Bash Shell Script Part 2

How to create and run a script in Linux?

This is an introduction to Scripts in Linux

This is the second part of a series of 3 lessons. Make sure you go through them all successively to better understand the concept.

Let's get started!

Table of content

1 Environment variables

How are they different from normal variables?

Environment variables

- There are some variables that are already set in the system for each user: **Environment variables**
- When they are called, the **system prints their current value depending on the logged in user.**
- You can run the command **# env** or **# printenv** in the Terminal to check those variables

Example: Run the command **# env** and let's check some important environment variables we will use frequently

Environment variables

Variable name	Description
HOME	The user's Home directory
PWD	The present working directory
SHELL	The shell of the user
HOSTNAME	The Hostname
USER	The current username
LOGNAME	The logged in user

Environment variables

Let's implement that in the **variable.sh** script:

 # vim variables.sh

```
#!/bin/bash
# Description:
# Author:
# Date:
echo "Hello there, your username is ${USER}"
echo"Your User ID is ${UID}"
echo "Your Shell is ${SHELL} "
echo "Your home directory is ${HOME}"
echo "This server's Hostname is  ${HOSTNAME}"
```

Save and Quit
Run the script

Environment variables

```
echo " Hello there your username is ${USER}"
echo " Your uid is  ${UID} "
echo " Your shell is ${SHELL} "
echo " Your home directory is ${HOME} "
echo " this serve's hostname is ${HOSTNAME} "
```

Save and Quit
Run the script

```
[root@puppetagent ~]# ./variables.sh
 Hello there your username is root
 Your uid is  0
 Your shell is /bin/bash
 Your home directory is /root
 this serve's hostname_is puppetagent
```

Environment variables

- The **environment variables** are not that different from the **regular variables** we used earlier in the previous lesson.
- Just that, **you don’t need to declare them, you can call them anywhere in your script**
- The **system already knows them and has their various values stored at any time**

Environment variables

Let's try to **run this script as a regular user**. (we were running as the root since)

- To do that, we first of all need to **copy the script to a directory that can be accessible to any user on the system**. Let's take the **/tmp** directory: **# cp /root/variable.sh /tmp**
- Now, let's check a **regular user** on the system and **login** to that user's account
- **# tail -10 /etc/passwd** (here I picked the user **serge**)

```
natasha:x:1002:1002::/home/natasha:/bin/bash
harry:x:1003:1003::/home/harry:/bin/bash
serge:x:1004:1004::/home/serge:/bin/bash
```

Environment variables

- **# su - serge** (eventually enter the password)
- Now run the script with: **/tmp/variables.sh**

```
[serge@puppetagent ~]$ /tmp/variables.sh
 Hello there your username is serge
 Your uid is  1004
 Your shell is /bin/bash
 Your home directory is /home/serge
 this serve's hostname is puppetagent
```

Environment variables

- You can use the environment variable **to restrict access to a script**
- That is, **you can put a condition to define which user can run it**, with an if statement inside the script

Let's use or **pkg.sh** script:

- Remember this script was used to **install some packages** on the system.
- Now we want **only the root to be able to run that script successfully** (since we know only the root can run the **yum** command)

Environment variables

Switch back to the root user (**# su - root**) and modify the **pkg.sh** script

 # vim pkg.sh then go to the INSERT mode

```
#!/bin/bash

# Description: Script to install some packages
# Author: serge
# Date: May 2020

yum install finger -y
yum install curl -y
yum install zip -y
yum install vim -y
```

Environment variables

Now, let's put our condition:

```
#!/bin/bash

# Description: Script to install some packages
# Author: serge
# Date: May 2020

If [ ${USER} != root ]
then
echo "you need root access to run this"
exit 1
fi
```

```
yum install finger -y
yum install curl -y
yum install zip -y
yum install vim -y
```

Here, we need to **exit** if the condition is not satisfied. If not the script will continue its execution.

You give an exit code that is different from 0, for the system to know that the **script did not execute successfully till the end**. (**Example: exit 99**)

Let's check how the script will run for the **root user** and for **a regular user**

Environment variables

- Open a **new terminal in which you will connect as a regular user** (I will still use user **serge** here)

  ssh serge@192.168.1.141 (this is my host's IP address)

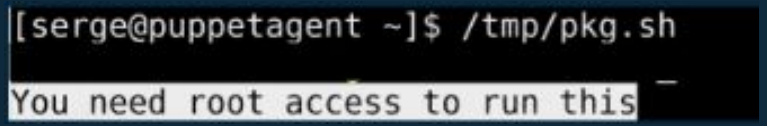
- In **Terminal 1**, we are going to run the script as the **root** user
- In **Terminal 2**, we are going to run the script as a regular user (**serge**)
- Here we are trying to check if our **if statement is working perfectly**
- For any user to access the script, let's copy it in the **/tmp** directory with: **# cp pkg.sh /tmp**

Environment variables

- In **Terminal 1**, make sure you are the root user:
  - **# echo $USER** (If it does not display root then you need to switch to the root user: **# su - root** then **# id** to check)
  - Now run the script: **./pkg.sh**
- **The script will run successfully till the end**
- Another condition that could be used is **if [${UID} -ne 0] since we know only the root has the UID = 0**

Environment variables

- In **Terminal 2**, we are going to run the script as a regular user (**serge**)
  - **# echo $USER** (make sure it displays your username (**serge for me)**
  - Now run the script: **/tmp/pkg.sh**
- **The script will exit**



Create an environment variable

How can we create an environment variable?

Environment variables

- To create an environment variable, we use:

  # export VarNAme=Value

Example: Let's create an environment variable called **TMPDIR** that will carry the **/tmp** directory path.

 # export TMPDIR=/tmp

- Check with **# env** and you will see **a new variable created**
- Now, even **if the system is rebooted, the variable will still be there.**
- To **unset** it, you can use: **# unset TMPDIR**
- Check back in **# env**: It is no more there!

2

The Read statement

What is it used for?

The read statement

- When you need some informations from a user in order to run a script, you need to use a **read statement**
- This statement is **used to take input from the user**

Example: Let's write a script that takes as input the **name, the birth year, the city, the reason why the user came to the store** and **displays those informations.**

Solution: to solve this we need to **create a script** and use some variables that will contain the various informations:

na for the name,

y for the year of birth

CITY for the city

R for the reason

The read statement

 # vim read.sh

```
#!/bin/bash

# Description
# Author
# Date

echo “What is your name? ”
read na
echo “What year were you born? ”
read y
echo “What city are you from?
read CITY
echo “What brought you to the store today? ”
read R
```

```
echo “Hello ${na}, you were born in ${y}, you live in ${CITY} and the reason for coming here is : ${R}  ”
```

- **Save and Quit**
- Give the execute permission on the script: **# chmod +x read.sh**
- Run the script: **./read.sh**
  - na: **serge**
  - y: **1950**
  - CITY: **Bafoussam**
  - R: **I am running out of water**

The “” error is a common mistake with the **echo command**.

If you encounter this error, just go back to the script and check where the “ is missing!

When you run a script and it throws and error, just read the line where the error appears and try to fix it!

The read statement

You can see the script ran successfully and displayed the informations as expected

```
[root@puppetagent ~]# ./read.sh
what is your name?
serge
what year were you born?
1920
what city are you from?
bafoussam
what brought you to the store today ?
I am running out of water
hello serge you were born in 1920, you live in bafoussam and the reason for coming here is : I am running
out of water
```

The read statement

- We can **add some conditions on variables in there**.
- For example, let's say **if a user skips the name** (ie he does not enter his name) we display a message that **asks him to enter a valid name**
- You can do that by adding the **if statement**:

```
if [ -z ${na} ]
then
echo "Please enter a valid name"
exit 2
fi
```

- You can do the same for all the variables. You can even **nest the if statements** using **elif (ie else if)**

The read statement

You can also link many conditions in the **if statement** using:

- **||** for **or**

  if [condition 1] || [condition 2] || [...] || [condition n]

- **&&** for **and**

  if [condition 1] && [condition 2] && [...] && [condition n]

- You can even mix **||** and **&&**

Play around with the **conditions on the various variables**

Make more research and practice on this concept.

See you guys in the next part!

Thanks!

Any questions?

You can find us at:

website: http://utrains.org/

Phone: +1 (302) 689 3440

Email: contact@unixtrainings.com

Click on the link below to contact the support team for any issue!

utrains.org/support/

Create a ticket for your problem and we will get back to you soon!