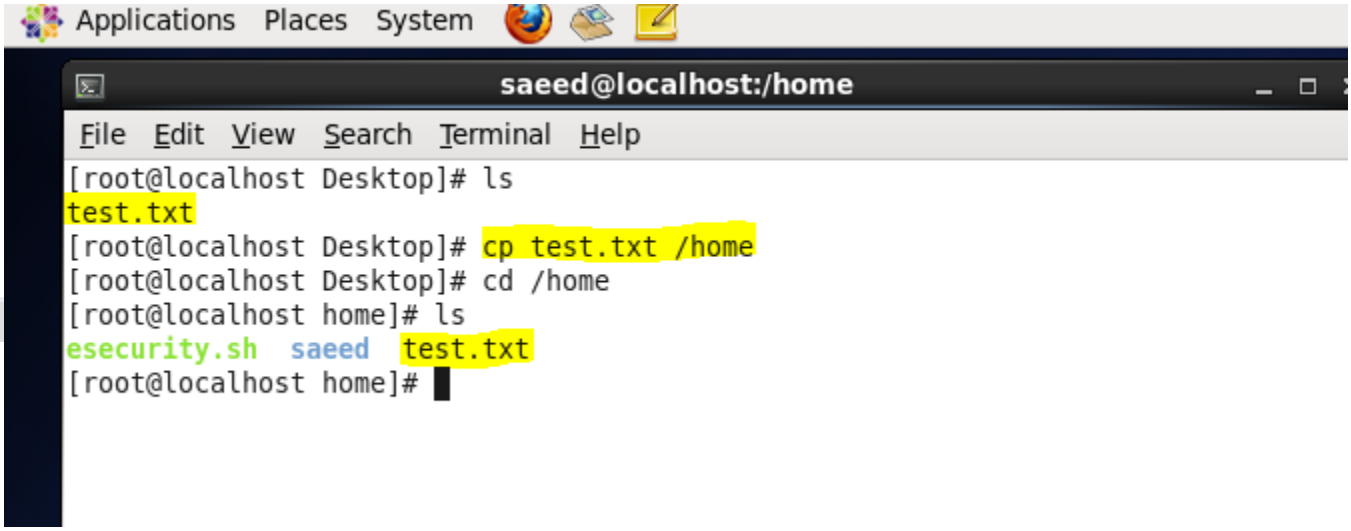


دستکاری فایل‌ها

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

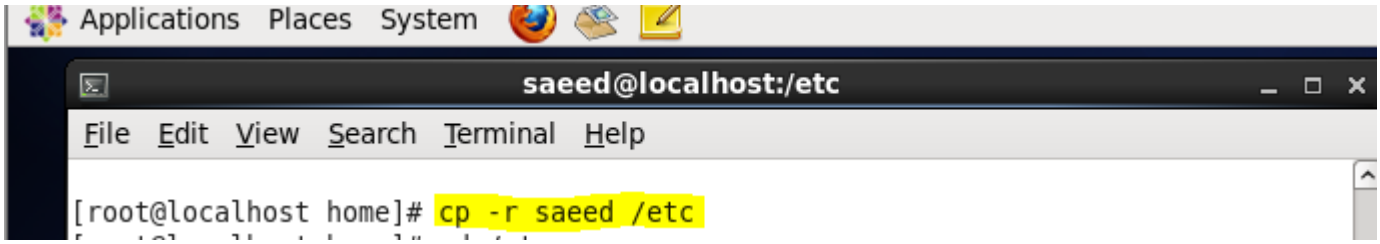
کپی کردن

شکل کلی دستور کپی(cp)

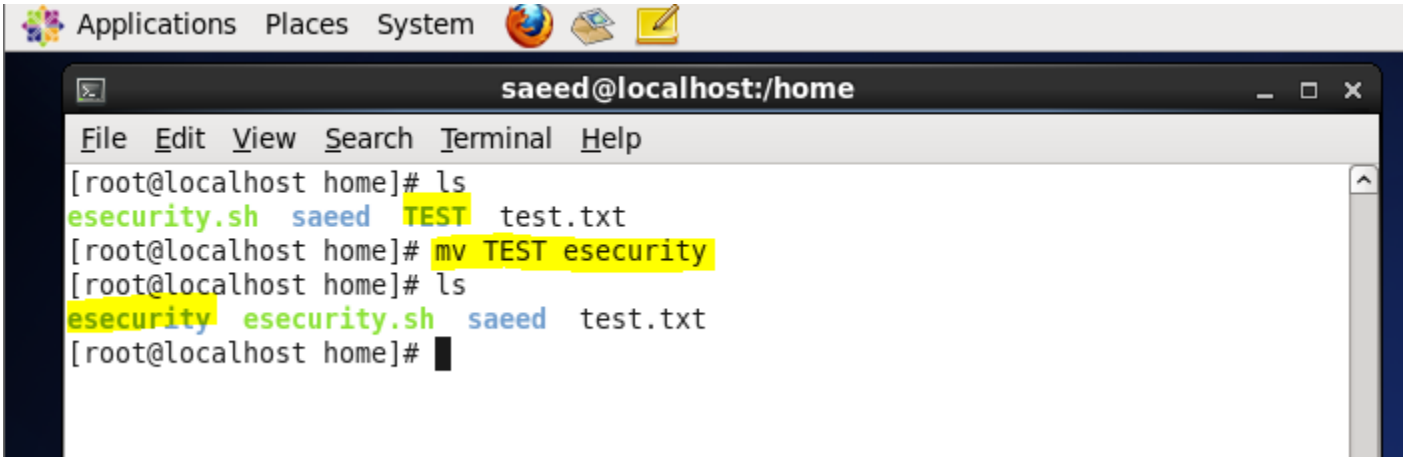


کپی کردن یک دایرکتوری

از آپشن r برای این مقصود استفاده می‌شود



تغییر نام و جابجا کردن(mv)



فصل چهارم - دایرکتوری‌ها

مقدمات

سیستم‌های مبتنی بر یونیکس برای مدیریت دایرکتوری‌ها و فایل‌ها از ساختارهای سلسه مراتبی که به صورت یک درخت وارون پیاده سازی میشود، استفاده میکنند که ریشه‌ی (root)این درخت با کاراکتر اسلش (/) مشخص میشود. و تمام دایرکتوری ها زیر شاخه‌های ریشه هستند.شما میتوانید از هر زیردایرکتوری 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 هارددیسک (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 .

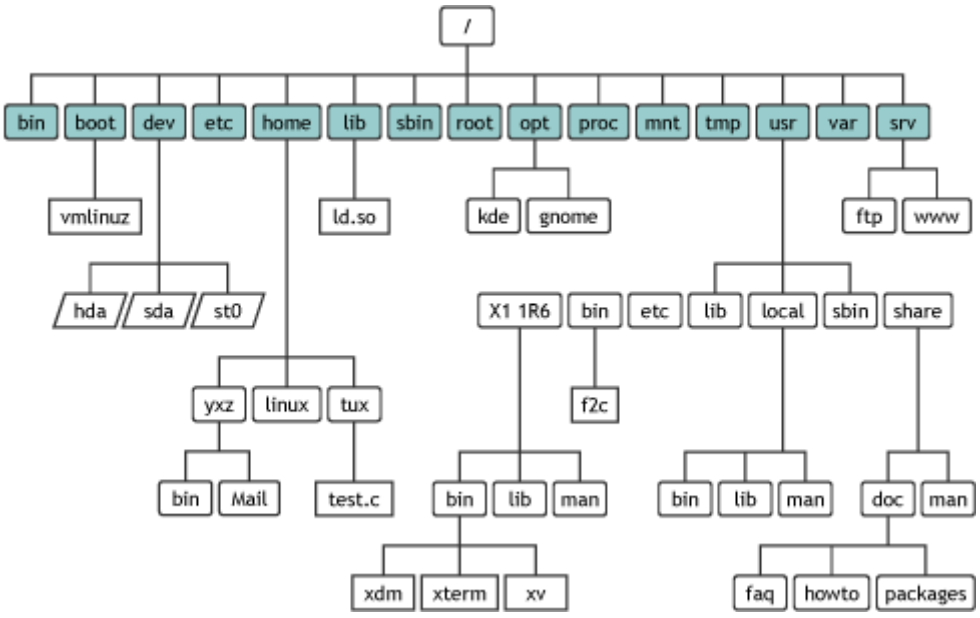
نکته: عمق درخت شامل دایرکتوری‌ها محدود و حداکثر ۱۰۲۴ است.

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 فزندان ریشه هستند و به همین خاطر است که در پیمایش سیستم فایل در سیستم عامل های شبه 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 Directoryیا 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 دستورfree ، touchیاpwd

دایرکتوری/sbin

- همانند دایرکتوری بالا این دایرکتوری تیز 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 هایfdisk, 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.

داریرکتوری/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.dقرار 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 vmlinuz 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 پیشفرض دایرکتوری /tmp است.
- فایل های زیر این دایرکتوری پس از rebootشدن سیستم به طور خودکار حذف می شوند.

دایرکتوری/var

- دایرکتوری برای فایل های متغیر یاvariable files
- در زیر این دایرکتوری فایل هایی مانند logها که به سرعت رشد می کنند قرار می گیرند.

دارکتوری/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 شدنCD/DVD ROM

دایرکتوری/usr

- دستور های پایه ای لینوکس مانند passwd در زیر دایرکتور های این دایرکتوری قرار دارند. دو زیر دایرکاوری /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.

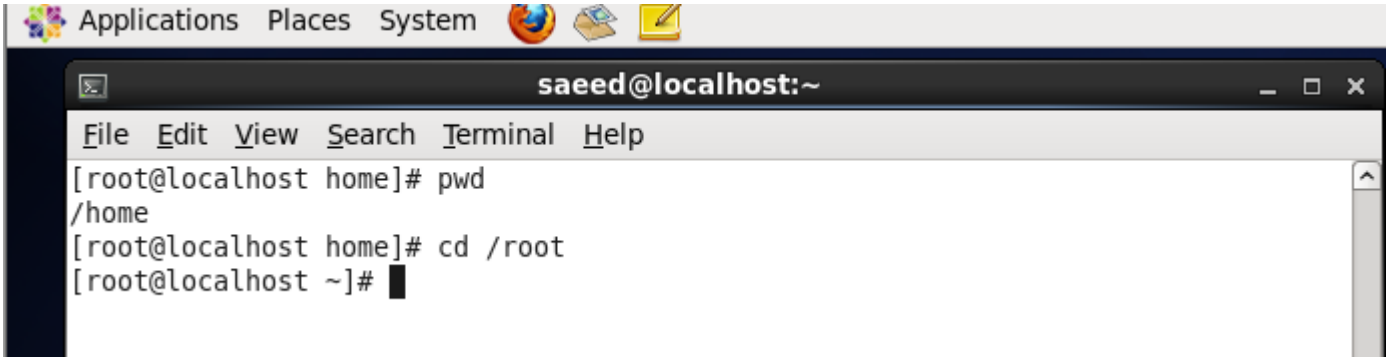
فرض کنید لینوکس را بر روی یک هارد می خواهیم نصب کنیم. در زمان نصب می توانیم دایرکتوری های زیر را به طور پیشفرض و در زمان نصب جدا از هم و بر روی پارتیشن های مجزایی نصب کنیم. به طور معمول لینوکس برای نصب شدن و بدون پارتیشن بندی سفارشی شده از سوی شما سه دایرکتوری / و دایرکتوری /home و swap را بر روی سه 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
- داریرکتوری/tmp
- دایرکتوری/usr
- دایرکتوری/var
- دایرکتوری/opt
- دایرکتوری/usr/loacl

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 بازهم همه ی این پارتیشن ها و دایرکتوری ها در پیملیش سر آغاز آنها دایرکتوری / یا دایرکتوری root است.

جابجایی بین دایرکتوری‌ها

ساده ترین 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.



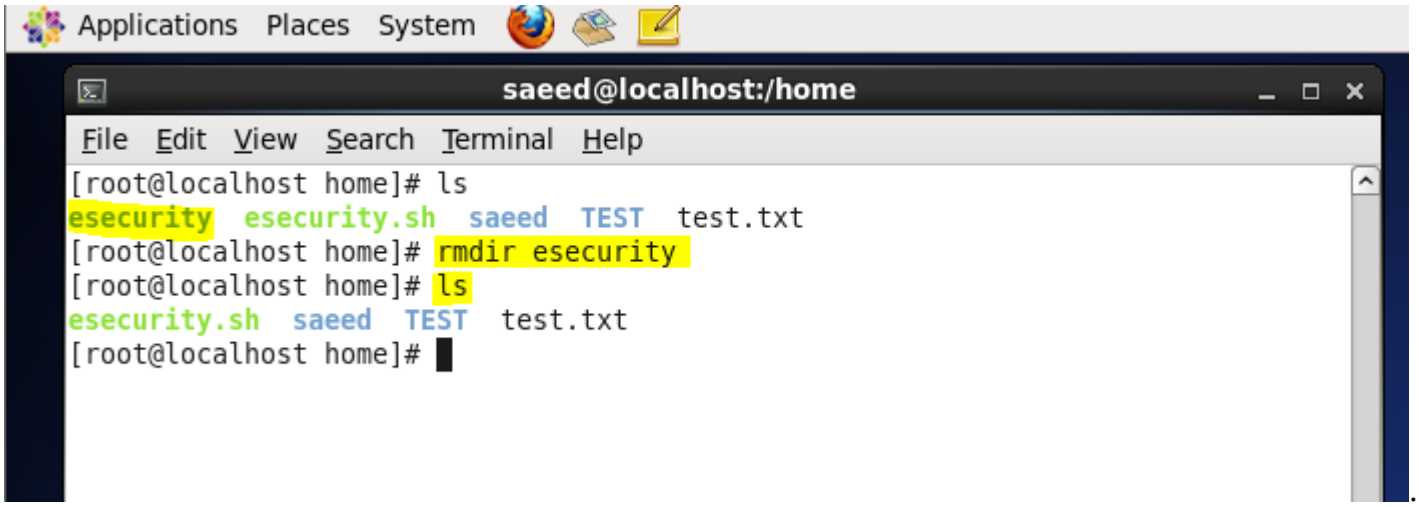
خطاهای متداولی که در استفاده از این دستور رخ می‌دهد :

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

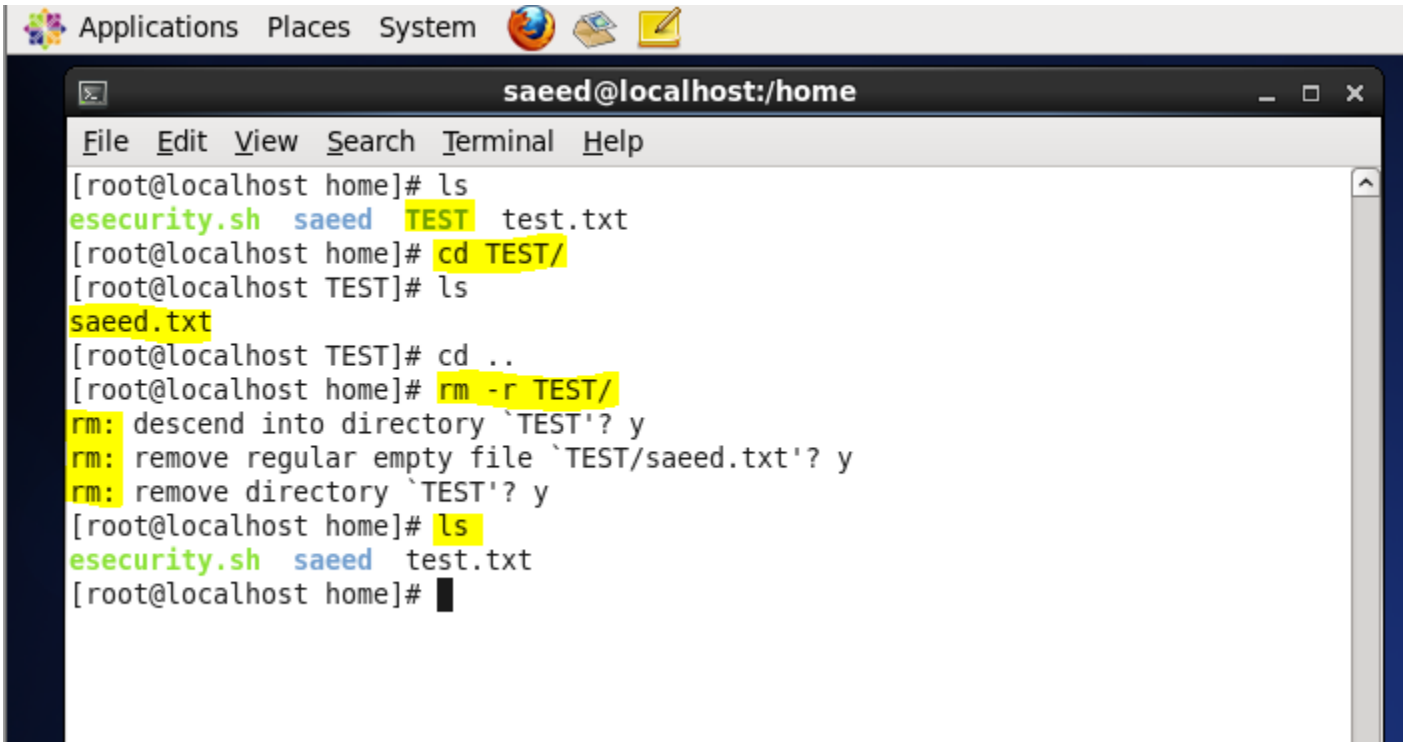
حذف یک دایرکتوری

برای این منظور میتوان از دستور زیر استفاده کرد :

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



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



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

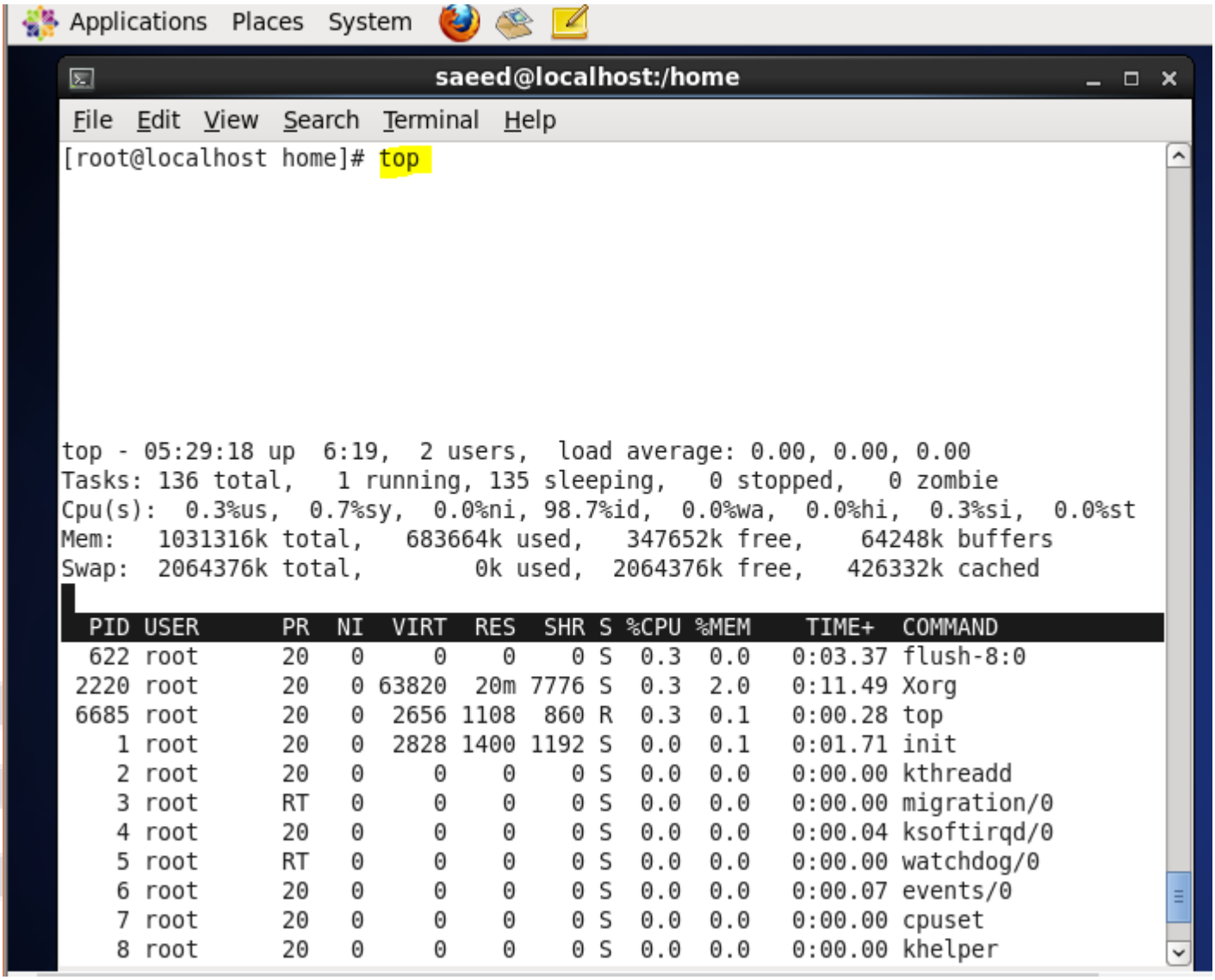
شما در این فصل مفهوم process و job در یک سیستم یونیکسی را خواهید آموخت .در یونیکس هر برنامه‌ای به عنوان یک process اجرا میشود. همچنین در این فصل به تعریف اصطلاحات foreground و background پرداخته شده و چگونی شروع یک processشرح داده میشود، در ادامه نحوه‌ی kill کردنprocess ها و در نهایت مفهوم والد و فرزند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

شروع یکprocess

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 منحصربه‌فرد دارد. و در هر زمان هر pid تنها به یک process اختصاص پیدا میکند. در واقع pid یک عدد ۱۶بیتی است که میتواند حداکثر تا ۳۲۷۶۷ باشد .

نمایش پروسس ها

شما در با وارد کردن دستور تاپ top می توانید کلیه پردازش هایی را که در سرور شما فعال می باشد را مشاهده نمائید. همچنین با وارد کردن دستور M با حروف بزرگ می توانید لیس پروسس ها را بر اساس میزان رم مصرفی منظم نمائید.
با دستور O با حروف بزرگ نیز کلیه حالات مختلف چینش پروسس ها را می توانید مشاهده نمائید:



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 سیستم‌عامل 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 دراینست 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 .

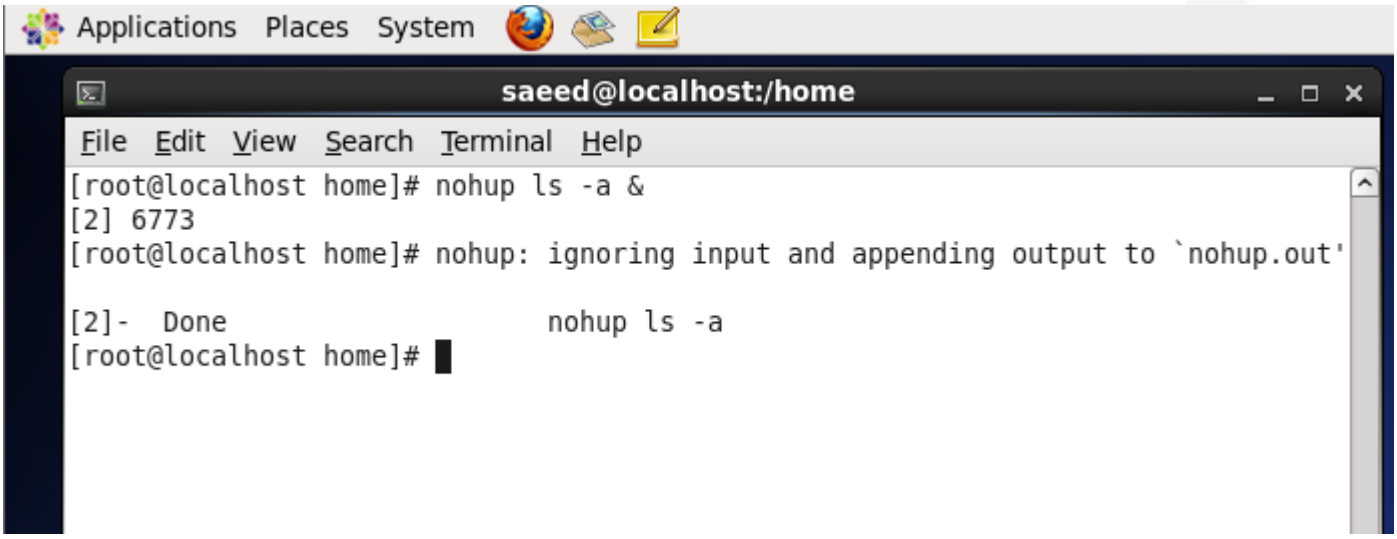
نکته: در صورتی که دستوری که به background فرستاده شده نیاز به ورودی داشته باشد، اجرای آن متوقف شده و تا زمانی که ورودی خود را دریافت کند منتظر میماند .

فرستادن یک دستور از foreground بهbackground

علاوه‌بر 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 .

دستورnohup

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:



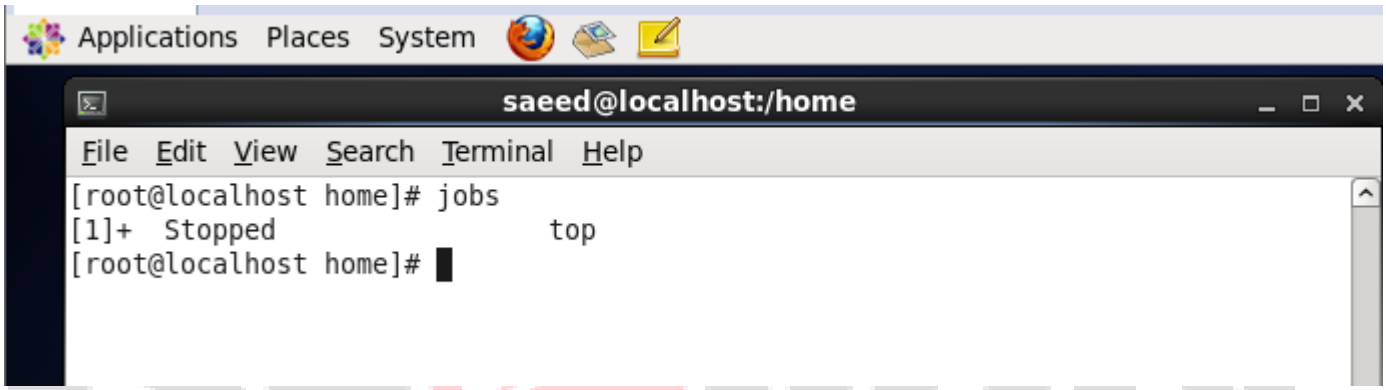
که در این حالت خروجی دستور در فایل nohup.out ذخیره خواهد شد .

لیستprocess های در حال اجرا

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

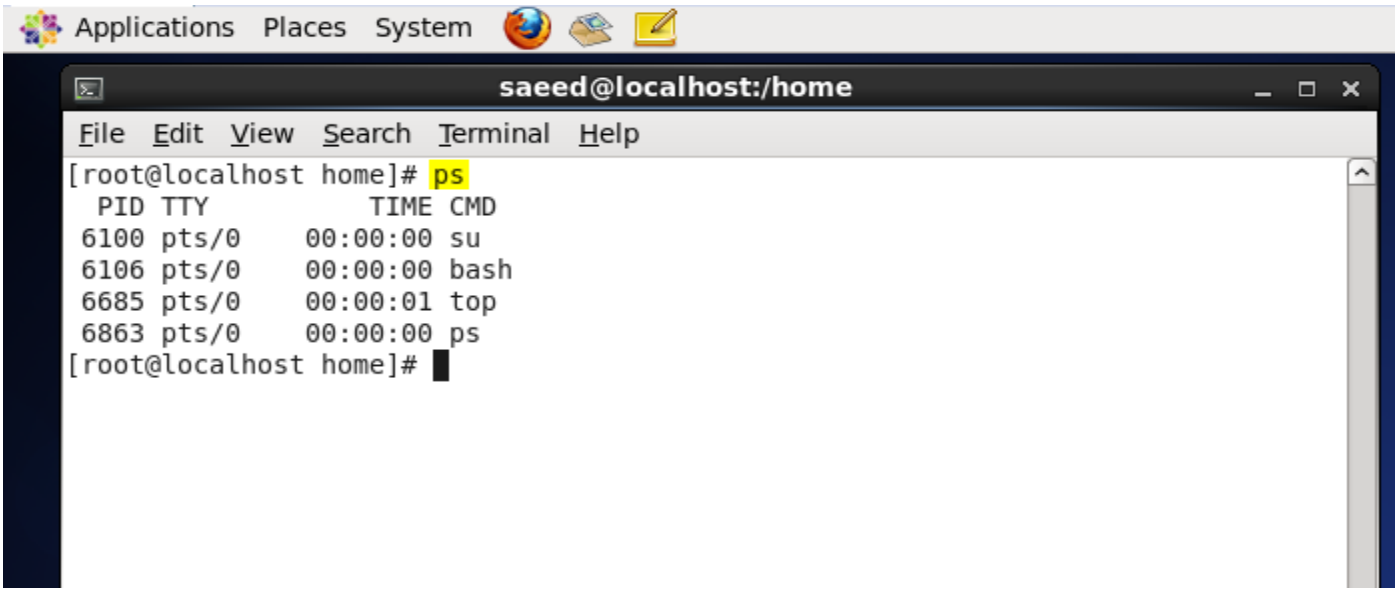
دستورjobs

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



دستور PS

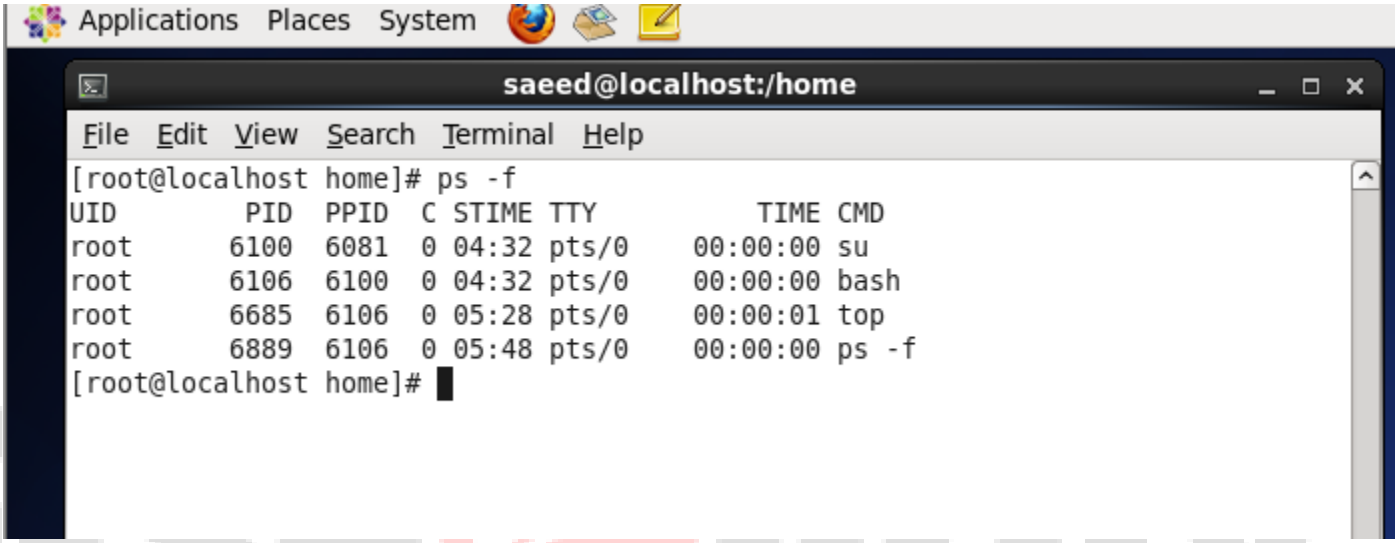
از این دستور میتوان برای نمایش تمامprocess های در حال اجرا استفاده کرد .



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.

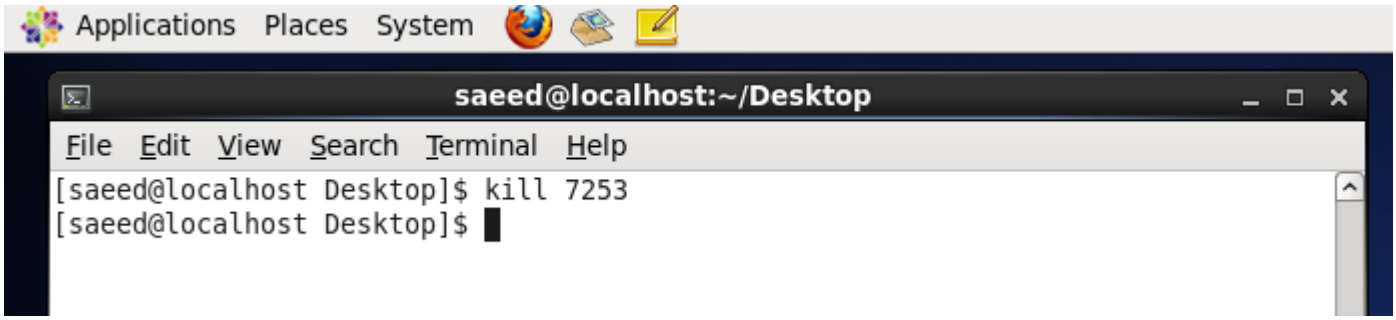


چهار ستون جدید در این حالت به شرح زیر هستند :

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

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

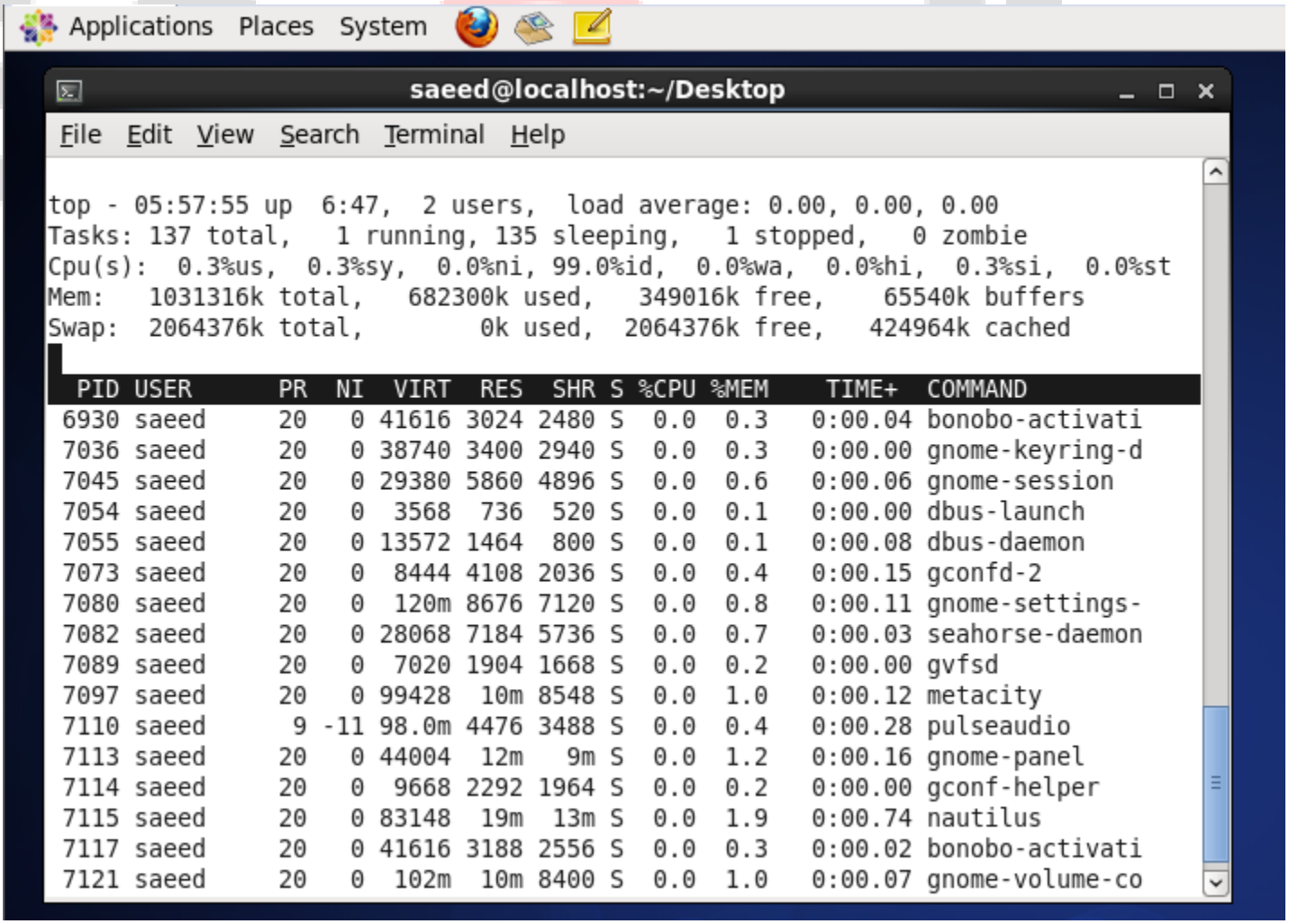
متوقف کردن یک پروسس
زمانیکه از دستور top استفاده می کنید، می توانید با استفاده از دستور k یک پروسس خاص را در صورتیکه از دسترسی لازم برخوردار باشید، حذف نمائید.
برای این کار باید پس از آن شماره پردازش (PID) مورد نظر را وارد نمائید.



نمایش پردازش های یک کاربر خاص

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

Top –u saeed



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 نمائید، 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 نمائید.
- 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 نمائید.
- 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 نمائید.

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 نمائید. 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 - متغییرها

متغیر ها محل هایی از حافظه اند که مقداری را در خو نگه می دارند. دارای نام، مقدار و نوع هستند. با نام متغیر به محل حافظه ای که مقدار را در خود نگه داشته مراجعه می کنیم. متغیر های BASH یا هر پوسته دیگر یونیکسی نیازی به تعیین نوع ندارند، بلکه با اختصاص مقدار به متغیر، نوع آن معلوم BASH می شود. متغیر ها می توانند مقادیر عددی، رشته ای، آرایه ای و یا حتی خروجی یک دستور را در خود نگه دارند. در BASH یا هر پوسته دیگری دو متغیر های محیطی دارای کاربرد های خاص هستند و برای منظور های 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 متغیر ها 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 PATHمقدار متغیر که مسیر دایرکتوری خانگی کاربر فعلی (کاربری که در حال حاضر در سیستم HOMEمی 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 نتغیر ها زمانی که به سیستم printenv,envمتغیر های محلی را با دستور های وجود PATHبرای اینکه ایا مسیر منتهی به دستور میان مسیر های 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 متغیر ها استفاده کنید مانند متغیر .(اطلاعات بیشتر) .دارد یا نه استفاده می کند نبود مجاز به اجرای rootاست 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 dadeh 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 MyVarمتفاوت است ولی در نامگذاری آنها نباید اعداد در ابتدای نام متغیر بیایند و همچنین در نامگذاری آنها از کاراکتر های خاص مانند @%^ اتسفاده نکنید. کاراکتر $ تعیین کننده متغیر است یعنی برای استفاده و چاپ متغیر باید علانت $ 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 زیز 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 هستنذ یعنی اگر پنجره یا tab جدیدی باز کنید دیگر متغیر اعتبار ندارد.

چاپ کردن متغیر ها

با استفاده از دستور های printenv و echi 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 echoباید 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 همره دیگر کارکتر ها چاپ می کند. پس برای چاپ مقدار یک متغیر به همراه دیگر کارکتر ها باید مانند خط اول زیر میان " " قرار گرد. 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 غر 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
```

متغییر 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 FreeBSDو 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 Loginشده 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 userasdd 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 آورن 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.

- $HOME : نام دایرکتوری خانگی کاربر را نشان می دهد.
- $HOSTTYPE یا $MACHTYPE : معماری ماشین یعنی ۳۲ بیتی یا ٦٤ 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 مقدارس 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 -4 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 (۱٤۱) نشان داده می شود.

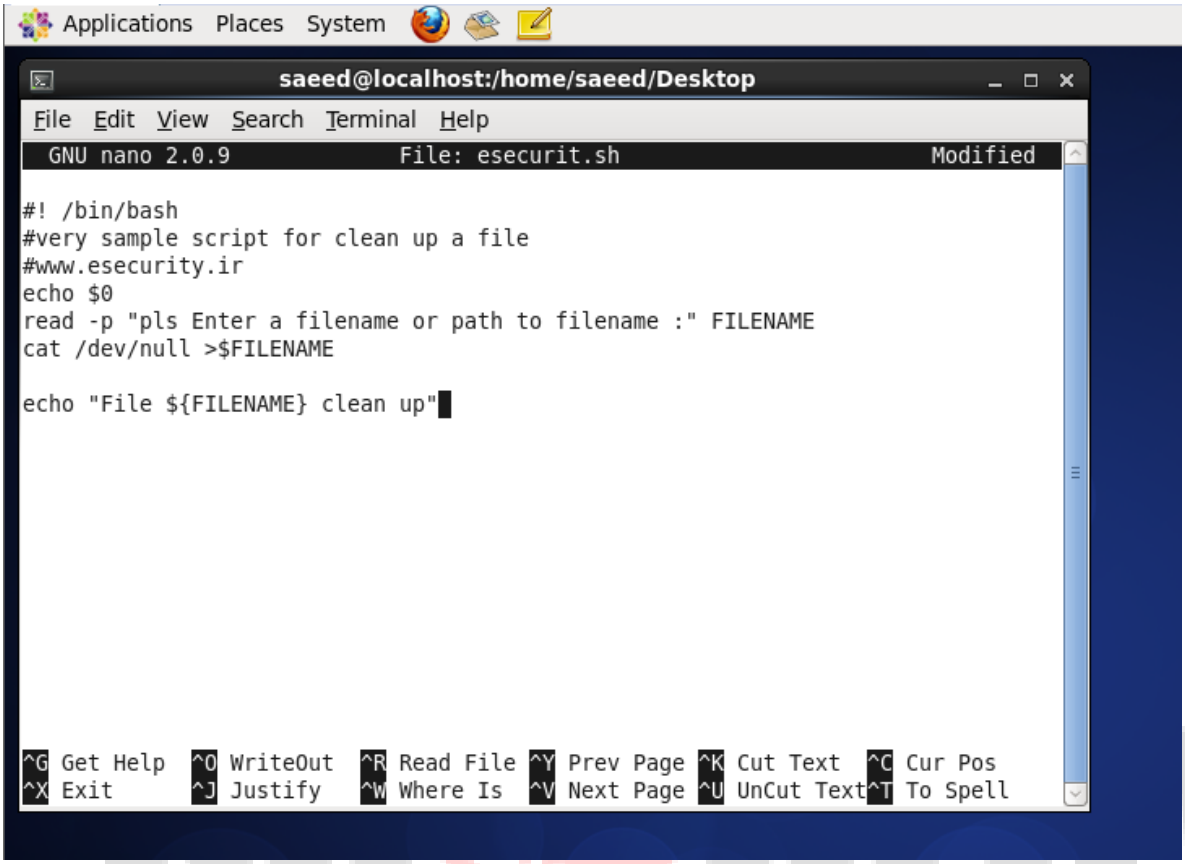
```
ls -l | wc -l

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 استفاد نمود.
- $SECONDS : تعداد ثانیه هایی که یک اسکریپت در حال اجرا شدن است.

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



- $#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.
- $NUMBERدر اصل این آرگومان ها بصورت $1و ۲$ و ۳$ و ... 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 ماخذ

فارسی

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

http://falearn.ir

http://bashlinux.persiangig.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/