Linux

mthree

Introduction to Linux

Module 11: Some Useful Scripting tools

ALUMNI
BY MTHREE CONSULTING

Introduction to Linux

In this module we will be looking at some tools to help us with scripting.

Basic for loops

A common tool in any programming language is the ability to be able to write a for loop. This allows you to do an action repeatedly over some kind of constraint.

Command	Comments
for i in {1..20}; do echo $i; done	Run loop for every number between 1 and 20, print each one
for i in {1..20}; do echo $i; done | tail -n 4	Only show the last 4
for i in {1..20}; do echo $i; done | tail -n +4	Show everything except the first 3
for flag in a s n r v m p i o; do echo uname -$flag: \`uname -$flag\`; done	This will print out each value for the uname command on a separate line

The if command

The if command is something you would want to run within a script than on the command line – but here is an example:

```
[ec2-user@ip-172-31-40-21 ~]$ i=2
[ec2-user@ip-172-31-40-21 ~]$ if(( $i==1 ))
> then
> echo "i is equal to 1"
> elif(($i==2))
> then
> echo "i is equal to 2"
> else
> echo "i is neither 1 or 2"
> fi
i is equal to 2
```

As you can see our basic format here is:

If commands;

Then commands;

[elif commands; then commands]

….

[else commands]

Terminal

The use of $$ always refers to the current process.

`ls /proc/$$/fd/ -l`	The use of the $$ here means you want to show file descriptors for the current process
`echo spiderman>/proc/$$/fd/1`	This will echo spiderman out to standard out of the current process
`Ps –ef $$`	Show the process info for our current session

std out and std error

When we output something to the screen, we use the default standard output (stdout).

There is also the concept of standard error (stderr), which is where the program sends error messages (also commonly goes to the terminal).

In Unix, everything is treated as a file: a file descriptor is nothing more than a positive integer that represents an open file – this means we have file descriptors for stdout and stderr.

It is always 1 for stdout and 2 for stderr.

Lets look at some examples.

stdout and stderr continued

Remember 1 is for stdout and 2 is for stderr.

Cat file.txt 1> file.log	This directs the stdout to the log file specified (command > is just a shortcut for 1>)
Cat file.txt 2> error.log	This directs the error to the error log
Cat file.txt > file.log 2>&1	Here we redirect the stderr to the same place we are redirecting stdout