Linux

mthree

Introduction to Linux

Module 9: A deeper dive into processes

ALUMNI
BY MTHREE CONSULTING

Introduction to Linux

In this module, we will look at some more advanced Linux commands with respect to processes and how you run them.

Running commands

So far, all the commands that we have run have been running in the foreground of our terminal – we cannot run any other commands while they execute. Linux offers the tools to run things in the background where needed, freeing up the terminal for you to do other things.

In Linux, a job is defined as a task that has started running and not yet completed. Each task is basically programs in execution which is a process. Each job is assigned a unique job number (Job ID).

Command	Comments
jobs	This will show jobs currently running
sleep 500	This will send the command prompt to sleep for 500 seconds, press ctrl Z
jobs	Now you should see the terminated job
bg %1	This will send the terminated job to run in the background
jobs	You should now see the job running
fg %1	This will bring the job back into the foreground

Running commands continued

Here are some further examples of things we can do while running commands.

man cat &	Putting the & at the end of the command will automatically send the process to run in the background
sleep 20 && echo second &	This will wait 20 seconds in the background before printing second to the screen – here we introduce the && command
sleep 10; echo hello	We can also list commands separated by ; on the same line. Once one command completes, the next runs
sleep no; echo hello	As no is not a unit of time, the sleep command fails but the echo command is still then executed
sleep no && echo hello	In this example, the echo command will not run as && requires the first command to exit successfully
sleep no || echo hello	This will execute the command on the right only if the command on the left returns an error

Process info

The below commands go through how to show some of the process info for things we have running in the background

Command	Comments
sleep 360 &	Run a sleep command in the background
Echo $!	Shows the process id of the last child process
ls /proc	Show the contents of the proc folder
ls /proc/$!	Show the process info for our sleep process
ls /proc/$!/fd	Show the file descriptors for our process
ls -l /proc/$!/fd	Show that stdin(0),stdout(1),stderr(2) are shortcuts to the current terminal

The watch command

The watch command is used to run any arbitrary command at regular intervals and display the output of the command on the terminal window.

This can be useful when you want to monitor things like uptime or disk usage over time.

watch date	This will watch the date command update every 2 seconds (this is the default)
watch –n 5 df -h	Here is a good example of watching disk space change every 5 seconds (-n allows you to specify the time interval)
watch –n 5 –d date	The –d option will get watch to highlight what is changing for each update (notice you cannot combine –dn here)
watch –d=cumulative date	This will make the highlighted changes sticky – any value that has ever changed will stay highlighted
watch 'CMD_1 | CMD_2'	To execute a command that continues pipe you need to enclose it in single or double quotes

nohup command

mthree

nohup will execute another program specified and ignore all SIGHUP (hangup) signals. SIGHUP is a signal sent to a process when its controlling terminal is closed.

This is useful if you are running commands over ssh and your connection drops: the process you are executing will not stop.

Command	Comments
nohup date	This is not a great example as date is a short running command, but this is the syntax to use the command
nohup mycommand > mycommand.out 2>&1 &	This redirects the standard output and standard error to the mycommand.out (and run in the background)
nohup mycommand > mycommand.out 2> mycommand.err &	This redirects standard out and standard error to different files (and runs in the background)