الگوها

انطباق الگو در پوسته، دو نقش ایفا می‌کند: انتخاب نام فایل‌ها درون یک دایرکتوری، یا تعیین آنکه آیا یک رشته با یک قالب دلخواه مطابقت می‌نماید.

در خط فرمان شما غالباً از جانشین‌ها (globs) استفاده می‌کنید. جانشین‌ها به طور مساعدی شکل ساده الگوها هستند که می‌توانند به آسانی برای انطباق با گروهی از فایل‌ها به کار بروند، یا متغیرها را در برابر قواعد ساده بررسی کنند.

دومین نوع انطباق الگوها، glob های توسعه یافته را در بر می‌گیرند که نسبت به جانشین‌های معمولی، کاربرد عبارت‌های پیچیده‌تری را اجازه می‌دهند.

الگوهای جانشین

انطباق الگو در پوسته، دو نقش ایفا می‌کند: انتخاب نام فایل‌ها درون یک دایرکتوری، یا تعیین آنکه آیا یک رشته با یک قالب دلخواه مطابقت می‌نماید.

جانشین‌ها (globs) اگر فقط برای راحتی باورنکردنی‌شان باشد هم مفهوم بسیار مهم در BASH می‌باشند. درک صحیح glob ها به طُرق بسیاری برای شما مفید خواهد بود. جانشین‌ها اساساً الگوهایی می‌باشند که می‌توانند برای انطباق با نام فایل‌ها یا سایر رشته‌ها به کار بروند.

جانشین‌ها مرکب از کاراکترهای معمولی و فوق کاراکترها هستند. فوق کاراکترها، آن کاراکترهایی هستند که معنی ویژه‌ای دارند. فوق کاراکترهای اصلی عبارتند از:

*: بر هر رشته‌ای ازجمله رشته تهی منطبق می‌گردد.

?: بر یک کاراکتر منفرد منطبق می‌شود.

[...]: بر هر یک از کاراکترهای محصور در کروشه‌ها منطبق می‌شود.

جانشین‌ها به طور صریح از هر دو طرف مهار می‌گردند. این به آن معناست که یک جانشین بایستی بر تمام رشته (نام فایل یا رشته داده‌ای) منطبق شود. *a با رشته cat منطبق نیست، به علت آنکه فقط بر at، منطبق می‌شود، نه بر تمام رشته. درحالی‌که یک جانشین *ca، با رشته cat منطبق می‌گردد.

در اینجا مثالی در مورد اینکه چگونه می‌توانیم از الگوهای جانشین برای بسط نام فایل‌ها استفاده کنیم، ارائه شده است:

```
$ ls
a  abc  b  c
$ echo *
a abc b c
$ echo a*
a abc
```

BASH جانشین را می‌بیند، به عنوان مثال *a را و این جانشین را از طریق نگاه کردن به دایرکتوری جاری و مطابقت glob با تمام فایل‌های

موجود در آن، بسط می‌دهد. هر نام فایلی که با الگوی جانشین مطابقت داشته باشد، به شمار آمده و به جای جانشین به کار می‌رود. در نتیجه جمله echo a* با جمله echo a abc تعویض شده و بعدا اجرا شده است.

BASH بسط نام فایل را بعد از تفکیک کلمه‌ای که قبلاً انجام داده است، اجرا می‌نماید، بنابراین، نام فایل‌های ایجاد شده توسط جانشین، همیشه به طور صحیح به کار خواهد رفت. برای مثال:

```
$ touch "a b.txt"
$ ls
a b.txt
$ rm *
$ ls
$
```

در اینجا، * به نام یک فایل منفرد "a b.txt" بسط یافته است. این نام فایل به عنوان یک شناسه منفرد به فرمان rm تحویل می‌گردد. مهم است که بدانیم کاربرد جانشین‌ها برای به شمار آوردن نام فایل‌ها همواره از ایده به کارگیری دستور `ls` برای این منظور، بهتر است. به مثال زیر توجه نمایید:

```
$ ls
a b.txt
$ for file in `ls`; do rm "$file"; done
rm: cannot remove `a': No such file or directory
rm: cannot remove `b.txt': No such file or directory
$ for file in *; do rm "$file"; done
```

```
$ ls
$
```

در اینجا از فرمان for برای پوشش دادن تمام خروجی دستور ls استفاده کرده‌ایم. دستور ls رشته a b.txt را به خروجی می‌دهد. دستور for آن رشته را به کلمات تفکیک می‌کند و به تعداد آن کلمات تکرار را انجام می‌دهد. در نتیجه، for ابتدا برای a و بعد هم برای b.txt تکرار می‌شود. بدیهی است، این، آنچه ما می‌خواهیم نیست. درحالی‌که glob، به شکل صحیح بسط می‌یابد و فایل "a b.txt" را نتیجه می‌دهد که فرمان for آن را به عنوان یک شناسه منفرد دریافت می‌کند.

BASH همچنین از یک ویژگی به نام جانشین‌های توسعه یافته پشتیبانی می‌کند. این جانشین‌ها در ماهیت قدرتمندتر هستند. از لحاظ فنی، آن‌ها معادل عبارت‌های معمولی هستند، اگرچه ساختار آن‌ها به ظاهر متفاوت با آنچه اکثریت مردم به کار می‌برند، باشد. این ویژگی به طور پیش‌فرض غیر فعال است، لیکن می‌توانید با دستور shopt که برای تغییر وضعیت گزینه‌های پوسته به کار می‌رود، فعال شود. این دستور کوته‌نوشتی از عبارت shell options می‌باشد:

```
$ shopt -s extglob
```

- (list)?: صفر یا یک مورد تطابق با الگوی داده شده.
- (list)*: هر یا هیچ مورد انطباق با الگوی مورد اشاره.
- (list)+: یک مورد انطباق با الگو یا بیشتر.
- (list)@: انطباق با یکی از نمونه‌های داده شده.
- (list)!: با هر چیزی غیر از موارد ذکر شده انطباق می‌یابد.

کلمه list داخل پرانتزها لیستی از جانشین‌های معمولی یا توسعه یافته می‌باشد که با کاراکتر | از یکدیگر جدا شده‌اند. مثال:

```
$ ls
names.txt  tokyo.jpg  california.bmp
$ echo !(*jpg|*bmp)
names.txt
```

در اینجا الگوی جانشین (list) به هر چیزی که بر *jpg یا *bmp منطبق نمی‌شود بسط داده می‌شود. فقط فایل‌های متن همان‌طور که بسط یافته‌اند به دستور تحویل شده‌اند.

علاوه بر بسط نام فایل، از جانشین‌ها می‌توان برای بررسی انطباق داده‌ها با یک قالب مشخص شده نیز استفاده نمود. برای مثال، ممکن است نام فایلی را داده باشیم و انتظار عملیات متفاوت بر اساس پسوند فایل را داشته باشیم:

```
$ filename="somefile.jpg"
$ if [[ $filename = *.jpg ]]; then
> echo "$filename is a jpeg"
> fi
somefile.jpg is a jpeg
```

کلمه کلیدی]] و دستور داخلی case (که بعداً با تفصیل بیشتری شرح داده می‌شوند) هر دو فرصت بررسی یک رشته در برابر جانشین معمولی و یا جانشین توسعه یافته درصورتی‌که فعال شده باشد را فراهم می‌کنند.

سپس بسط ابرو ({}) را داریم. از نظر تکنیکی بسط ابرو در زمره جانشین‌ها نمی‌باشد، اما مشابه آن است. جانشین‌ها فقط به نام فایل‌های حقیقی بسط می‌یابند، در جایی که بسط ابرو به هر جایگردی از الگو بسط خواهد یافت. در اینجا چگونگی کارکرد آن را مشاهده می‌کنید:

```
$ echo th{e,a}n
then than
$ echo {/home/*,/root}/.*profile
/home/axxo/.bash_profile /home/lhunath/.profile
/root/.bash_profile /root/.profile
$ echo {1..9}
1 2 3 4 5 6 7 8 9
$ echo {0,1}{0..9}
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17
18 19
```

عبارت‌های منظم

عبارت‌های منظم (regex) مشابه الگوهای جانشین هستند، اما در BASH نمی‌توانند برای انطباق با نام فایل به کار بروند. از نگارش ۳,۰، BASH عملگر =~ در کلمه کلیدی]] را پشتیبانی می‌کند. این عملگر رشته‌ای را که قبل از آن می‌آید با الگوی regex که بعد از آن می‌آید، مطابقت می‌دهد. موقعی که رشته با الگو منطبق گردد، کلمه کلیدی]] یک کد خروج ۰ (true) بر می‌گرداند. اگر رشته با الگو مطابقت نداشته باشد، یک کد خروج ۱ (false) باز گردانده می‌شود. درصورتی‌که ترکیب

دستوری الگو معتبر نباشد،]] از عملیات صرف نظر نموده و یک کد خروج ۲ صادر می‌کند.

BASH از عبارت منظم توسعه یافته (ERE) نیز استفاده می‌کند.

الگوهای عبارت منظم که برای گرفتن گروه‌ها (پرانتزها) به کار می‌روند، رشته‌های گرفته شده‌شان را برای بازیابی بعدی، به متغیر BASH_REMATCH، تخصیص خواهند داد. به مثال زیر توجه نمایید:

```
$ langRegex='(..)_(..)'
$ if [[ $LANG =~ $langRegex ]]
> then
> echo "Your country code (ISO 3166-1-alpha-2) is ${BASH_REMATCH[2]}."
> echo "Your language code (ISO 639-1) is ${BASH_REMATCH[1]}."
> else
> echo "Your locale was not recognised"
> fi
```

دستور tr

دستور tr برای Translate یا Delete کردن کاراکترهای ورودی‌های استاندارد (Stdin) و نوشتن آن‌ها در خروجی‌های استاندارد (Stdout) استفاده می‌شود. دو مجموعه از کاراکترها را به‌عنوان پارامتر دریافت می‌کند و کاراکترهایی را در جمله اول جایگزین دومین جمله می‌کند یا

اینکه اگر سوئیچ d- استفاده شده باشد از کلمه اول حذف می‌کند. شکل کلی استفاده از آن به‌صورت زیر است:

```
tr  OPTIONS set1 set2
echo  STRING | tr OPTIONS set1 set2
tr  OPTIONS set1 set2 < inputfile
tr  OPTIONS set1 set2 < inputfile > outputfile
```

در مثال‌های زیر تمامی کاراکترهای a با کاراکتر A جایگزین می‌شوند. اولین خط، خروجی دستور echo به دستور tr رفته و در دومین خط فایل inputfile به‌عنوان ورودی به دستور tr رفته و تمامی کاراکترهای a با A جایگزین می‌شوند.

```
echo  “My name is Mahdi” | tr a A
tr  a A < inputfile
```

البته می‌توان از متدهایی نیز استفاده کرد. به‌طور مثال اگر بخواهیم تمامی حروف کوچک را به بزرگ تبدیل (Translate) کنیم از دستور زیر استفاده می‌کنیم. tr به بزرگی و کوچکی حروف حساس است؛ یعنی بین a و A تفاوت می‌گذارد.

```
echo "My namE iS Mahdi" | tr  a-z  A-Z
MY NAME IS MAHDI
```

البته می‌توان از الگوهای از پیش تعریف شده ای استفاده کرد. در زیر این الگوها توضیح داده شده‌اند.

[:alnum:]: معرف تمامی حروف و اعداد (all letters and digits).

در مثال زیر بجای تمامی اعداد و حروف کاراکتر a قرار می‌گیرد ولی چون @ نه عدد است نه حرف پس باقی می‌ماند.

```
echo "My name is amir and 29 years@" | tr [:alnum:] a
aa aaaa aa aaaa aaa aa aaaaa@
```

[:alpha:]: معرف تمامی حروف (الفبا). در مثال زیر بجای تمامی حروف کاراکتر a می‌نشیند.

```
echo "My name is amir and 29 years@" | tr [:alpha:] a
aa aaaa aa aaaa aaa 29 aaaaa@
```

[:blank:]: معرف فضای خالی افقی. در مثال زیر بجای تمامی فضاهای خالی (Tab or Space) کاراکتر sp_ قرار می‌گیرد.

```
echo “My name is amir and 29 years@” | tr [:alpha:]
_sp
```

[:digit:]: معرف تمامی اعداد

[:xdigit:]: معرف اعداد هگزادسیمال (مبنای ۱۶)

[:graph:]: تمامی کاراکترهای قابل چاپ، بجز فضای خالی

[:print:]: تمامی کاراکترهای قابل چاپ و فضای خالی

[:space:]: تمامی فضاهای خالی عمودی و افقی

[:lower:]: معرف تمامی حروف کوچک

[:upper:] : معرف تمامی حروف بزرگ

سوئیچ c- یا complement– که به معنی مکمل است عکس مثال‌های بالاست. برای نمونه در مثال زیر، بجای اینکه کاراکتر a جایگزین تمامی اعداد شود، بجای دیگر کاراکترها می‌آید. برای درک بهتر باید دستور زیر را اجرا کنید تا از خروجی منظور این سوئیچ را درک کنید.

```
echo "Abc123d56E" | tr -c [[:digit:]] a
aaa123a56aa
```

سوئیچ دیگر d- یا delete– است که کاراکتر set2 را از کاراکتر اول حذف می‌کند. برای مثال در خط زیر تمامی اعداد از رشته حذف می‌شوند.

```
echo “Abc123d56E” | tr -d [[:digit:]]      #DELETE
ALL DIGITS
```

یا اینکه در مثال زیر تمامی کاراکترهای a حذف خواهند شد.

```
echo “My name is amir and 29 years@” | tr -d a
```

البته می‌توان دو پارامتر را باهم استفاده کرد. دستور زیر به دلیل استفاده از سوئیچ c- بجای اینکه اعداد را حذف کند، حروف را حذف می‌کند.

```
echo “Abc123d56E” | tr -cd  [[:digit:]]    #DELETE
ALL LETTER
```

دستور بالا معادل اجرای دستور زیر است:

```
echo “Abc123d56E” | tr -d [[:alpha:]]    #DELETE
ALL LETTER
```

مثال زیر استفاده این دستور در اسکریپت نویسی است. در این مثال برای پاک کردن یک فایل از کاربر به وسیله ورود yes یا no تأییدیه می‌گیریم که آیا فایل پاک شود یا خیر. خود دستور rm با استفاده از سوئیچ i- این کار را انجام می‌دهد و به کوچکی و بزرگی yes یا no حساس نیست ولی هدف از این دستور حساس کردن به ورود yes و no کوچک است.

```
1  #!/bin/bash
2  echo -n "Enter file name : "
3  read myfile
4  echo -n "Are you sure ( yes or no ) ? "
5  read confirmation
6  confirmation="$(echo ${confirmation} | tr 'A-Z' 'a-z')"
7  if [ "$confirmation" == "yes" ]; then
8    [ -f $myfile ] &&  /bin/rm $myfile || echo "Error - file $myfile not found"
9  else
10   : # do nothing
11 fi
```

در خطوط ۲ و ۴ یک اعلان با دستور echo نشان داده می‌شود که شما باید در خط ۲ نام یا مسیر یک فایل را وارد کنید. در خط ۳ هر چه که در خط ۲ وارد کرده‌اید در یک متغیر ذخیره می‌شود. خط ۴ هم مانند خط ۲ یک اعلان نشان می‌دهد که باید یا yes یا no را وارد کنید. فرقی نمی‌کند که حروف بزرگ باشند یا نباشند. هرچه که وارد کنید در خط ۵ در یک متغیر ذخیره می‌شود و در نهایت در خط ۶ تمامی حروف yes یا no را به حروف کوچک تبدیل می‌کند. در خط ۷ بررسی می‌شود که yes (حتماً کوچک) وارد شده باشد، در خط ۸ چک می‌کند که آیا فایل وجود دارد یا خیر.

متغیرهای درونی پوسته

در تمامی سیستم‌عامل‌ها یک سری متغیرها وجود دارند که اطلاعات خاصی را به ازای هر کاربر یا به‌صورت سراسری برای تمامی کاربران نگه‌داری می‌کنند. این‌گونه متغیرها را in-build variable گویند. در این قسمت اصلی‌ترین متغیرهای درونی سیستم‌عامل لینوکس را معرفی می‌کنیم. این متغیرها در بیشتر سیستم‌عامل‌های یونیکسی مانند BSD ها (مثل FreeBSD)، سیستم‌عامل سولاریس و همچنین سیستم‌عامل مکینتاش نیز وجود دارند. این متغیرها در Shell Scripting بسیار کاربردی هستند. به‌طور مثال برای اینکه بررسی کنیم که آیا کاربر Login شده root است یا نه از متغیر $UID استفاده می‌کنیم. توجه کنید که به ازای هر کاربری که Login می‌کند مقدار این متغیرها ممکن است متفاوت باشد. به‌طور مثال مقدار این متغیرها برای کاربر root نسبت به دیگر کاربران متفاوت است. برای فهرست کردن تمامی متغیرهای محیطی (in-build) از دستور env استفاده کنید.

```
env | less
HOSTNAME=rajacent.x.com
TERM=xterm
SHELL=/bin/bash
HISTSIZE=1000
SSH_CLIENT=196.75.83.120 62161 600
SSH_TTY=/dev/pts/2
USER=root
MAIL=/var/spool/mail/root
```

PATH=/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/s
bin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin

INPUTRC=/etc/inputrc

PWD=/root

LANG=en_US.UTF-8

SHLVL=1

HOME=/root

LOGNAME=root

$SHELL: نام پوسته ای که در هنگام ایجاد یک کاربر و با استفاده از سوییچ s- از دستور userasdd به کاربر داده‌ایم را نشان می‌دهد. لیستی پوسته‌های نصب شده در سیستم‌عامل لینوکسی (یونیکسی) در فایل etc/shell/ قرار دارد.

cat /etc/shells

اگر می‌خواهید بدانید که نام پوسته (Shell) پیش‌فرضتان چیست دستور را اجرا کنید.

echo $SHELL

$HISTSIZE: در دایرکتوری خانگی هر کاربر فایلی به نام bash_history وجود دارد که لیستی از دستورهای اجراشده آن کاربر را نگه می‌دارد. متغیر $HISTSIZE تعداد دستورهایی که در این فایل نگه‌داری خواهد شد را تعیین می‌کند. مقدار پیش‌فرض آن ۱۰۰۰ است.

$HISTFILE: محل ذخیره فایل را نشان می‌دهد؛ که به‌صورت پیش‌فرض در دایرکتوری خانگی هر کاربر قرار دارد.

$USER: نام کاربری‌تان را ذخیره می‌کند.

$UID: شناسه کاربر را نشان می‌دهد. شناسه کاربر root برابر عدد ۰ و شناسه کاربران سیستمی‌مانند mysql اعدادی از ۱ تا ۱۰۰ می‌باشد. کاربران سیستمی امکان Login به سیستم را ندارند و تنها توسط سرویس‌هایشان برای کنترل، دسترسی و نوشتن در فایل‌هایشان استفاده می‌شوند. به این خاطر نمی‌توانند وارد (Login) به سیستم شوند، چون دارای nologin Shell هستند. کاربران معمولی نیز عددی مانند ۵۰۰ به بالا دارند. به‌طور مثال برای اینکه اجبار کنیم تا اسکریپتی توسط کاربر root انجام شود می‌توان از این متغیر استفاده کرد.

$GROUPS: به دست آوردن اطلاعات گروه

$PWD: حاوی دایرکتوری جاری است. دایرکتوری جاری دایرکتوری است که کاربر در حال حاضر در آن قرار دارد. دستور pwd از مقدار این متغیر استفاده می‌کند.

$HOSTNAME: نام ماشین یا hostname را نشان می‌دهد.

$HOME: نام دایرکتوری خانگی کاربر را نشان می‌دهد.

$HOSTTYPE یا $MACHTYPE: معماری ماشین یعنی ۳۲ بیتی یا ۶۴ بیتی بودن را نشان می‌دهد.

$OSTYPE: نوع سیستم‌عامل را نشان می‌دهد. به‌طور مثال اگر از لینوکس استفاده می‌کنید مقدارش Gnu Linux و یا اگر از سولاریس استفاده می‌کنید مقدارش Sun Solaris خواهد بود.

$TERM: نام Terminal را نشان می‌دهد.

$TMP یا $TMPDIR: مسیر دایرکتوری Template را نشان می‌دهد.

$PATH: مسیر دایرکتوری‌هایی که حاوی فایل‌های باینری برای کاربر هستند را نشان می‌دهد.

$PIPESTATUS: عدد وضعیت اجرای یک Pipe را نشان می‌دهد. Pipe به معنی ارسال خروجی یک دستور به ورودی دستور دیگر است. به‌طور مثال در دستور زیر ابتدا ls -l انجام می‌شود و خروجی آن به دستور wc -l ارسال می‌شود که تعداد سطرهای خروجی دستور ls -l را می‌شمارد و نشان می‌دهد. چون اجرای دستور زیر درست است، پس وضعیت خروجی آن باید عدد ۰ باشد.

```
ls -l | wc -l
echo $PIPESTATUS
0
```

اما اگر بجای دستور بالا از دستور زیر استفاده کنیم، چون از دستوری اشتباه استفاده کرده‌ایم (یک اشتباه در استفاده نادرست از سوئیچ‌ها در دستور wc)، پس عددی به‌جز صفر (۱۴۱) نشان داده می‌شود.

```
ls -l | wc -u
echo $PIPESTATUS
141
```

$PPID: شناسه والد فرایند را نشان می‌دهد.

متغیرهای PS1 و PS2 و PS3 و PS4 نیز برای سفارشی کردن ترمینال به کار می‌روند. (توجه کنید که اول نامشان $ است.)

پارامترهای زیر در اسکریپت نویسی بسیار مفید هستند.