Linux

mthree

Introduction to Linux

Module 4: Working with Variables

ALUMNI
BY MTHREE CONSULTING

Introduction to Linux

mthree

Variables will make up a big part of your Linux interactions when you start scripting.

In this module we will start to learn about variables and some other important things relating to file systems and permissioning within the Linux environment.

Files and Variables

mthree

MY_AGE_VARIABLE=42

MY_AGE_VARIABLE=“With great power comes great responsibility“

echo $MY_AGE_VARIABLE - Print the value of the MYVAR variable

Quiz: What will the value of ANOTHER_VAR be?

ANOTHER_VAR=Bananas_$ANOTHER_VAR - Use the current value of ANOTHER_VAR to reassign a new value to it

filenameVariable=~/.bashrc - Create a variable called *filenameVariable* give it some text. The text happens to be the location of a file.

cat $filenameVariable - Print the contents of the file specified in *filenameVariable*

fileContentsVariable=`cat $filenameVariable`- Use the *filenameVariable* to read a file from disk and store its contents in a variable called *fileContentsVariable*

echo $fileContentsVariable - Print the contents of the variable to the screen

Commands in Detail

mthree

On the previous slide we saw the introduction of a couple of new concepts.

Command	Comments
echo hello world	This will simply print hello world to the screen
echo “hello world”	This will also print hello world to the screen
echo MY_VARIABLE	This will simply echo out “MY_VARIABLE” to the screen
echo $MY_VARIABLE	The addition of the $ asks bash to return the value of the variable and that value will be printed out
` `	Use back ticks when you wish to execute a command and use its return value for something else – for example assigning to a variable.
MY_BASH=`cat ~/.bashrc`	This example shows that you want to use the output from the cat command to assign to the variable MY_BASH

Environment – further details

mthree

It is important to understand how to get access to details of your environment.

Command	Comments
echo $USER	This will print out the current user you are logged in as
logname	This will show the user that was used to log in – this can be different from the current user – we will go through how on the next slide
ME=logname	Creating this variable ME will assign it the value “logname” as you are not specifying here to actually run the command (test by running echo $ME)
ME=`logname`	This will now assign the variable ME the value of running the command logname
PREVUSER=$(logname)	This will assign the variable PREVUSER the value of running logname (alternate syntax)

Environment continued

mthree

Let's look at how the output changes depending on how we run a command. For these examples, CMD=logname

echo '$CMD'	Single quotes: this will simply echo $CMD to the screen
echo `$CMD`	Back ticks: this will return the value of logname, e.g., ec2-user
echo "$CMD"	This will return the value logname – but not actually run the command
echo $CMD	This will return what the value of CMD is
echo $($CMD)	This is equivalent to using the ` ` and will return the value of running logname, e.g., ec2-user

Environment – $PATH and env

The following commands will also be useful to you when trying to understand your environment setup in more detail.

Command	Comments
env	Running this command will show you all the variables that are set as part of your environment when you login
env LOGPATH=/var/log	As well as being able to see the default settings here, you can also use the env command to update any of these variables as necessary for a program to run If you run env again after this, you will see the variable LOGPATH set
env –u LOGPATH	This will unset the variable LOGPATH
$PATH	This is a variable set in env that tells Linux exactly where to search for executable files (ready to run commands), e.g., PATH=/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/opt/aws/bin:/home/ec2-user/.local/bin:/home/ec2-user/bin
export PATH=$PATH:/execlocation/	If you have written your own executable and you want to be able to call it without having the fully qualify its path, you can simply add to your $PATH using export

Sudo and Su commands

mthree

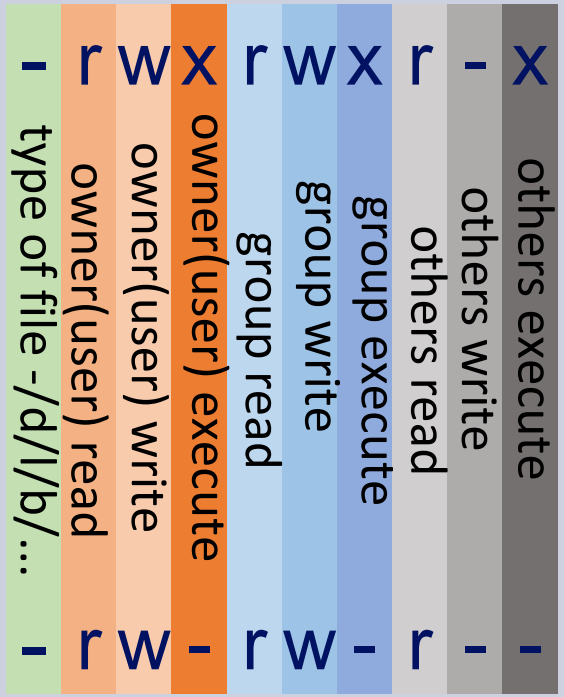
In Linux environments you will have different user accounts setup. You may have a personal account on a host as well as permission to run as an application account or maybe even root. Linux provides an easy way for you to switch between users.

Command	Comments
su **su appaccount**	Short for Switch User – this command will allow (if you are permissioned in the system) to switch between user accounts. The user will be prompted for the password of the user they are switching to. This can be common when needing to switch from a readonly user to a user with write access during a production issue
exit	To leave the account you are currently running as, you can simply type exit
sudo command	This allows you to run programs with the security privileges of another account (as a super user by default). You will be prompted for your personal password and then the OS will check the sudoers file to see if you are permissioned to run that package.

In a production environment there will be different levels of setup you will have a certain level of access, where depending on your role.

It may be that you have sudo permissions to run certain commands as needed.

File Permissions




```
[ec2-user@ip-172-31-40-21 ~]$ ls -ltr
total 76
-rw-rw-r-- 1 ec2-user ec2-user   842 Apr 23 18:36 ~
-rwxrwxr-x 1 ec2-user ec2-user  5945 Apr 24 19:53 fixGenerator.sh.good
-rwxrwxrwx 1 ec2-user ec2-user  6234 Apr 24 20:22 fixGenerator.sh
-rw-rw-r-- 1 ec2-user ec2-user 50051 Apr 24 20:26 fixlog20200424202259.log
```

```
rwxrwxr-x
775
rwxr-xr-x
755
```

```
rw-rw-r--
664
rw-r--r--
644
```

#	Permission	rwx
7	read, write and execute	rwx
6	read and write	rw-
5	read and execute	r-x
4	read only	r--
3	write and execute	-wx
2	write only	-w-
1	execute only	--x
0	none	---

Permissions Continued

Let's look at some of this permissioning in more detail

Command	Comments
touch ro	This will create a file called ro
chmod 444 ro	This will change the permissions of this file to be read only
echo avengers>ro	Here the > will try and write avengers to the ro file and get an error due to permissioning
chmod +w ro	As well as using numbers, you can use the letter notation to update the file to have write permissions – this did not give others write permission
Chmod o+w ro	This will also allow others to write to the file
chmod og-w ro	This will not allow others or group to write to the file
chmod u=rwx,og=r ro	This will give the user full access, but the group only read
chown alice file1 dir1	This will allow you to change ownership of a file/directory – this example simply transfers the ownership of both file1 and dir1 to alice.