Linux Academy

Hands-on Lab

Change Runlevels and Boot Targets on a Sysvinit System

Contents

Lab Connection Information

- Labs may take up to five minutes to build
- The IP address of your server is located on the Hands-on Lab page
- Username: linuxacademy
- Password: 123456
- Root Password: 123456

Related Courses

LPIC-1 System Administratior - Exam 101

Related Videos

Booting the System - Sysvinit

Change Runlevels/ Boot Targets and Shutdown or Reboot the System

Need Help?

Linux Academy Community

... and you can always send in a support ticket on our website to talk to an instructor!

In this lab, we review system runlevels and their relationship with startup scripts, learn to change and set default runlevels, and explore various shutdown options.

Boot Scripts and Runlevels

Log in to the server using the credentials provided on the Hands-on Lab page.

Init scripts on CentOS 6 are located in the `/etc/rc.d` directory. The `/etc/rc.d/init.d` directory, in particular, hosts scripts for any service we can start or stop, whereas the `/etc/rc.d/rc<NUMBER>.d` directories contains the scripts that start at the defined runlevel. For example, since our server boots to runlevel 3:

```
[linuxacademy@ip] runlevel
```

All of the scripts in the `/etc/rc.d/rc3.d` directory have been run. Should we view the content of this directory, we can see how these are used:

```
[linuxacademy@ip] ls -l /etc/rc.d/rc3.d
1 root root 18 Jun  3  2016 K01yum-cron -> ../init.d/yum-cron
1 root root 19 Jun  3  2016 K10saslauthd -> ../init.d/saslauthd
1 root root 15 Mar  7 15:50 K50kdump -> ../init.d/kdump
1 root root 18 Mar  7 04:19 K61nfs-rdma -> ../init.d/nfs-rdma
1 root root 17 Jun  3  2016 K75ntpdate -> ../init.d/ntpdate
1 root root 19 Mar  7 04:23 K85mdmonitor -> ../init.d/mdmonitor
1 root root 15 Jun  3  2016 K86cgred -> ../init.d/cgred
1 root root 21 Jun  3  2016 K87restorecond -> ../init.d/restorecond
1 root root 16 Mar  7 04:22 K88auditd -> ../init.d/auditd
1 root root 20 Jun  3  2016 K89netconsole -> ../init.d/netconsole
1 root root 15 Jun  3  2016 K89rdisc -> ../init.d/rdisc
1 root root 19 Mar  7 04:23 K92ip6tables -> ../init.d/ip6tables
1 root root 18 Mar  7 04:23 K92iptables -> ../init.d/iptables
1 root root 14 Jun  3  2016 S55sshd -> ../init.d/sshd
1 root root 14 Jun  3  2016 S58ntpd -> ../init.d/ntpd
1 root root 17 Jun  3  2016 S80postfix -> ../init.d/postfix
1 root root 15 Jun  3  2016 S90crond -> ../init.d/crond
1 root root 11 Mar  7 04:17 S99local -> ../rc.local
```

Here we can see that each script is mapped to a service script in the `init.d` directory. The `S` or `K` in front of the script name itself defines whether the service is using a start up script (S) or a kill script (K), while the numbers that come afterward indicate the order the scripts are run in at start. The scripts in the `init.d` directory themselves contain the code needed to start, stop, restart, or perform any other necessary actions.

These runlevels can be further customized by adding or altering scripts in `init.d` and adding a link in the appropriate runlevel directory. Note that `rc.local` should always be the last script in the directory.

Changing Boot Targets and Runlevels

Changing Runlevels

To view the current runlevel of a system use the `runlevel` command:

```
[linuxacademy@ip] runlevel
N 3
```

This output the previous runlevel (N) and the current runlevel (3), with the `N` denoting that there are no previous runlevel because the current runlevel is the one the system booted into. Should be change the runlevel once using the system, a number would be in the `N` spot.

We can change the runlevel using the `init` command or with the legacy command `telinit`:

```
[linuxacademy@ip] init 5
```

This would start X Windows if our server had a graphical environment. Since ours does not, nothing appears to happen; however, we are now using a different runlevel regardless:

```
[linuxacademy@ip] runlevel
3 5
```

As we can see, the output indicates that the previous runlevel was 3 and the current is now 5.

Change the runlevel back:

```
[linuxacademy@ip] init 3
```

Setting Default Runlevel

View the `/etc/inittab`:

```
[linuxacademy@ip] cat /etc/inittab
# inittab is only used by upstart for the default runlevel.
#
# ADDING OTHER CONFIGURATION HERE WILL HAVE NO EFFECT ON YOUR SYSTEM.
#
# System initialization is started by /etc/init/rcS.conf
#
# Individual runlevels are started by /etc/init/rc.conf
#
# Ctrl-Alt-Delete is handled by /etc/init/control-alt-delete.conf
#
```

```
# Terminal gettys are handled by /etc/init/tty.conf and /etc/init/
serial.conf,
# with configuration in /etc/sysconfig/init.
#
# For information on how to write upstart event handlers, or how
# upstart works, see init(5), init(8), and initctl(8).
#
# Default runlevel. The runlevels used are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have
networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
id:3:initdefault:
```

The last line, `id:3:initdefault`, defines our default runlevel, which is 3. Note that we do not want to change the value of this to `0` or `6` because then we would be stuck in a reboot or shutdown loop.

Shutting Down the System

Note: Do not shut down the lab server at any point; this will end the lab.

To reboot the system, we can use the `reboot` command:

```
[linuxacademy@ip] reboot
```

Meanwhile, the `shutdown` command is capable of shutting down, rebooting, halting, and powering off a system, or broadcasting a message. We can also use this command to schedule a shutdown in the future. For example:

```
[linuxacademy@ip] shutdown -r
```

Would shutdown the system immediately. Similarly, `shutdown -h` would immediately halt the system and `shutdown -p` would immediately power off the system, with the different between halting and powering off being that a halt will bring the system to the point where it no longer responds but does not fully power off while the power off command does.

Now, if we wanted to shut down the system in 30 minutes, we could use:

```
[linuxacademy@ip] shutdown -r 30
```

This will broadcast a message to all logged in users using the `wall` command, warning them about an impending shutdown.

We can also fake a shutdown and use it to warn users of an impending shutdown with the `-k` flag:

```
[linuxacademy@ip] shutdown -r 30 -k "YOU BETTER GET OFF NOW"
```

This would broadcast the message "YOU BETTER GET OFF NOW" to all logged in users. However, this will not bring a system down -- only prevent any more users from logging in.

Should we decide to cancel a shutdown, we can use:

```
[linuxacademy@ip] shutdown -c
```

Finally, the `halt` command halts a server. This is an older method of stopping the server:

```
[linuxacademy@ip] halt
```

A `halt -p` triggers a power off, a `halt -w` fakes a shut down by logging it without actually shutting down, and `halt -f` forces a halt. Use a `-v` for verbose mode.

This lab is now complete!