جلسه‌ی دوم — ورودی و خروجی در پوسته

در این جلسه با مدیریت ورودی و خروجی در پوسته و استفاده از لوله برای ترکیب دستورها آشنا خواهید شد.

مدیریت ورودی و خروجی در پوسته

در حالت عادی، برنامه‌هایی که توسط پوسته اجرا می‌شوند منتظر دریافت ورودی از کاربر می‌شوند و خروجی خود را در صفحه چاپ می‌کنند. امکان «Redirection» در پوسته، فرستادن خروجی یک برنامه به یک فایل و خواندن ورودی آن از یک فایل را ممکن می‌سازد. این کار با استفاده از علامت‌های کوچک‌تر و بزرگ‌تر به صورت زیر قابل انجام است:

```
$ cmd >path          #  خروجی دستور «cmd» را به فایل مشخص شده با آدرس «path» می‌نویسد
$ cmd 1>path         #  معادل دستور قبل
$ cmd >>path         #  خروجی دستور «cmd» را به انتهای فایل «path» اضافه می‌کند
$ cmd <path          #  ورودی دستور «cmd» را از فایل «path» می‌خواند
$ cmd 2>path         #  خروجی خطای دستور «cmd» را به فایل «path» می‌نویسد
$ cmd 1>path 2>&1    #  خروجی دستور «cmd» و خروجی خطای آن را به فایل «path» می‌نویسد
```

استفاده از لوله در پوسته

می‌توان خروجی یک برنامه را توسط لوله[1] به برنامه‌ی دیگری فرستاد؛ یک لوله، که با علامت «|» نشان داده می‌شود، دو سر دارد؛ خروجی برنامه‌ای که در سمت چپ لوله قرار گرفته است به عنوان ورودی به برنامه‌ای که در سمت راست آن قرار گرفته داده می‌شود. در مثال زیر خروجی برنامه‌ی cmd1 به عنوان ورودی به برنامه‌ی cmd2 فرستاده می‌شود:

```
$ cmd1 | cmd2        #  خروجی دستور «cmd1» را به صورت ورودی به دستور «cmd2» می‌دهد
```

همان طور که در مثال زیر نشان داده شده است، تعداد زیادی دستور می‌توانند به وسیله‌ی لوله به صورت زنجیره‌ای با هم ترکیب شوند تا خروجی هر دستور به عنوان ورودی به دستور بعدی انتقال یابد.

1 Pipe

```
$ cmd | grep "error" | sort | uniq | head -n10
```

در محیط یونیکس برنامه‌های زیادی وجود دارند که کار ساده‌ای را انجام می‌دهند. لوله امکان ترکیب این برنامه‌ها را ایجاد می‌کند. برنامه‌های زیادی در یونیکس برای چنین کاربردی طراحی شده‌اند:

```
$ cat                 #  ورودی دستور را بدون تغییر چاپ می‌کند
$ wc                  #  تعداد خط‌ها، کلمه‌ها و حرف‌های ورودی را چاپ می‌کند
$ sort                #  خط‌های ورودی را به صورت مرتب شده چاپ می‌کند
$ uniq                #  خط‌های ورودی را پس از حذف خط‌های تکراری متوالی چاپ می‌کند
$ tac                 #  خط‌های ورودی را از آخر به اول چاپ می‌کند
$ grep kwd            #  خط‌های ورودی که شامل الگوی «kwd» هستند را چاپ می‌کند
$ head -n X           #  «X» خط اول ورودی را چاپ می‌کند
$ tail -n X           #  «X» خط آخر ورودی را چاپ می‌کند
$ tee out             #  مشابه دستور «cat» با این تفاوت که یک کپی از ورودی را در فایل «out» می‌نویسد
$ rev                 #  خط‌های ورودی را با معکوس کردن ترتیب حرف‌های آنها چاپ می‌کند
$ shuf                #  خط‌های ورودی را با ترتیب تصادفی چاپ می‌کند
$ seq X               #  اعداد یک تا «X» را در خروجی چاپ می‌کند
$ tr X Y              #  حرف «X» در خط‌های ورودی را با «Y» جابجا می‌کند و آنها را چاپ می‌کند
$ fmt                 #  پاراگراف‌های ورودی را به خط‌های تقریبا هم اندازه می‌شکند
$ cut -f X            #  ستون شماره‌ی «X» از خط‌های ورودی را چاپ می‌کند
```

برای اطلاعات بیشتر در مورد این دستورات، به صفحه‌ی راهنمای آنها (با دستور «man cmd») مراجعه شود. توجه به این نکته نیز لازم است که بیشتر این دستورات، با گرفتن آدرس تعدادی فایل به عنوان پارامتر، محتوای آن فایل‌ها را به عنوان ورودی در نظر می‌گیرند. برای نمونه دستور «cat xyz.txt» محتویات فایل «xyz.txt» را چاپ می‌کند و «head -n5 xyz.txt» پنج خط اول فایل «xyz.txt» را چاپ می‌کند. بنابراین، سه دستور زیر خروجی یکسانی دارند:

```
$ cat xyz.txt | head -n5      #  استفاده از لوله
$ head -n5 <xyz.txt           #  استفاده از Redirection
$ head -n5 xyz.txt            #  مشخص کردن فایل به عنوان پارامتر
```

یکی از برنامه‌های پرکاربرد محیط یونیکس برای تغییر محتوای فایل‌ها «sed» می‌باشد. این دستور عملیات زیادی را برای تغییر جریان ورودی پشتیبانی می‌کند. یکی از مهم‌ترین کاربردهای این دستور، جایگزینی یک الگوی عبارت منظم در جریان ورودی با رشته‌ی دیگری است. ادامه شیوه‌ی انجام این کار نشان داده شده است (برای کاربردهای بیشتر، به صفحه‌ی راهنمای این دستور مراجعه شود).

```
$ sed 's/pat/rep/g'        #        برای خط‌های ورودی الگوی «pat» را با عبارت «rep» جایگزین می‌کند
$ sed '/pat/d'             #        خطوط ورودی شامل الگوی «pat» را حذف و بقیه را چاپ می‌نماید
```

در این مثال‌ها، الگوی «pat» می‌تواند یک عبارت منظم باشد.

تمرین دوم

مشابه تمرین یک، پس از دریافت فایل فشرده‌ی «git-2.6.0.tar.gz»، آن را در شاخه‌ی «~/git-2.6.0/» خانه باز نمایید. سپس شاخه‌ی «ex2» را در شاخه‌ی خانه‌ی خود ایجاد نمایید و فایل‌های خواسته شده را در تمرین‌های زیر (بجز قسمت الف)، با توجه به فایل‌های موجود در شاخه‌ی «~/git-2.6.0/» بسازید.

الف) دستورات زیر را اجرا کنید، توضیح دهید چه عملی انجام می‌دهند و در چه شرایطی مفید هستند؛ در صورت نیاز به صفحه‌ی راهنمای آنها مراجعه نمایید.

```
$ grep malloc test.c | wc -l        # «test.c» یک فایل ورودی نمونه می‌باشد
$ seq 12 | shuf | head -n1          # عدد ۱۲ را می‌توانید تغییر دهید
$ cat list | sort | uniq | wc -l    # فایل «list» می‌تواند فهرستی از نام‌ها باشد
```

ب) فایل «~/ex2/query1.out» را بسازید که شامل فهرست فایل‌هایی که شامل عبارت «get_indexed_object» هستند، باشد.

ج) فایل «~/ex2/query2.out» را بسازید که شامل خطهای ۵۹ تا ۱۰۰ فایل «quote.c» باشد.

د) فایل «~/ex2/query3.out» ایجاد کنید که شامل خطهای متمایز شامل عبارت «#include» در فایل‌های با پسوند «.c» باشد.