جلسه‌ی سوم اسکریپت‌های پوسته

در این جلسه با عبارت‌های شرطی، حلقه‌ها و اسکریپت‌های پوسته آشنا خواهید شد.

اسکریپت‌های پوسته

می‌توان دنباله‌ای از دستورات پوسته را در یک فایل قرار داد تا آنها را در هنگام نیاز اجرا نمود؛ به این فایل‌ها اسکریپت پوسته[1] گفته می‌شود. در مثال زیر، سه دستور در فایل «cmd.sh» قرار داده می‌شوند و سپس دستورات موجود در این فایل توسط پوسته‌ی «sh» اجرا می‌گردند.

```
$ cat >cmd.sh                          #          نوشتن سه خط در فایل «cmd.sh»
date
sleep 1
date
^D
$ sh <cmd.sh                           #          اجرای اسکریپت «cmd.sh»
Sat Oct 17 14:27:40 IRST 2015
Sat Oct 17 14:27:41 IRST 2015
$ sh cmd.sh                            #          معادل دستور قبل
Sat Oct 17 14:27:57 IRST 2015
Sat Oct 17 14:27:58 IRST 2015
```

همان طور که مشاهده می‌شود پوسته‌ی «sh» دستورات ورودی را یکی پس از دیگری اجرا می‌کند و در صورتی که یک فایل به عنوان پارامتر به آن داده شود، به جای خواندن ورودی، آن را اجرا می‌نماید.

متغیرهای پوسته

در پوسته می‌توان متغیر تعریف کرد و از آنها استفاده نمود؛ پوسته عبارت «$name» را با مقدار متغیر «name» جایگزین می‌کند.

1 Shell script

```
$ var="abc"                    # تعریف متغیر «var» با مقدار «abc»
$ echo $var                    # مشاهده‌ی مقدار متغیر «var»
abc
$ var="`pwd`"                  # انتساب خروجی دستور «pwd» به متغیر «var»
$ echo $var                    # مشاهده‌ی مقدار متغیر «var»
/home/user
```

پارامترهایی که به یک اسکریپت فرستاده می‌شوند نیز به صورت مشابهی قابل دسترسی هستند؛ «$1» با پارامتر اول، «$2» با پارامتر دوم و در حالت کلی «${n}$» با پارامتر «n»-ام جایگزین می‌شوند. در مثال زیر، اسکریپت «cmd.sh» سه پارامتر اول خود را چاپ می‌نماید.

```
$ cat >cmd.sh
echo "Arg #1: $1"
echo "Arg #2: $2"
echo "Arg #3: $3"
^D
$ sh cmd.sh abc def ghi
Arg #1: abc
Arg #2: def
Arg #3: ghi
```

موفقیت دستورها در پوسته

بدیهی است که دستوراتی که در پوسته اجرا می‌شوند می‌توانند موفقیت‌آمیز باشند یا ناموفق خاتمه یابند. برای مثال، در صورتی که آدرس شاخه‌ای که وجود ندارد به دستور «cd» داده شود، این دستور نمی‌تواند شاخه‌ی جاری را تغییر دهد و با خطا خاتمه می‌یابد. اجرای موفق یک دستور به صورت قراردادی با کد برگشتی[1] آن مشخص می‌شود (برای مثال، در زبان C کد برگشتی، مقداری است که از تابع «main» برگشت داده می‌شود). اجرای موفق یک دستور به صورت قراردادی با کد برگشتی صفر مشخص می‌گردد. پوسته کد برگشتی آخرین دستور اجرا شده را در متغیری به نام «?» قرار می‌دهد:

```
$ cd                           # نمونه‌ای از یک دستور موفق
$ echo $?                      # چاپ کد برگشتی بعد از یک دستور موفق
0
$ cd xyz                       # نمونه‌ای از یک دستور ناموفق
```

1 Return code

```
$ echo $?                              #          چاپ کد برگشتی بعد از یک دستور ناموفق
1
```

در پوسته عبارت‌های شرطی با توجه بـه موفقیت اجرای دستورها اجرا می‌شونـد. از ایـن رو، دستـوری به نـام «test» وجود داد که موفقیت آن با توجه به برقرار بودن شرطهای مشخص شده، تعیین می‌شود.

```
$ test "abc" == "def"          #     موفقیت، در صورتی که رشته‌ی اول با دوم برابر باشد
$ test "abc" != "def"          #     موفقیت، در صورتی که رشته‌ی اول با دوم برابر نباشد
$ test -f xyz.txt              #     موفقیت، در صورتی که «xyz.txt» یک فایل باشد
$ test -d xyz                  #     موفقیت، در صورتی که «xyz» یک شاخه باشد
$ test ! -d xyz                #     موفقیت، در صورتی که «xyz» یک شاخه نباشد
```

برای اطلاع از سایر شرطهای دستور «test»، به صفحه‌ی راهنمای این دستـور مراجعه نماییـد. دستور «true» همواره موفق است (به صورت مشابه دستور «false» همواره ناموفق است) و می‌توان از آن بـرای حلقه‌ی بی‌نهایت استفاده نمود.

عبارت‌های شرطی و حلقه‌ها در پوسته

برای تکرار تعدادی دستور بـه ازای مقادیـر یا فایـل‌های مختـلف، پوسته حلقـه‌های «for» و «while» را ارائه می‌دهد. حلقه‌ی «for» دنباله‌ای از کلمه‌ها را دریافت می‌کند و به ازای هر یک از آنها یک بار اجرا می‌شود. برای نمونه، حلقه‌ی زیر، شاخه‌های «dir1»، «dir2» و «dir3» را می‌سازد.

```
$ for dir in dir1 dir2 dir3          #این حلقه به ازای همه‌ی عبارت‌های پس از «in» تکرار می‌شود
do
        mkdir $dir
done
```

در حلقه‌ی قبل، در هر بار اجرای حلقه، یکی از رشته‌های مشخص شده بعد از کلمه‌ی کلیدی «in» (در این مثال «dir1»، «dir2» و «dir3») به متغیر پوسته‌ی مشخص شده بعد از کلمه‌ی کلیدی «for» (در این مثال «dir») منسوب می‌گردد. در تعیین دنبـاله‌ی کلمـات برای حلقـه‌ی «for» می‌تـوان از ویژگـی گسترش نـام فایل در پوسته نیز استفاده نمود. در مثال زیر، نام همه‌ی فایل‌ها با پسونـد «.h» در شاخه‌ی «/usr/include/» چاپ می‌شود.

```
#           اجرای بدنه‌ی حلقه به ازای همه‌ی فایل‌های با پسوند «.h» در شاخه‌ی «/usr/include»
$ for f in /usr/include/*.h
do
        echo $f
done
```

با استفاده از «if» در پوسته می‌توان تعدادی دستور را در صورت برقراری شرطی اجرا نمود. بدنه‌ی «if» تنها در صورتی که اجرای دستور مشخص شده بعد از کلمه‌ی کلیدی «if» موفق باشد، اجرا می‌شود. در مثال زیر، اگر شاخه‌ی «xyz» وجود نداشته باشد، ساخته می‌شود:

```
$ if test ! -d xyz      #                   اجرای بدنه در صورتی که شاخه‌ی «xyz» موجود نباشد
then
         mkdir xyz
fi
```

حلقه‌ی «while» تا وقتی که دستوری که بعد از کلمه‌ی کلیدی «while» مشخص می‌شود با موفقیت اجرا می‌شود، اجرا می‌گردد. برای مثال، دستور زیر تا ساخته‌شدن شاخه‌ی «xyz» صبر می‌کند.

```
$ while test ! -d xyz   #                   تکرار بدنه‌ی حلقه تا وقتی «test» با موفقیت اجرا شود
do
         sleep 1
done
```

یکی از پر استفاده‌ترین کاربردهای حلقه‌های «while» خواندن خطوط ورودی می‌باشد؛ این کار را می‌توان با استفاده از دستور داخلی پوسته «read» به صورت زیر انجام داد:

```
$ find /usr/include -name '*.h' | while read ln
do
         basename "$ln"
done | sort | uniq
```

هر خط ورودی یک بار به متغیر «ln» منسوب می‌شود و بدنه‌ی حلقه یک بار برای آن تکرار می‌گردد. همان طور که در این مثال نشان داده شده است، مشابه دستورات معمولی، حلقه‌ها نیز می‌توانند در دنباله‌ی لوله‌ها استفاده شوند. بنابراین، خروجی دستور «find» به حلقه‌ی «while» فرستاده می‌شود و خروجی این حلقه به

دستور «sort» و خروجی این دستور نیز به دستور «uniq» فرستاده می‌شود. مجموعه‌ی این دستورات، نام‌های متمایز همه‌ی فایل‌های با پسوند «.h» در شاخه‌ی «/usr/include/» را چاپ می‌کند. دستور «basename» که در بدنه‌ی حلقه فراخوانی شده است، نام فایل در آدرس داده شده را چاپ می‌کند. چگونگی استفاده از دستور «basename» و «dirname» در ادامه نشان داده می‌شود.

```
$ basename /usr/include/stdio.h          # چاپ نام فایل در یک آدرس
stdio.h
$ basename /usr/include/stdio.h .h       # چاپ نام فایل بدون پسوند «.h»
stdio
$ dirname /usr/include/stdio.h           # چاپ نام شاخه در یک آدرس
/usr/include
```

تمرین سوم

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

```
$ cmp f1 f2 || echo "Files do not match"          # فایل‌های «f1» و «f2» ورودی هستند
$ cmp f1 f2 && echo "Files match"                 # فایل‌های «f1» و «f2» ورودی هستند
$ test -d dir && echo "Directory already exists" # شاخه‌ی «dir» ورودی است
```

ب) همه‌ی فایل‌های شامل رشته‌ی «get_indexed_object» را از شاخه‌ی «/git-2.6.0» به شاخه‌ی «~/ex3» کپی نمایید و رشته‌ی «get_indexed_object» در این فایل‌ها را با «get_iobject» جایگزین کنید.

ج) شاخه‌ی «~/ex3/c» را بسازید و همه‌ی فایل‌های شاخه‌ی «~/ex3» که پسوند «.c» دارند را به این شاخه انتقال دهید و پسوند آنها را به «.txt» تغییر دهید.

د) اسکریپتی برای قسمت ب بنویسید که با گرفتن نام یک شاخه (پارامتر اول) همه‌ی فایل‌های موجود در آن شاخه و زیر شاخه‌های آن که شامل یک عبارت ورودی هستند (پارامتر دوم) را با عبارت دیگری (پارامتر سوم) جایگزین کند و در شاخه‌ی جاری قرار دهد.