ده فرمان ضروری از ترمینال لینوکس

نوشته جک والن

می دانم که خیلی از شما نمی خواهید از محیط دستوری لینوکس (یا هر سیستم عامل دیگری) استفاده کنید. اما حقیقت این است که برای یک مدیر سیستم خوب شدن، باید محیط دستوری را بشناسید. چرا؟ خوب، در ویندوز مواردی وجود دارد که محیط دستوری صرفا برای صرفه جویی در پوسته گرافیکی است. در لینوکس، محیط دستوری وسیع، قابل اعتماد، انعطاف پذیر و سریع است... من می توانم با آوردن صفات بیشتر ادامه دهم.

2119 دستور ممکن در پوشه /usr/bin (در مندریوا اسپرینگ 2008) و 388 دستور ممکن در /user/sbin کمی صرفنظر نکردنی به نظر می آیند. در اینجا ده فرمان از آن ها مطرح می شود که زندگی مدیر سیستم لینوکس – یا فرآیند آشنایی با لینوکس- را بسیار ساده می کند.

می توانستم این متن را با آوردن دستورات پر استفاده مثل (cd, ls, rm و غیره – خیلی خوب "غیره" فرمان نیست! ولی شما منظورم را متوجه می شوید) آسان کنم. اما در عوض، قصد دارم به سراغ مفیدترین دستورات بروم و تا جایی که بتوانم دستوراتی مستقل از توزیع بیاورم.

1. top

به نظرم مناسبتر بود که دستور top را اول بگذارم. با وجود اینکه top در واقع برای لیست کردن وظایف جاری استفاده می شود، اولین دستوری است که کاربران لینوکس برای فهمیدن اینکه چه برنامه ای از حافظه استفاده می کند یا اساسا کل حافظه چقدر است به آن رجوع می کنند. من اغلب اجازه می دهم دستور top روی دسکتاپ در حال اجرا باقی بماند به این ترتیب همیشه می توانم بفهمم چه اتفاقی پیش خواهد آمد. گاهی، من حتی یک ترمینال اجرا می کنم، پنجره اش را جایی که می خواهم قرار می دهم، بعد از آن حاشیه (بردر) پنجره را مخفی می کنم. بدون حاشیه، ترمینال نمی تواند جابجا شود به این ترتیب من همیشه به اطلاعاتی که نیاز دارم دسترسی سریع پیدا می کنم.

top یک سیستم گزارشی بلادرنگ است، به همین دلیل وقتی یک پروسه تغییر می کند، این موضوع بلافاصله در پنجره ترمینال نمایش داده می شود. top چند آرگومان مفید دارد (مثل آرگومان p– ، که پروسه های با PID ی تعیین شده از سوی کاربر را نمایش می دهد.) اما اجرای بدون پارامتر top، به صورت پیش فرض کلیه اطلاعات مربوط به تمامی وظایف جاری را نمایش می دهد.

2. ln

برای بسیاری از مدیران سیستم، لینک ها ابزار هایی فوق العاده گرانبها هستند که نه تنها باعث می شوند کاربران ساده تر زندگی کنند بلکه به شدت استفاده از فضای خالی دیسک را کاهش می دهند. اگر نمی دانید چطور لینک ها به شما کمک می کنند، اجازه دهید این سناریو ساده را مطرح کنم: شما یک تعدادی کاربر دارید که باید به یک دایرکتوری بزرگ (پر از فایل های بزرگ) روی یک درایو در طول روز دسترسی داشته باشند تمام کاربران از یک سیستم استفاده می کنند، و شما نمی خواهید تمام محتویات دایرکتوری را در پوشه ی مربوط به هر کاربر کپی کنید. در عوض در پوشه ی هر کاربر، یک لینک به دایرکتوری مذکور می سازید. شما فضایی مصرف نمی کنید و کاربران به آن دایرکتوری دسترسی سریع پیدا خواهند کرد. البته در هنگام افزایش درایو ها، شما باید از لینک های نرم (سمبولیک) استفاده کنید. یکی دیگر از استفاده های بی نظیر از لینک ها در مرتبط کردن دایرکتوری های متفاوت به پوشه ی اسناد وب سرور آپاچی است. این تکنیک نه تنها در مصرف فضای دیسک صرفه جویی می کند، بلکه از نقطه نظر امنیتی سودمند است.

3. tar/zip/gzip

tar، zip و gzip ابزار های آرشیو/فشرده سازی هستند که زندگی مدیر سیستمی شما را بسیار آسانتر می کنند. من این ها را با هم ترکیب کردم به این دلیل که این ابزار ها وظایف مشابهی انجام می دهند و با این وجود تفاوت هایی ساختاری با هم دارند. (تفاوتشان در حدی نبوده که آن ها را جداگانه مطرح کنم). بدون این ابزار ها، نصب کردن از روی سورس چیزی کمتر از آسان می بود. بدون این سه ابزار پشتیبان گرفتن نیازمند فضایی بیشتر از آنچه بود که شما معمولا در اختیار دارید. یکی از امکانات این ابزار ها که کمتر استفاده می شود، (اما اغلب بسیار مفید است) توانایی استخراج فایلهای مستقل از یک آرشیو است. هم اکنون انجام این عملیات با zip و gzip بسیار آسانتر از tar است. با tar ، برای استخراج یک فایل از آرشیو باید حجم دقیق فایلی که قرار است استخراج شود را بدانید. یکی از مواردی که tar/gzip/zip در آن مدیریت سیستم را ساده می کنند، ایجاد شل اسکریپت هایی است که برای خودکار کردن فرآیند پشتیبان گیری استفاده می شوند. هر سه ابزار می توانند در شل اسکریپت ها استفاده شوند و بهترین، بی دردسر ترین و مطمئن ترین ابزار های پشتیبان گیری هستند که شما خواهید یافت.

4. nano, vi, emacs

در اینجا نمی خواستم فقط یک ویرایشگر متن قرار دهم، از ترس اینکه به آتش جنگ vi در برابر emacs دامن زده باشم. برای کاهش آن، به نظرم رسید که ویرایشگر محبوبم – nano- را به ترکیب اضافه کنم. بعضی ها ممکن است اعتراض کنند که این ها بیشتر از اینکه دستور باشند، برنامه هستند. اما تمام این ابزار ها در محیط خط فرمان، استفاده می شوند، به همین دلیل من به آن ها "دستور" می گویم! بدون یک ویرایشگر متن خوب، مدیریت یک ماشین لینوکسی دچار اشکال می شود. تصور کنید، تلاش برای ویرایش /etc/fstab یا /etc/samba/smb.conf از OpenOffice استفاده شود. عده ای ممکن است بگویند، این که مشکل نیست!، ولی open Office به

انتهای فایل های متنی کاراکتر پایان–خط مخفی اضافه می کند، که می تواند فایل پیکربندی را ناکار آمد کند. برای ویرایش فایل های پیکربندی یا bash ، تنها راه قابل رفتن، همراه شدن با ویرایشگری مثل nano، vi یا emacs است.

5. grep

بسیاری از مردم این ابزار فوق العاده را اورکلاک می کنند. grep، خطوط منطبق با الگوی تعیین شده کاربر، را چاپ می کند. به عنوان نمونه، فکر کنید در یک فایل httpd. conf که بیش از 1000 خط نوشته دارد، به دنبال ورودی "AccessFileName .htaccess" می گردید. شما می توانید با comb فقط تا ورودی خط 429 را پیمایش کنید، یا می توانید از دستور

grep –n "AccessFileName .htaccess" /etc/httpd/ conf /http.conf

استفاده کنید. با وارد کردن این دستور، برگردانده می شود: 439:AccessFileName .htaccess این نشان می دهد که ورودی مورد نظر شما، شگفت تر از هر شگفتی، در خط 439 وجود دارد.

دستور grep برای لوله کشی سایر دستورات – به آن – نیز مفید است. یک مثال از این مورد، استفاده از فرمان ps (که از پروسه های جاری یک تصویر لحظه ای می گیرد) است. تصور کنید، می خواهید PID مرورگر تازه آسیب دیده فایرفاکس را بدانید. می توانید ps aux را وارد کنید و در نتیجه ی این فرمان به دنبال ورودی firef ox بگردید یا دستور ps aux|grep firef ox را وارد کنید که در آن صورت چیزی شبیه این می بینید:

```
jlwallen 17475 0.0 0.1 3604 1180 ? Ss 10:54 0:00 /bin/sh
/home/jwallen/firefox/firefox
jlwallen 17478 0.0 0.1 3660 1276 ? S 10:54 0:00 /bin/sh
/home/jlwallen/firefox/run-mozilla.sh /home/jlwallen/firefox/firefox-bin
jlwallen 17484 11.0 10.7 227504 97104 ? Sl 10:54 11:50
/home/jlwallenfirefox/firefox-bin
jlwallen 17987 0.0 0.0 3112 736 pts/0 R+ 12:42 0:00 grep --color firefox
```

به این ترتیب PID های هر firef ox ی که در حال اجراست، پیدا می کنید.

6. chmod

به کسی اجازه دسترسی بدهید؟ بدون کمک chmod مدیریت و امنیت لینوکس کار بسیار سختی می شد. تصور کنید قادر نباشید با chmod u+x یک شل اسکریپت را اجرایی کنید. البته کار این دستور فقط اجرایی کردن یک فایل نیست. بسیاری از ابزارهای وب به اجازه های دسترسی مشخصی حتی برای نصب شدن، نیاز دارند. برای این منظور، دستور chmod –R 666 DIRECTORY/ یکی از دستوراتی است که بسیار مورد سوء استفاده قرار می گیرد. بسیاری از کاربران تازه وارد، وقتی در نصب یک برنامه با مسئله دسترسی رو به رو می شوند، قبل از اینکه بدانند یک دایرکتوری دقیقا چه اجازه ای باید داشته باشد، یک راست به سراغ 666 می روند. با وجود

اینکه این دستور برای اداره کردن سیستم حیاتی است، باید قبل از استفاده، مطالعه شود. مطمئن شوید قبل از استفاده، ورودی ها و خروجی های chmod را می دانید. به یاد داشته باشید w = نوشتن، r = خواندن و x = اجرا کردن است. همچنین UGO ("کاربر"، "گروه" و "سایرین") را فراموش نکنید. UGO یک راه ساده برای به خاطر سپردن این موضوع است که کدام اجازه به چه کسی تعلق دارد. بنابراین اجازه rw- rw –rw- یعنی، "کاربر"، "گروه" و "سایرین" همه می تواند اجازه خواندن و نوشتن داشته باشند. بهترین کار این است که "سایرین" را به شدت محدود کنیم.

7. dmesg

اگر تمایل دارید می توانید من را سنتی صدا کنید، اما هر وقت که یک سیستم مبتنی بر لینوکس را روشن می کنم، اولین کاری که انجام می دهم این است که dmseg را اجرا می کنم. این فرمان پیام های بافر کرنل را نمایش می دهد. پس، آره، این یک دستور مهم است. از دستور dmseg اطلاعاتی زیادی می توان به دست آورد معماری سیستم، gpu، ابزار شبکه، آپشن های استفاده شده در بوت شدن کرنل، میزان حافظه RAM، و غیره را می توانید از این دستور بفهمید.

یک تکنیک خوب، لوله کشی dmseg به tail برای دیدن هر پیامی است که وارد dmseg می شود. برای اجرای این تکنیک، فرمان زیر را وارد کنید:

```
dmseg |tail –f
```

و آخرین خطوط dmseg در ترمینال باقی می مانند و هر بار که یک ورودی جدید برسد، به خط انتهای tail اضافه می شود. این پنجره را وقتی وظایف سنگین مدیریتی یا دیباگ انجام می دهید، باز نگه دارید.

8. kill/killall

یکی از بزرگترین مزیت های لینوکس، ثبات آن است. ولی آن ثبات، همیشه به برنامه های بیرون از کرنل اعمال نمی شود. بعضی از برنامه ها می توانند حقیقتا قفل کنند. و وقتی قفل کنند، شما می خواهید بتوانید از دستشان خلاص شوید. سریعترین راه برای خلاص شدن از برنامه های قفل کرده، استفاده از دستور kill/killall است. تفاوت بین دو دستور این است که دستور kill به (شناسه عددی پروسه) PID (Process ID number) نیاز دارد. و فرمان killall فقط به نام اجرایی نیاز دارد. بیاید فرض کنیم firefox فقل کرده است. برای بستنش با kill باید اول PID آن را با دستور ps aux|grep firefox پیدا کنید. به محض اینکه PID را یافتید، می توانید دستور kill PID (به جای PID شناسه عددی یافته شده را وارد کنید) را اجرا کنید. اگر نمی خواهید دنبال پیدا کردن PID بروید، می توانید دستور killall firefox را اجرا کنید. (در بعضی موارد killall firefox-bin نیاز است.) البته نمی توان (نباید) kill/killall را با برنامه های کمکی – (daemon) هایی – مثل Apache، Samba و غیره استفاده کرد.

man .9

تا به حال چند بار "RTFM" را دیده اید؟ خیلی ها می گویند این سر واژه ی "راهنمای مفید* را بخوان" است. البته این واژه در واقع برای قابل چاپ شدن تغییر یافته است. به نظر من این عبارت، سرواژه "صفحه man مفید، را بخوان" است. صفحات man برای دلیلی وجود دارند– برای کمک به شما در فهم نحوه استفاده از یک دستور -. صفحات man عموما با فرمت مشابهی نوشته می شوند، پس به محض اینکه به ادراکی از فرمت یک دستور دسترسی پیدا کردید، می توانید همه آن ها را بخوانید (و درک کنید). ارزش صفحه man را نادیده نگیرید، حتی اگر نمی توانید تمام اطلاعات داده شده را بفهمید، همیشه می توانید به پایین صفحه اسکرول کنید تا بفهمید که آرگومان ها چه کار می کنند و بهترین قسمت استفاده از صفحات man این است که وقتی کسی می گوید "RTFM" شما می توانید بگویید I have "RTFMd" (یعنی: من راهنمای مفید را خوانده و درک کرده ام – مترجم)

mount/unmount .10

بدون این دو دستور، استفاده از حافظه های جدا شدنی یا افزودن درایو های خارجی ممکن نمی شود. دستور mount/unmount برای نصب کردن یک درایو در یک دایرکتوری در ساختار فایل لینوکس استفاده می شود (اغلب به صورت /dev/sda طبقه بندی می شوند). هر دو دستور mount و unmount از فایل /etc/fstab استفاده می کنند که کاربردشان را بسیار آسانتر می کند. به عنوان مثال، اگر برای /dev/sda1 در فایل /etc/fstab یک ورودی باشد که آن را به /data نگاشت کند ، آن درایو می تواند با دستور mount /data نصب شود. واضح است که، mount/unmount باید با سطح دسترسی کاربر ریشه اجرا شوند (مگر آنکه، fstab یک ورودی داشته باشد که به کاربر استاندارد (عادی) اجازه ی نصب و حذف درایو ها را بدهد). همچنین شما می توانید دستور mount را بدون هیچ آرگومانی اجرا کنید به این ترتیب تمامی درایو های که نصب شده درکنار مسیری که به آن نگاشت شده اند را می توانید ببینید..(به همراه نوع فایل سیستم و اجازه های دسترسی آن ها)

نمی توان بدون آن ها زیست!

این ها ده فرمانی بودند که مدیریت سیستم لینوکس را ممکن می سازند. فرمان های مفید دیگری هم وجود دارند، به همراه فرمان هایی که بسیار بیشتر از این ها استفاده می شوند. اما دستورات ذکر شده، در دسته ضروری قرار می گیرند. شما را نمی دانم، ولی من نمی توانم یک روز را بدون استفاده از حداقل نیمی از آن ها، بگذرانم. آیا شما هم یکی دو فرمان دارید که نمی توانید بدون آن زندگی کنید؟ اگر چنین است، به ما اطلاع دهید.