

فصل 5- هدایت ورودی/خروجی ها

در این فصل در خصوص راهکارهای قدرتمند یونیکس در هدایت ورودی، خروجی و خطاها صحبت می شود. مباحث این فصل شامل موارد زیر می باشد:

- ورودی، خروجی و خطاهای استاندارد
- عملگرهای هدایت
- چگونگی استفاده خروجی یک دستور به عنوان ورودی دستوری دیگر
- هدایت ورودی
- اداره کردن پیام های خطای استاندارد
- ترکیب هدایت جریان ورودی، خروجی و خطا
- فیلترهای خروجی

5.1: ورودی و خروجی های استاندارد چه هستند؟

اکثر دستورات لینوکسی ورودی را به عنوان فایل ورودی یا دیگر خصوصیات دستور می خوانند، و نتیجه کار را در خروجی می نویسند. به طور پیش فرض ورودی از صفحه کلید خوانده شده و خروجی روی صفحه به نمایش در می آید. صفحه کلید شما به عنوان دستگاه «ورودی استاندارد» (stdin) و صفحه نمایش شما یا هر پنجره پایانه دیگر به عنوان دستگاه «خروجی استاندارد» (stdout) خوانده می شود.

با این حال، از آن جا که لینوکس یک سیستم انعطاف پذیر است، لزومی ندارد که در آن حتما این تنظیمات پیش فرض به کار برده شود. به عنوان نمونه خروجی استاندارد در سروری که شدیداً تحت کنترل و بررسی است ممکن است یک پرینتر باشد.

5.1.1 عملگرهای هدایت

5.1.1.1 هدایت خروجی با < و |

گاهی اوقات می خواهید خروجی یک دستور را در یک فایل بنویسید یا اینکه از آن به عنوان ورودی یک دستور دیگر استفاده کنید. به این کار هدایت خروجی می گویند. هدایت با استفاده از عملگرهای «<» (علامت بزرگتر)، یا «|» (پایپ) انجام می گیرد که خروجی استاندارد یک دستور را به ورودی استاندارد یک دستور دیگر می فرستد.

همانطور که قبلاً دیدیم دستور **cat** چند فایل را به هم می چسباند و آن ها را با هم در خروجی استاندارد نمایش می دهد. با هدایت خروجی به یک فایل، فایل نامبرده ساخته می شود یا روی فایل هم نام قبلی نوشته می شود، پس مواظب باشید:

```
nancy:~> cat test1 some words
```

```
nancy:~> cat test2 some other words
```

```
nancy:~> cat test1 test2 > test3
```

```
nancy:~> cat test3 some words some other words
```

هدایت «هیچ چیز» به یک فایل موجود معادل خالی کردن آن فایل می باشد:

```
nancy:~> ls -l list
```

```
rw-rw-r-- 1 nancy nancy 117 Apr 2 18:09 list-
```

```
nancy:~> > list
```

```
nancy:~> ls -l list -rw-rw-r-- 1
```

```
nancy nancy 0 Apr 4 12:01 list
```

این عمل کوتاه کردن (truncating) نامیده می شود.

هدایت هیچ چیز به فایل که وجود ندارد موجب ساخته شدن یک فایل خالی می گردد:

```
nancy:~> ls -l newlist
```

```
ls: newlist: No such file or directory
```

```
nancy:~> > newlist
```

```
nancy:~> ls -l newlist
```

```
rw-rw-r-- 1 nancy nancy 0 Apr 4 12:05 newlist-
```

فصل 7 شامل مثال های بیشتری در مورد این نوع هدایت می باشد.

چند مثال در مورد پایپ کردن دستورات:

برای یافتن کلمه ای در یک متن، تمام خطوط حاوی «pattern1» را نمایش بده و از نمایش تمام خطوط شامل «pattern2» خودداری کن:

```
grep pattern1 file | grep -v pattern2
```

برای نمایش صفحه به صفحه خروجی لیست محتویات یک دایرکتوری:

```
ls -la | less
```

برای جستجوی یک فایل در یک دایرکتوری:

```
ls -l | grep part_of_file_name
```

5.1.1.2- هدایت ورودی

در مواردی دیگر شما می خواهید فایلی را به عنوان ورودی به دستوری بدهید که به طور عادی ورودی به صورت فایل قبول نمی کند. این هدایت ورودی توسط عملگر «>» (علامت کوچکتر) انجام می شود.

در زیر مثالی از ارسال یک فایل به یک نفر توسط هدایت ورودی مشاهده می گردد:

```
andy:~> mail mike@somewhere.org < to_do
```

اگر کاربر *mike* روی سیستم وجود دارد نیازی به تایپ کامل آدرس نیست. ولی اگر می خواهید به شخصی روی اینترنت دسترسی داشته باشید آدرس کامل را به عنوان آرگومان ورودی وارد نمایید.

این دستور کمی ناخوانا تر از دستور مبتدی **cat file | mail someone** می باشد ولی بدون شک یک راه صحیح تر استفاده از امکانات موجود می باشد.

5.1.1.3 ترکیب هدایت ها

دستور زیر هدایت ورودی و خروجی را ترکیب می کند. ابتدا اشکالات املایی فایل **text.txt** بررسی می شود و سپس خروجی به یک فایل **log** خطاها هدایت می شود:

```
aspell < text.txt > error.log
```

ممکن است از نگارش زیر برای کار فوق استفاده کنید (به صفحات راهنما رجوع کنید):

```
aspell -H list < file.txt | sort -u
```

این هم نوعی از هدایت ورودی و خروجی ها می باشد دستور زیر تمام دستوراتی را که با استفاده از **less** می توانید با آن یک فایل دیگر را بررسی کنید لیست می کند:

```
mike:~> less --help | grep -i examine
```

.e [file] Examine a new file:

.n * Examine the (N-th) next file from the command line:

.p * Examine the (N-th) previous file from the command line:

.x * Examine the first (or N-th) file from the command line:

گزینه **i** به منظور جستجوی غیر حساس به حروف بزرگ و کوچک استفاده شده است. به یاد داشته باشید که یونیکس به شدت به حروف بزرگ و کوچک حساس است.

اگر می خواهید خروجی این دستور را برای مراجعات بعدی در اختیار داشته باشید، خروجی آن را به یک فایل هدایت کنید:

```
mike:~> less --help | grep -i examine > examine-files-in-less
```

```
mike:~> cat examine-files-in-less
```

.e [file] Examine a new file:

.n * Examine the (N-th) next file from the command line:

.p * Examine the (N-th) previous file from the command line:

.x * Examine the first (or N-th) file from the command line:

خروجی یک دستور به طور مجازی می تواند بارها به ورودی یک دستور دیگر پیاف شود، این عمل تا زمانی که این دستور به طور معمولی از ورودی استاندارد می خواند و در خروجی استاندارد می نویسد می تواند انجام شود. گاهی اوقات دستورات

این کار را نمی کنند ولی ممکن است این دستورات گزینه هایی داشته باشند که آن ها را وادار به این کند که با تعاریف استاندارد کار کنند. بنابراین در صورت برخورد به خطا صفحات راهنما (man و info)ی دستورات را مطالعه کنید.

روی فایل های موجود ننویسید!

مواظب باشید که هنگام هدایت خروجی، روی فایل ها (ی مهم) بازنویسی نکنید. در بسیاری از shell ها از جمله Bash امکانی درون ساخته وجود دارد که شما را از این خطر در امان نگه می دارد: noclobber. صفحات info را برای اطلاعات بیشتر مطالعه کنید. در bash می توانید با اضافه کردن دستور set -o noclobber در فایل پیکره بندی .bashrc. تان مانع بازنویسی اتفاقی فایل های تان شوید.

5.1.1.4 عملگر <<

به جای باز نویسی اطلاعات فایل تان می توانید از دو علامت بزرگتر متوالی برای اضافه کردن به محتویات فایل تان استفاده کنید:

```
mike:~> date >> wishlist
```

```
mike:~> cat wishlist
```

```
more money
```

```
less work
```

```
Thu Feb 28 20:23:07 CET 2002
```

دستور date در حالت عادی خط آخر را نمایش می دهد. اما اکنون خروجی آن به آخر wishlist اضافه شده است.

5.1.2 ویژگی های پیشرفته هدایت

5.1.2.1: استفاده از توصیف گر های فایل ها

سه نوع ورودی/خروجی، با عددی مخصوص به خود به نام توصیف گر های فایل موجودند:

- ورودی استاندارد: 0
- خروجی استاندارد: 1
- خطای استاندارد: 2

در توصیف گر های زیر اگر از عدد توصیف گر فایل صرف نظر شده باشد و کاراکتر اول عملگر هدایت > باشد، هدایت به سمت ورودی استاندارد خواهد بود (توصیف گر فایل 0). اگر اولین کاراکتر عملگر هدایت < باشد هدایت به سمت خروجی استاندارد می باشد (توصیف گر فایل 1).

چند مثال عملی این موضوع را روشن تر می کند:

```
ls > dirlist 2>&1
```

هر دوی خروجی استاندارد و خطای استاندارد را به dirlist هدایت می کند، در حالی که دستور:

```
ls 2>&1 > dirlist
```

فقط خروجی استاندارد را به dirlist هدایت می کند. این یک امکان مناسب برای برنامه نویسان است.

تمام این موضوعات به طور تفصیلی در صفحات info مورد بررسی قرار گرفته است.

5.1.2.2: چند مثال

5.1.2.2.1: تحلیل خطاها

اگر process شما خطاهای زیادی تولید می کند این یک راه برای بررسی کامل آن است

```
command 2>&1 | less
```

این کار معمولا در هنگام ساخت نرم افزاری جدید با make صورت می گیرد. مثلا:

```
--andy:~/newsoft> make all 2>&1 | less --output omitted
```

5.1.2.2.2: جداسازی خروجی استاندارد از خطای استاندارد

چنین کاری معمولا توسط برنامه نویسیان صورت می پذیرد و در آن خروجی را به یک پایانه نمایشی می فرستند و خطا را به یکی دیگر، ابتدا با استفاده از tty ببینید که از کدام شبه ترمینال استفاده می کنید:

```
andy:~/newsoft> make all 2> /dev/pts/7
```

5.1.3: فیلترها

وقتی برنامه ای عملیاتی را روی ورودی انجام دهد و حاصل را در خروجی بنویسد یک فیلتر نامیده می شود. یکی از معمول ترین استفاده ها از فیلترها بازسازی خروجی است. در زیر چند فیلتر مهم را بررسی می کنیم.

5.1.3.1: کمی بیشتر درباره grep

همانطور که در بخش 3.3.4 دیدیم، grep خروجی را خط به خط در جستجوی یک الگو بررسی می کند. تمام خطوط حاوی الگوی مزبور در خروجی استاندارد نوشته می شوند. این عملکرد می تواند توسط گزینه -v معکوس شود.

چند مثال: فرض کنید می خواهیم ببینیم چند فایل در یک دایرکتوری در فوریه دستکاری شده اند:

```
jenny:~> ls -la | grep Feb
```

دستور grep، مانند بسیاری از دستورات، حساس به حروف بزرگ و کوچک است. برای اینکه آن را نسبت به این موضوع غیر حساس کنید از گزینه -i استفاده کنید. ویژگی های اضافی GNU بسیاری نیز موجود است. مثلا -colour که در هایلایت کردن عبارات طولانی جستجو مفید است، یا -after-context که تعداد خطوط را پس از آخرین خط می نویسد. شما می توانید از گزینه -۲ برای جستجوی بازگشتی دایرکتوری ها از یک دایرکتوری مشخص استفاده کنید. مانند قبل گزینه های مختلف می توانند با هم ترکیب شوند.

5.1.3.2: فیلتر کردن خروجی

دستور sort به طور پیش فرض خطوط خروجی را به ترتیب الفبایی مرتب می کند.

```
thomas:~> cat people-I-like | sort
```

```
Auntie Emmy
```

```
Boyfriend
```

Dad

Grandma

Mum

My boss

ولي sort بسياري کارهاي ديگر نيز مي تواند انجام دهد. به عنوان نمونه نگاه کردن اندازه فایل. با این دستور خروجي بر حسب اندازه فایل ها مرتب مي شود:

ls -la | sort -nk 5

نگارش قديمي sort

ممکن است نتايج مشابهي را به شکل ls -la | sort +4n بدست آورید ولي این يك راه قديمي است که با استانداردهاي فعلي همخواني ندارد

.دستور sort به صورت ترکیبي با برنامه uniq نيز به کار مي رود (يا sort -u) تا بعد از مرتب سازي موارد مشابه حذف شوند.

5.2: خلاصه

در این فصل آموختيم که چگونه مي توان چند دستور را با هم ترکیب کرد و چگونه خروجي يك دستور را ميتوان به عنوان ورودی يك دستور ديگر به کار برد.

هدايت ورودی ها و خروجي ها به عنوان يك عمل معمول در ماشين هاي لينوکسي و انجام مي پذيرد. این مکانيزم قدرتمند امکان استفاده انعطاف پذيري از اجزاي سازنده يونیکسي به ما مي دهد.

5.3: تمرين ها

این تمرين ها نمونه هاي بيشتري از ترکیب دستورات ارائه مي کند. هدف استفاده هر چه کمتر از کلید Enter مي باشد.

- دستور cut را روی خروجي يك ليست بلند از محتويات يك دایرکتوري به کار ببرید تا فقط مجوزهاي فایل ها نمایش داده شود. سپس خروجي را به sort و uniq بفرستید تا تمام خطوط تکراري حذف شوند و در نهایت از WC براي شمارش انواع مختلف مجوزهاي فایل در این دایرکتوري استفاده نمایید.
- خروجي date را در يك فایل قرار دهید. خروجي ls را به این فایل اضافه نمایید. این فایل را به صندوق نامه هاي محلي خودتان ارسال کنید (از چيزي مثل <domain.com> استفاده نکنید، فقط نام کاربري را وارد کنید). اگر از bash استفاده مي کنید يك پیام مبني بر اینکه يك نامه جديد دریافت کرده اید مشاهده خواهید کرد.
- دستگاه هاي در /dev/ را که UID شما از آن استفاده کرده است ليست کنید. خروجي را در less پایپ کنید تا به طور قابل استفاده نمایش داده شوند.
- از دستور bash -x براي اجراي دایمون httpd با استفاده از اسکريپت موجود در /etc/rc.d/init.d/ استفاده کنید. خطا ها را به يك فایل بفرستید. همین کار را مجدداً تکرار کنید ولي این بار خروجي را به يك فایل بفرستید. تفاوت را مشاهده نمایید.

- در حال حاضر چند process اجرا کرده اید؟
- چند فایل مخفی در دایرکتوری خانگی شما موجود است؟
- از locate برای یافتن مستندات در باره کرنل استفاده کنید.
- فایلی که حاوی خط زیر است را پیدا کنید:

```
root:x:0:0:root:/root:/bin/bash
```

و همچنین این یکی:

```
system: root
```

- ببینید با اجرای دستور زیر چه اتفاقی می افتد:

```
time; date >> time; cat < time <
```