جلسه‌ی اول — آشنایی با پوسته

در این جلسه با مقدمات استفاده از رابط پوسته[۱] برای مدیریت فایل‌ها در محیط‌های مشابه یونیکس[۲] آشنا خواهید شد.

ساختار فایل سیستم

فایل سیستم در یونیکس دارای یک ساختار درختی است که رأس‌های میانی آن شاخه[۳] هستند. این درخت از ریشه (که با علامت «/» نشان داده می‌شود) شروع می‌شود. شکل زیر ساختار درختی یک فایل سیستم نمونه را نشان می‌دهد.

```
/
├── bin/
│   ├── cat
│   ├── echo
│   ├── ls
│   ├── ps
│   └── sed
├── boot/
├── etc/
├── home/
│   └── user/
├── lib/
├── mnt/
├── root/
├── usr/
│   ├── bin/
│   ├── man/
│   └── lib/
└── var/
    └── log/
```

با پیمودن مسیر شروع شده از ریشه به فایل‌ها و شاخه‌ها در این درخت آدرس آنها تعیین می‌شود: شاخه‌های این مسیر از ریشه از چپ به راست کنار هم قرار داده می‌شوند و با علامت «/» جدا می‌گردند.

۱ Shell
۲ Unix like
۳ Directory

برای نمونه آدرس فایل ls در شکل قبل «/bin/ls» می‌باشد. هر شاخه در فایل سیستم دو زیر شاخه‌ی مجازی دارد: «.» به همان شاخه و «..» به شاخه‌ی بالاتر از آن شاخه اشاره می‌کند. بنابراین دو آدرس «/root/» و /./root/../bin/ به یک شاخه اشاره می‌کنند.

به هر پردازه (از جمله پوسته) در سیستم عامل شاخه‌ای به نام شاخه‌ی جاری[1] اختصاص داده می‌شود. با توجه به این شاخه می‌توان آدرس فایل‌ها و شاخه‌ها را به صورت نسبی بیان کرد؛ آدرس‌های نسبی آدرس‌هایی هستند که با «/» شروع نمی‌شوند. برای یافتن مقصد این آدرس‌ها، آدرس شاخه‌ی جاری به ابتدای آنها اضافه می‌گردد. به عنوان مثال، در صورتی که شاخه‌ی جاری «/home/user/» باشد، آدرس نسبی «/..» به شاخه‌ی «/home/» اشاره می‌کند.

برخی از دستورات ابتدایی پوسته و محیط یونیکس در شکل زیر نشان داده شده‌اند.

```
$ pwd              # Print shell's current working directory
$ cd path          # Change shell's working directory
$ cd ~             # Change the working directory to user's HOME
$ cd               # Equivalent to "cd ~"
$ ls               # List the files in the current directory
$ find -name "pat" # Find files whose names matches "pat"
$ find path -name "pat"  # Find files whose names matches "pat" in "path"
$ mkdir XYZ        # Create a directory named XYZ
$ rmdir XYZ        # Remove directory XYZ; it should be empty
$ rm XYZ           # Remove the file at XYZ
$ rm -r XYZ        # Recursively remove files and directories in XYZ
$ cp TEST dir1/    # Copy TEST to dir1/TEST
$ cp -r dir1/ dir2 # Copy the directory dir1 to dir2
$ mv dir1 dir3     # Move directory dir1 to dir3
$ echo "Hello!" >TEST    # Create the file TEST containing "Hello!"
$ cat TEST         # Print the contents of the file TEST
$ passwd           # Change login password
$ ls --help        # Show ls' options
$ man ls           # The manual page of ls (press q to exit)
$ date             # Print system date and time
```

یکی از ویژگی‌های پوسته که مشخص کردن تعداد زیادی فایل را آسان می‌کند، ویژگی گسترش نام فایل[2] در آن می‌باشد. پوسته عبارت‌های شامل علامت‌های «?»، «*» یا «[..]» را به عنوان الگوی فایل‌ها می‌پذیرد و آن عبارت را با فهرست فایل‌هایی که با آنها مطابقت دارند جایگزین می‌کند. در این الگوها «?» با هر حرفی، «*» با هر رشته‌ای و «[..]» با هر یک از حروف مشخص شده در آن مطابقت می‌کنند. برای

۱ Current working directory

۲ File name expansion

مثال «[hc].*» با نام همه‌ی فایل‌های شاخه‌ی جاری که پسوند «.c» یا «.h» دارند جایگزین می‌گردد.

علاوه بر گسترش نام فایل‌ها، پوسته عبارت‌های دیگری را نیز گسترش می‌دهد. نام‌های پس از علامت $ با مقدار متغیر پوسته یا مقدار متغیر محیطی با آن نام جایگزین می‌گردند.

```
$ echo $VAR      # print the value of VAR environment/shell variable
$ echo ${VAR}    # same as the previous command
$ VAR="abc"      # assign "abc" to variable VAR
```

دستورات را در پوسته می‌توان به شکل‌های گوناگونی ترکیب نمود. در ادامه چند مثال برای ترکیب دستورها نشان داده شده است.

```
# a) execute cmd1 and then cmd2
$ cmd1; cmd2
# b) replace `cmd1` with the output of cmd1 and then execute cmd2
$ cmd2 `cmd1`
# c) execute cmd2 only if cmd1 succeeds (returns zero)
$ cmd1 && cmd2
# d) execute cmd2 only if cmd1 fails (returns nonzero)
$ cmd1 || cmd2
```

در پایان هر دستور در پوسته مانند برخی از زبان‌های برنامه‌نویسی می‌توان علامت ; را قرار داد. در صورتی که دو دستور مستقل در یک خط بیان گردند می‌توان آنها را با این علامت جدا ساخت (قسمت a از شکل قبل). همچنین، پوسته قبل از اجرای یک دستور، عبارت‌های داخل دو علامت «`» را اجرا می‌کند و آنها را با خروجی‌شان جایگزین می‌نماید (قسمت b). در قسمت‌های c و d از شکل قبل دو دستور به صورت شرطی با هم ترکیب می‌گردند: موفقیت یک دستور با توجه به کد برگشتی[1] آن دستور تعیین می‌گردد: به صورت قراردادی در صورتی که یک برنامه مقدار صفر را به عنوان کد برگشتی برگرداند موفقیت آمیز بوده است و در غیر این صورت مشکلی در اجرای برنامه بوجود آمده است. برای مثال دستور ls در صورتی که به آن یک آدرس غیر موجود به عنوان ورودی داده شود، مقداری غیر صفر بر می‌گرداند.

---

۱ Return code

تمرین یک

پس از دریافت فایل فشرده‌ی git-2.6.0.tar.gz آن را با این دستور در شاخه‌ی git-2.6.0/ در شاخه‌ی جاری باز نمایید:

```
$ tar xzvf git-2.6.0.tar.gz
```

سپس شاخه‌ای با نام ex1 در شاخه‌ی خانه‌ی خود بسازید که ساختاری مانند شکل زیر داشته باشند. دقت نمایید که فایل‌های این درخت را باید از فایل‌های باز شده در git-2.6.0/ بگیرید.

```
git/
├── include/
│   ├── diff.h
│   ├── khash.h
│   ├── refs.h
│   ├── tar.h
│   ├── url.h
│   └── utf8.h
├── src/
│   ├── diff.c
│   ├── pager.c
│   ├── refs.c
│   ├── url.c
│   └── utf8.c
├── contacts/
│   ├── Makefile
│   ├── git-contacts
│   └── git-contacts.txt
├── git-log.sh
└── git-clean.sh
```