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       # write the output of cmd to the file at <path>
$ cmd >>path      # append the output of cmd to the end of <path>
$ cmd <path       # read the input of cmd from the file at <path>
$ cmd 2>path      # the standard error to <path>
$ cmd 2>path 1>&1 # write the standard output and error to <path>
```

استفاده از لوله در پوسته

همچنین می‌توان خروجی یک برنامه را توسط لوله[1] به برنامه‌ی دیگری فرستاد. در مثال زیر خروجی برنامه‌ی cmd1 به عنوان ورودی به برنامه‌ی cmd2 فرستاده می‌شود:

```
# send the output of <cmd1> to <cmd2>
$ cmd1 | cmd2
# an example combining several commands with pipe
$ cmd | grep "error" | sort | uniq | head -n10
```

همان طور در خط دوم نشان داده شده است، تعداد زیادی دستور می‌توانند به وسیله‌ی لوله با هم ترکیب شوند تا خروجی هر دستور به عنوان ورودی به دستور بعدی انتقال یابد. استفاده از لوله در پوسته امکان ترکیب برنامه‌های کوچکی که یک کار را انجام می‌دهند ایجاد می‌کند. برنامه‌های زیادی در یونیکس برای

۱ Pipe

چنین کاربردی طراحی شده‌اند:

```
$ cat           # write the input to the output verbatim
$ wc            # count the number of lines, words, and characters
$ sort          # sort input lines
$ uniq          # remove duplicate consecutive lines
$ tac           # reverse the order of the input lines
$ grep kwd      # print only lines matching <kwd>
$ grep -v kwd   # print lines not matching <kwd>
$ head -n X     # print the first <X> lines
$ tail -n X     # print the last <X> lines
$ tee out       # like cat, but save a copy of input in file <out>
$ rev           # reverse the characters in each line
$ tr X Y        # translate character <X> to <Y>
$ fmt           # format the input (by breaking long lines)
$ cut -f X      # print selected columns
```

البته لازم به اشاره است که بیشتر این دستورات، با گرفتن آدرس تعدادی فایل به عنوان پارامتر، محتوای آن فایل‌ها را به عنوان ورودی در نظر می‌گیرند. برای اطلاعات بیشتر در مورد این دستورات، به صفحه‌ی راهنمای آنها (با دستور «man cmd») مراجعه شود.

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 که در ادامه نشان داده شده است (برای کاربردهای بیشتر، به صفحه‌ی راهنمای این دستور مراجعه شود).

```
# replace <pat> with <rep> in the input stream
$ sed 's/pat/rep/g'   # <pat> can be a regular expression
# remove lines containing <pat>
$ sed '/pat/d'
```

تمرین دوم

پس از دریافت فایل فشرده‌ی git-2.6.0.tar.gz آن را در شاخه‌ی /git-2.6.0 باز نمایید. سپس شاخه‌ی ex2 را در شاخه‌ی خانه‌ی خود ایجاد نمایید و فایل‌های زیر را در این شاخه، با توجه به فایل‌های موجود در /git-2.6.0 بسازید.

الف) فایل query1.out: لیست فایل‌هایی که شامل عبارت «get_indexed_object» هستند.

ب) فایل query2.out: خطهای ۵۹ تا ۱۰۰ فایل «quote.c».

ج) فایل query3.out: خطهای متمایز شامل عبارت «#include» در فایل‌های با پسوند «.c».