An introduction to services, runlevels, and rc.d scripts - Linux.com

By The Linux Foundation

Published: 2006-01-12 · Archived: 2026-04-05 18:23:26 UTC

A Linux service is an application (or set of applications) that runs in the background waiting to be used, or carrying out essential tasks. I've already mentioned a couple of typical ones (Apache and MySQL). You will generally be unaware of services until you need them.

How can you tell what services are running, and more importantly, how can you set up your own?

System V vs. BSD

In this article I'm dealing with System V (derived from AT&T System V) based distributions. This is the most common Linux init system. Another is based on BSD (Berkeley Software Distribution). What's the difference between the two? Basically BSD doesn't have any runlevels. This means that System V gives a lot more flexibility to a system administrator.

Most Linux distros put startup scripts in the rc subdirectories (rc1.d, rc2.d, etc.), whereas BSD systems house the system scripts in /etc/rc.d. Slackware's init setup is similar to BSD systems, though Slackware does have runlevels and has had System V compatibility since Slackware 7.

Let's start by looking at how the system is set up, and in particular at the directory /etc/rc.d. Here you will find either a set of files named rc.0, rc.1, rc.2, rc.3, rc.4, rc.5, and rc.6, or a set of directories named rc0.d, rc1.d, rc2.d, rc3.d, rc4.d, rc5.d, and rc6.d. You will also find a file named /etc/inittab. The system uses these files (and/or directories) to control the services to be started.

If you look in the file /etc/inittab you will see something like:

```
id:4:initdefault:l
```

```
0:0:wait:/etc/rc.d/rc.0l
```

```
6:6:wait:/etc/rc.d/rc.6x
```

```
1:4:wait:/etc/rc.d/rc.4
```

The boot process uses these parameters to identify the default runlevel and the files that will be used by that runlevel. In this example, runlevel 4 is the default and the scripts that define runlevel 4 can be found in

/etc/rc.d/rc.4.

And what is a runlevel? You might assume that this refers to different levels that the system goes through during a boot up. Instead, think of the runlevel as the point at which the system is entered. Runlevel 1 is the most basic configuration (simple single user access using an text interface), while runlevel 5 is the most advanced (multi-user, networking, and a GUI front end). Runlevels 0 and 6 are used for halting and rebooting the system.

There are, however, differences between Linux distributions. For instance, Fedora uses runlevel 5 for X-based logins, whereas Slackware uses runlevel 4 to do the same job. Therefore, you should check your documentation before making any changes. This table shows a generic list of configurations (and some examples of different distros) taken from *Linux – The Complete Reference* (R.Peterson, Osbourne/McGraw-Hill).

Run Level	**Generic**	**Fedora Core**	**Slackware**	**Debian**
0	Halt	Halt	Halt	Halt
1	Single-user mode	Single-user mode	Single-user mode	Single-user mode
2	Basic multi-user mode (without networking)	User definable (Unused)	User definable – configured the same as runlevel 3	Multi-user mode
3	Full (text based) multi-user mode	Multi-user mode	Multi-user mode – default Slackware runlevel	
4	Not used	Not used	X11 with KDM/GDM/XDM (session managers)	Multi-user mode
5	Full (GUI based) multi-user mode	Full multi-user mode (with an X-based login screen) – default runlevel	User definable – configured the same as runlevel 3	Multi-user mode
6	Reboot	Reboot	Reboot	Reboot

As you can see there are slight (but important) differences between Linux distributions. One thing is common between them — if you want to change the default level, you must edit /etc/initab. You will need to be root or use sudo to edit this file, naturally.

Why would you want to change the runlevel? Normally you will only use full GUI or text multi-user mode — runlevels 4 or 5. You'd only want runlevels 1 or 2 if you have some system problems and you want the most basic access. Runlevels 0 and 6 should never be used as a default (for obvious reasons — you don't want the system to

shutdown or reboot as soon as you turn it on). You can, of course, change mode whilst the system is running. Type init followed by the required runlevel e.g.:

init 6

This will reboot the system.

The boot process, or to be more accurate the init command, will decide the runlevel to select (in the example above it's 4) and from that will decide the rc.d script files to be run. In this case either the file /etc/rc.d/rc.4 or any files in the directory /etc/rc.d/rc4.d. Let's look at an example rc.d script file. Here's the default rc.4 file for Slackware 10.2:

```
# Try to use GNOME's gdm session manager:
```

```
if [ -x /usr/bin/gdm ];
```

```
then  exec /usr/bin/gdm -nodaemonfi
```

```
# Not there?  OK, try to use KDE's KDM session manager:
```

```
if [ -x /opt/kde/bin/kdm ];
```

```
 then  exec /opt/kde/bin/kdm -nodaemonfi
```

```
# If all you have is XDM, I guess it will have to do:
```

```
if [ -x /usr/X11R6/bin/xdm ];
```

```
 then  exec /usr/X11R6/bin/xdm -nodaemonfi
```

A quick guide to the boot process

When you boot your computer, the first thing that it will do is load the bootloader — either GRUB or LILO in most cases. The bootloader will then load the Linux kernel — the core operating system. Next, a process called init starts. This process reads the file /etc/inittab to establish the runlevel to use. The runlevel is the start mode for the computer.

Once init knows the runlevel it will look for the appropriate files or directories as defined in /etc/initab.

> Init will then either run the script files defined by /etc/initab, or run the script files in the directories defined by /etc/initab (depending on the set up of your system).
>
> Finally, init will present you with the logon mode that you've selected.

As you would expect, since runlevel 4 is the Slackware X11 mode, the commands are all concerned with the setting up of the graphical interface.

In the other distros (such as Fedora and Debian) you'll find that the scripts to be run are actually symbolic links to files in the directory /etc/init.d — the central repository for all startup scripts. So all you have to do is to write your startup script, place it in /etc/init.d, and then create a symbolic link to it from the appropriate runlevel directory (or runlevel file, if that's what your system uses).

For example, runlevel 2 is the default runlevel for Debian in non-GUI mode. If you're running Apache 2 on Debian, you'd find an init script for Apache 2 under /etc/init.d called apache2. A symlink, S91apache2, points to /etc/init.d/apache2 from /etc/rc2.d — this tells init to start Apache 2 in runlevel 2, but only after other services with lower S numbers.

When the system is shut down, there is another symlink in the /etc/rc0.d and /etc/rc6.d directories (halt and reboot, respectively) that starts with a K instead of an S, which tells init to shut down the process.

If this all still sounds a bit too complicated, you can instead simply make use of the /etc/rc.d/rc.local file. This script file is run once, before all other scripts have run but before the logon prompt appears. By default it looks something like:

```
#!/bin/bash## /etc/rc.local - run once at boot time
```

```
# Put any local setup commands in here:
```

You can append your instructions onto the end of the file by defining another script to be run:

```
/root/bin/start_bb
```

Or you can modify rc.local by adding the commands themselves:

```
modprobe -r uhcimodprobe usb-uhcieciadsl-startiptable -Fiptables -A
```

```
INPUT -i ppp0 -p tcp --syb -j DROPnetdate time.nist.gov
```

Here a USB modem is initialized, a connection set up to a broadband network, some basic security is set up, and then the local time is synchronized with a time server. You can also start Apache or MySQL:

```
apachectl startecho "/usr/bin/mysqld_safe &" | su mysql
```

Note that some distros, such as Debian, do not use rc.local for startup scripts. See the Debian FAQ if you'd like to add startup scripts for Debian or Debian-derived distros.

One final thought — in addition to startup scripts (for rc.local), try to remember to write close-down scripts to be added to rc.0 and rc.6. This ensures that your services are shut down neatly and not left in strange states when the system halts.

About shutting down — how do you stop a service from starting when you reboot? It's just the reverse of what we've already looked at. Either edit the relevant runlevel file (comment the lines out rather than removing them) or remove the link from the runlevel directory. Note that it may not be necessary to do this manually, as many distros include tools to manage services. For example, Red Hat and Fedora use chkconfig, while Debian uses update-rc.d.

From this brief discussion, I hope you can see how useful rc.d scripts can be when it comes to controlling the services to be run on your PC. You can now add your own as required, as well as look at existing ones that you may not require and which are slowing down your logon or overloading your PC.
