آموزش

Shell Programming

کاری از :

محمد استادهاشمی

http://linuxnews.persianblog.com + mohammad@ostadhashemi.com

مقدمه و معرفی Shell

وقتی شما وارد لینوکس می شوید، اولین برنامه ای که اجرا می شود Shell است. در واقع Shell یک پردازشگر است که فرامین ورودی در خط فرمان را اجرا می کند.

برای شما تابحال اتفاق افتاده که بخواهیدیک یا چند کار را در محیط متنی و با استفاده از خط فرمان، هر روز یا روزی چند بار اجرا کنید. این کار مدت زیادی از وقت شما را می گیرد. اما شما می توانید چندین دستور را با استفاده از یک برنامهٔ ساده، در یک فایل متنی ذخیره کنید و با هر بار اجرای آن فایل، همهٔ دستورات مورد نظر شما اجرا شود. به این برنامه نویسی، «برنامه نویسی شل (Shell Programming)» و به آن فایل، یک « اسکریپت شل (Shell Script) » گفته می شود.

اسکریپت نویسی در لینوکس، کارها را بسیار آسان کرده و سرعت کارها را بالا می برد، پس باید ابتدا کمی بیشتر در مورد شل ها بدانیم :

انواع مختلفی از شل ها وجود دارند که معروفترین آنها عبارتند از :

bash (Bourne – Again Shell)
csh (C Shell)
tcsh
ksh (Korn Shell)

از بین شل ها، معمولا از « bash » استفاده می شود. برای اینکه مطمئن شوید که شل اجرایی روی سیستم شما چیست، دستور زیر را در خط فرمان وارد کنید : $cat/etc/Shells و برای اینکه شمارهٔ نسخهٔ bash که از آن استفاده می کنید را بفهمید، از دستور $echo $BASH_VERSION در خط فرمان استفاده کنید.

(برای اطلاعات بیشتر در مورد bash ، در خط فرمان تایپ کنید : $man bash)

شروع کار با bash shell

برای شروع بهتر است که با یک اسکریپت ساده شروع کنیم که عبارت Salam را نمایش دهد. برای این کار باید این برنامه کوچک را در یک ویرایشگر متن مثل Vi یا gEdit بنویسید و با نام salam.sh ذخیره کنید.

```
#!bin/bash/
#salam.sh
echo "Salam"
```

در مرحله بعد باید این فایل را قابل اجرا کنید یعنی Mode آن را تغییر دهید.

برای این منظور می توانید از دستور chmod استفاده کنید.

در پنجره Xtrem این دستور را تایپ کنید : $chmod +x salam.sh

توجه داشته باشید که این دستور را در همان محلی که فایل را ذخیره اید اجرا کنید.
حال این فایل قابل اجرا شده و برای اجرای این فایل هم کافیست که تایپ کنید : ./salam.sh
اگر درست عمل کرده باشید ، عبارت Salam نمایش داده خواهد شد.
حال کمی به عقب برگردیم و ببینیم که چه کرده ایم.
خط اول برنامه با # شروع شده است. این کاراکتر به شل می گوید که این سطر، یک سطر توضیحی (Comment) است و علامت تعجب (!) که بعد از آن است، می گوید که بعد از آن را اجرا کند. یعنی بعد از علامت تعجب در خط اول، شما باید محل مفسر خط فرمان خود یعنی bash را بدهید تا سیستم بداند برای اجرای بقیه دستورات، از چه نوع شل و در کجا استفاده کند.
برای فهمیدن محل bash دستور روبرو را در خط فرمان وارد کنید: whereis bash
خط دوم نیز یک سطر توضیحی است که نام و شماره نسخه و دیگر اطلاعات برنامه را برای دیگر کاربران معین می کند.
لذا تمامی کاری که انجام داده ایم مربوط به خط سوم است. (echo "Salam") واضح است که بعد از دستور echo هر عبارتی که بین دو کاراکتر (") بیاید، روی صفحه چاپ می شود. مثلا اگر نام خود را به جای Salam بنویسید، همان نام چاپ می شود.

معرفی و کار با متغیرها

bash و سایر شل ها، به عنوان یک زبان دارای تمام قابلیتها، از متغیرها پشتیبانی می کنند. یک متغیر (variable)، مکانی در حافظه است که شما به bash می گویید برای محتوایی که شما تعیین می کنید، آن جا را رزرو کند. این نوع، یک متغیر تعریف شده توسط کاربر (user-defined variable) نامیده می شود. کاربر باید تصمیم بگیرد که چه چیزی در آن قسمت از حافظه، نگه داری شود.

فرض کنید می خواهید متغیری برای ذخیره نام خودتان داشته باشید. در اغلب زبانها، این متغیر، رشته ای (String) نامیده می شود. در اینجا شما باید متغیر را نوشته، سپس مقداری را به آن نسبت دهید. مثلا بنویسید: myname=mohammad

در اینجا نام متغیر، myname و مقدار نسبت داده شده به آن، mohammad است.

```
#!/bin/bash
#name.sh
myname="mohammad"
echo $myname
```

یا اگر بخواهید، می توانید یک مقدار عددی به متغیر بدهید : count=56 (در اینجا هم، count متغیر و 56 مقدار نسبت داده شده به متغیر است.)

در bash ، از هر رشته حرفی- عددی، به عنوان نام متغیر می توان استفاده نمود. زیرا در bash و سایر شل های موجود، لازم نیست ابتدا متغیر را تعریف کرده و سپس از آن استفاده نمود. مثلا در برخی زبانهای برنامه نویسی مثل Visual Basic یا C++ باید ابتدا نوع متغیر را تعریف کرده سپس از آن استفاده کرد و لی در bash همچین کاری لازم نیست.

نکته : برای رجوع به متغیر، دو راه وجود دارد. یکی به وسیله نسبت دادن، یعنی اگر بخواهیم نام mohammad را به متغیر myname نسبت دهیم، باید اینگونه بنویسیم:

```
myname="mohammad"
```

یکی هم اینکه برای رجوع به متغیر از مقدار آن استفاده کنیم.

در bash برای گرفتن محتوای متغیر، قبل از نام متغیر از $ استفاده می شود. در این صورت به شل می گویید که می خواهید مقدار متغیر برگردانده شود. برای رجوع به محتوای متغیر باید بنویسید : $myname

حتما در لینوکس دیده اید که گاهی در جلوی یک فرمان یا فایل اجرایی از یک یا چند پارامتر استفاده می شود. این عمل را در این برنامه مشاهده خواهید کرد:

```
#!/bin/bash
#parameters.sh
echo "First para = $1"
echo "Second para = $2"
```

در این برنامه، 1 و 2 متغیر هستند که ما از شل خواسته ایم که محتوای آنها را چاپ کند یعنی اگر این برنامه را به کمک دو پارامتر اجرا کنید، (یعنی بنویسید : parameters.sh hello man) عبارت زیر را خواهید دید:

```
First para = hello
Second para = man
```

در آخر این قسمت هم، انواع متغیرها را معرفی می کنیم:

متغیر های کاربر (User Variables) : این دسته از متغیرها به وسیله کاربر تعریف می شوند و در فایل های اسکریپت، می توانند تغییر کنند.

متغیرهای محیطی (Environment Variables) : این ها بخشی از محیط سیستم هستند. لازم نیست آنها را تعریف کنید. آنها می توانند در داخل برنامه، مانند هر متغیر دیگری مورد استفاده قرار گیرند و برخی از آنها می توانند تغییر کنند مثل PATH که در صورت نیاز می توان آنرا تعریف کرد یا تغییر داد.

متغیرهای داخلی (Built-in Variables) : این متغیرها به وسیله سیستم در اختیار شما قرار می گیرند و به هیچ وجه نمی توانند تغییر کنند.

همانطور که گفتیم، متغیرهای داخلی و متغیرهای محیطی، توسط سیستم ارائه می شوند. البته بعضی از آنها را نمی توان تغییر داد ولی بعضی هم قابل تغییرند. به این متغیرها، متغیرهای سیستمی (System Variables) می گویند که در سیستم ما تعریف شده هستند. در اینجا به چندی از مهمترین متغیرهای سیستمی و مثال هایی اشاره شده است :

متغیر سیستمی	معنی (کاربرد)
BASH = /bin/bash	نام شل مورد استفاده ما
BASH_VERSION = 1.14.7(1)	شماره نسخه شل مورد استفاده ما
COLUMNS = 80	شماره ستون صفحه نمایش ما
HOME = /home/yourusername	دایرکتوری خانه شما (دایرکتوری اصلی)
LINES = 25	شماره ستون صفحه نمایش ما
LOGNAME = student	نام ورودی (loggin name)
OSTYPE = Linux	سیستم عامل شما
PS1 = [\u@\h\w]\s	تنظیمات prompt ما
PWD = /home/yourusername/temp	دایرکتوری جاری ما
SHELL = /bin/bash	نام شل مورد استفاده ما
USERNAME = yourusername	نام ورودی کسی که به pc وارد شده (user name)

شما از این متغیرها به صورت های مختلفی می توانید استفاده کنید، مثلا با این دستور می توانید هر یک از اطلاعات جدول بالا را در مورد سبستم خود بگیرید:

$echo $OSTYPE (نوع سیستم عامل شما را چاپ می کند)

$echo $HOME (دایرکتوری خانه را نشان می دهد)

نکته : برای احتیاط، از تغییر دادن این متغیرهای سیستمی خودداری کنید چون امکان دارد مشکلاتی را برای شما پیش آورد.

قواعد نامگذاری متغیرها :

۱ . نام متغیر باید با حروف الفبا شروع شود و می تواند شامل حروف، اعداد و تنها، علامت (_) باشد.

۲ . نه در نام متغیر و نه در مقدار متغیر، نباید از Space استفاده شود.

۳ . متغیرها به حروف بزرگ و کوچک حساسند. درست مثل نام فایلها و دستورات، در لینوکس.

٤ . شما می توانید متغیرهای پوچ تعریف کنید. مثلا :

$vech= (برای تعریف متغیر، ازاین دستور نیز استفاده می شود.)

$vech=""

۵ . از علائمی مثل : ؟ , * و غیره نباید در نام متغیر استفاده شود.

عملگرهای ریاضی و استفاده از دستور read

عملگر های ریاضی :

ما در bash همانند دیگر برامه نویسی ها می توانیم از عملگرهای ریاضی استفاده کنیم. به این شکل:

$expr عدد اول عملگر ریاضی عدد دوم

$expr 6+3

عملگرهای قابل استفاده در bash عبارتنداز :

جمع	+
تفریق	-
تقسیم	/
باقیمانده	%
ضرب	* (علامت ضرب *\ است، نه *)

اگر بخواهید حاصل این عمل ها را با دستور echo چاپ کنید، باید بنویسید :

echo `expr 6+3`

توجه کنید، علامتی که دو طرف دستور بعد از echo گذاشته شده است، (`) است یعنی دکمه بالای دکمه tab یا همان دکمه ~ .

این علامت در بعضی جاهای دیگر موجب اجرای دستورات می شود. مثلا اگر بنویسید :

echo "today is date"

دقیقا همین جمله چاپ خواهد شد ولی اگر بنویسید :

echo "today is `date`"

عبارت today is چاپ شده و در جلوی آن تاریخ همان روز چاپ می شود.

پایان وضعیت :

بعد از پایان هر برنامه یا اجرای یک دستور، اگر بنویسیم : $? دو حالت ممکن است. اگر عددی که سیستم به ما می دهد، صفر باشد، یعنی این دستور یا برنامه با موفقیت انجام شده. مثل پاک کردن یک فایل یا نصب یک برنامه یا ... . اما اگر عدد نمایش داده شده در جواب این دستور ($?) ، یک عدد غیر صفر بود، یعنی دستور یا برنامه قبلی با موفقیت به پایان نرسیده و عمل مورد نظر، انجام نشده است.

دستور read :

دستور read برای گرفتن اطلاعات از کاربر و ذخیره آن در یک متغیر استفاده می شود و بدین صورت استفاده می شود :

read متغیر اول , متغیر دوم , ... , متغیر آخر

به برنامه زیر توجه کنید :

```
$vi sayH
#script to read your name
echo "what is your first name?"
read fname
echo "Hello $fname, be lucky"
```

و به این شکل آن را اجرا کنید :

```
$chmod 755 sayH
$./sayH
```

این برنامه ابتدا از شما می پرسد که نام کوچک شما چیست. (شما باید پس از وارد کردن نام خود، کلید enter را بزنید.) سپس به شما سلام می کند و...

مثلا اگر نام خود را mohammad وارد کنید، سیستم پاسخ می دهد :

```
Hello mohammad, be lucky
```

پردازش خط فرمان در bash shell

این دستور را در خط فرمان اجرا کنید : $ls long_story (فرض می کنیم که فایل "long_story" موجود نباشد.) اجرای این دستور موجب می شود پیامی مثل این را ببینید :

```
long_story : No such file or directory
```

همانطور که می دانید، ls یک دستور واقعی شل است. حال باید بدانیم که وقتی این دستور را اجرا می کنیم، چه اتفاقی می افتد؟

اولین کلمه در خط فرمان (ls) همیشه نام فرمان یا برنامه ای است که اجرا می شود. هر چیز دیگر بعد از آن، به عنوان آرگومان ها یا نشانه های این فرمان در نظر گرفته می شوند و بعد از آن اجرا می شوند. مثلا : $tail +10 myf

در اینجا tail نام فرمان و 10+ و myf آرگومان های آن هستند.

بعضی از فرمان های مهم و آرگومان های آنها بدین قرار است :

فرمان	تعداد آرگومان ها ($#)	آرگومان های مربوطه
ls	1	foo
cp	2	y و y.bak
mv	2	y.bak و y.okay
tail	2	-10 و myf
mail	1	raj
sort	3	-r و -n و myf
date	0	
clear	0	

توجه : علامت $# تعداد آرگومان های تعیین شده در خط فرمان را در خود نگه می دارد و علامت های $@ , $* نیز به همه آرگومان های درست استفاده شده (تست شده) برمیگردد.

این خط فرمان را در نظر بگیرید : $myshell foo bar در اینجا نام شل اسکریپت شما "myshell" ، اولین آرگومان خط فرمان، foo و دومین آرگومان bar است.

در shell، اگر بخواهیم به آرگومان های خط فرمان یا دستورات مراجعه کنیم، این گونه عمل می کنیم :

Myshell → $0
foo → $1
bar → $2

منظور از $* همین ($0 , $1 , $2 , ...) است که در بالا گفته شد.

برای مثال برنامه زیر را بنویسید و چگونگی دسترسی به آرگومان ها را مشاهده کنید :

```
$ vi demo
#!/bin/sh
# command line args
echo "Total number of command line argument are $#"
echo "$0 is script name"
echo "$1 is first argument"
echo "$2 is second argument"
echo "All of them are :-  $*  or $@"
```

mode این برنامه را اینگونه تغییر دهید : $chmod 755 demo

و آنرا اجرا کنید : $./demo Hello World

این اسکریپت را در دایرکتوری bin در سیستم خود ذخیره و نگه داری کنید چون این اسکریپت بعدا لازم خواهد شد.

جملات شرطی و حلقه ها

Bash امکان استفاده از جملات شرطی و حلقه ها را به ما می دهد که توسط آنها می توان مسیر اجرای برنامه را تغییر داد. برخی از آنها را در اینجا بررسی می کنیم:

دستور if :

if …. then …. else …. fi

این جمله، یک شرط را محاسبه خواهد نمود تا ببیند که آیا آن درست است یا غلط. اگر شرط صحیح نبود، کار دیگری را انجام خواهد داد.

اغلب شل ها به شما اجازه می دهند از جملات شرطی استفاده نمایید که به قدرت محیط برنامه نویسی می افزاید. جملات شرطی اجازه می دهند تا تستهای شرطی کامل را بتوان در برنامه شل انجام داد. ترکیب ابتدایی جمله if به این شکل است :

```
if [شرط]
    then [دستور۱]
elif [شرط]
    then [دستور۲]
else [دستور۳]
fi [finished]
```

bash از فرمانی به نام test برای کنترل محتوای متغیرها استفاده می کند. از test برای ارزیابی عبارات و بازگرداندن درست (true) یا غلط (false) ، بسته به پارامترهایی که به آن فرستاده مـی شـود، در اسکریپت ها می توان استفاده نمود.

فرض کنید می خواهید برنامه ای بنویسید که ببیند آیا نام mohammad در متغیری بـه نـام $name ذخیره شده است یا خیـر و در صـورتی کـه محتـوای مـورد نظـر را نداشـت، عبـارت "you're not mohammad" را چاپ کرده و خارج شود.

```
#! /bin/bash
# if then else loop to test value of $name variable.
if test"$name"="mohammad"
      then
echo "hi mohammad"
      else
echo "you're not mohammad"
fi
```

این برنامه کوتاه، طرز استفاده از دستور if … then … else را به شما نشان می دهد. البته این فقط یک مثال ساده است. با استفاده از این دستورات، کارهای بسیار جالبی می توان انجام داد.

دستور while :

این دستور برای تکرار بخشی از برنامه تا زمانی که شرط خاصی برقرار باشد، مورد استفاده قرار می گیرد. فرض کنید که قصد داریدجلوی کاربر را در مقابل وارد کردن کلمه "superuser" بگیرید. یک راه این است که اگر کسی کلمه را وارد کرد، از او خواسته شود چیز دیگری وارد کند.

ترکیب دستور while به این شکل است : while … do … done

برنامه مثالی که در بالا گفته شد.به این صورت نوشته می شود :

```
#! /bin/bash
# chck login name
echo "enter your name :"
logname="superuser"
while test $logname="superuser"
do
echo –n "Please Enter Another Login Name."
read logname
done
```

البته این فایل اسکریپت، هنگامی باید اجرا شود که یک کاربر بخواهد به root دسترسی پیدا کند.

این فقط یک مثال بود که با این حلقه نوشته شده بود. حلقه های شرطی دیگری نیز وجود دارنـد کـه می توان از آنها استفاده کرد اما پرکاربردترین آنها همین بود که در موردشان توضیح داده شد.

حلقه ها در bash

در ادامه کار با دستورات if … then … fi و حلقه ها ، حال به طور مختصر و سریع چند مورد از دستورات bash که مربوط به حلقه ها می شود را شرح می دهیم :

۱ . اگر بخواهیم از چندین if به صورت تودرتو استفاده کنیم، به صورت زیر عمل می کنیم :

```
if شرط
then
        if شرط
          then
             دستورات
            .....
           else
             دستورات
            .....
      fi
else
      دستورات
fi
```

۲ . **loop ها در شل اسکریپت** : loop به گروهی از دستورات گفته می شود که در یک حلقه، بارها و بارها اجرا می شود تا وقتی که شرط مورد نظر ما برقرار باشد.

Bash دارای دو نوع loop است : for loop (A while loop (B

A) for loop : این دستور به این شکل استفاده می شود :

```
for      نام متغیر      in      لیست (مجموعه)
do
در این قسمت باید دستورات، نوشته شود که این دستورات روی تک تک
اعضای لیست، اجرا می شود تا وقتی که اعضای لیست یا مجموعه به پایان برسد
done
```

B) while loop : این دستور به این شکل استفاده می شود :

```
while      شرط
do
      دستور ۱
      دستور ۲
      دستور ۳
     ....
     ...
done
```

و تا وقتی که شرط مورد نظر ما درست باشد، این حلقه تکرار می شود.

۳ . **دستور case** : این دستور هم یکی از نمونه های دستورات متناوب است که شما را قادر می سازد از مقادیر مختلفی برای یک متغیر استفاده کنید. استفاده از این دستور، از استفاده از read و write بسیار آسان تر و به صرفه تر است.

دستور case بدین شکل استفاده می شود : (command به معنای فرمان است)

```
case      نام متغیر      in
الگوی ۱ )      Command
                    ....
                    ....
                        Command ;;
الگوی ۲ )      Command
                    ....
                    ....
                        Command ;;
الگوی ۳ )      Command
                    ....
                    ....
                        Command ;;
      ..
      ..
      ..
esac
```

اشکال زدایی شل اسکریپت (de-bug)

در درسهای پیش، شما با برنامه نویسی شل (Shell Programming) به طور مقدماتی و ابتدایی آشنا شدید. گاهی اوقات برنامه نویسی شل، دچار مشکلاتی می شود (error) که شما باید این اشکالات را یافته و درست کنید.

بدین منظور می توانید از گزینه های v– یا x– به همراه دستورات sh یا bash استفاده کنید.

نحوه کلی استفاده از این دستور بدین شکل است :

sh نوع گزینه نام شل اسکریپت

یا

bash نوع گزینه نام شل اسکریپت

اگر برای اشکال زدایی، از گزینه x– استفاده کنید، این گزینه برنامه را دوباره چاپ کرده و مقادیر متغیرها را هم می نویسد و تا خروجی پیش می رود.

مثال) برنامه زیر را بنویسید.

```
$ cat > debug.sh
#!/bin/bash
#
# Script to show debug of shell
#
tot=`expr $1 + $2`
echo $tot
```

و آن را اینگونه با ورودی ٤ و ٥ ،اجرا کنید :

```
$ chmod 755 debug.sh
$ ./debug.sh 4 5
```

در این صورت، خروجی شما برابر ۹ خواهد بود. اگر آن را با گزینه x– اشکال زدایی کنید، با این خروجی ها مواجه می شوید :

```
$ sh -x dsh1.sh 4 5
#
# Script to show debug of shell
#
tot=`expr $1 + $2`
expr $1 + $2
++ expr 4 + 5
+ tot=9
echo $tot
+ echo 9
9
```

از گزینه v– نیز برای debug کردن اسکریپتهای مرکب و پیچیده استفاده می شود.

((مثالهایی از شل اسکریپت))

این برنامه زمان را با حساب am و pm به ما نشان می دهد.

```
# script to display time in am/pm format using "expr" command.
# we are using the back quote characters (`) to capture the output
# of the date command and assign that output to shell variables.

hours=`date +'%k'`
mins=`date +'%M'`
if [ $hours -eq 0 ]
then
   hours=12
fi
if [ $hours -gt 12 ]
then
   hours=`expr $hours - 12`
   ampm=pm
else
   ampm=am
fi
echo $hours:$mins $ampm
```

این برنامه کار قبلی را انجام میدهد ولی با استفاده از دستور case .

```
# script to display time in am/pm format using "case" statement

hours=`date +'%k'`
mins=`date +'%M'`
if [ $hours -ge 12 ]
then
   case $hours in
   13) hours=1 ;;
   14) hours=2 ;;
   15) hours=3 ;;
   16) hours=4 ;;
   17) hours=5 ;;
   18) hours=6 ;;
   19) hours=7 ;;
   20) hours=8 ;;
   21) hours=9 ;;
   22) hours=10 ;;
   23) hours=11 ;;
   esac
   ampm=pm
else
   case $hours in
   0 ) hours=12 ;;
   esac
   ampm=am
fi
echo $hours:$mins $ampm
```

برنامه ای که مشخص می کند، عدد ورودی، مثبت است یا منفی.

```
$ vi   isnump_n
#!/bin/sh
#
# Script to see whether argument is positive or negative
#
if [ $# -eq 0 ]
then
    echo "$0 : You must give/supply one integers"
    exit 1
fi


if test $1 -gt 0
then
    echo "$1 number is positive"
else
    echo "$1 number is negative"
fi
```

برنامه ای که با if…then…fi و case نوشته شده و اطلاعاتی را در مورد وسایل نقلیه میدهد.

```
$ cat > car
#
# if no vehicle name is given
# i.e. -z $1 is defined and it is NULL
#
# if no command line arg

if [ -z $1 ]
then
    rental="*** Unknown vehicle ***"
elif [ -n $1 ]
then
# otherwise make first arg as rental
    rental=$1
fi

case $rental in
    "car") echo "For $rental Rs.20 per k/m";;
    "van") echo "For $rental Rs.10 per k/m";;
    "jeep") echo "For $rental Rs.5 per k/m";;
    "bicycle") echo "For $rental 20 paisa per k/m";;
    *) echo "Sorry, I can not gat a $rental for you";;
esac
```