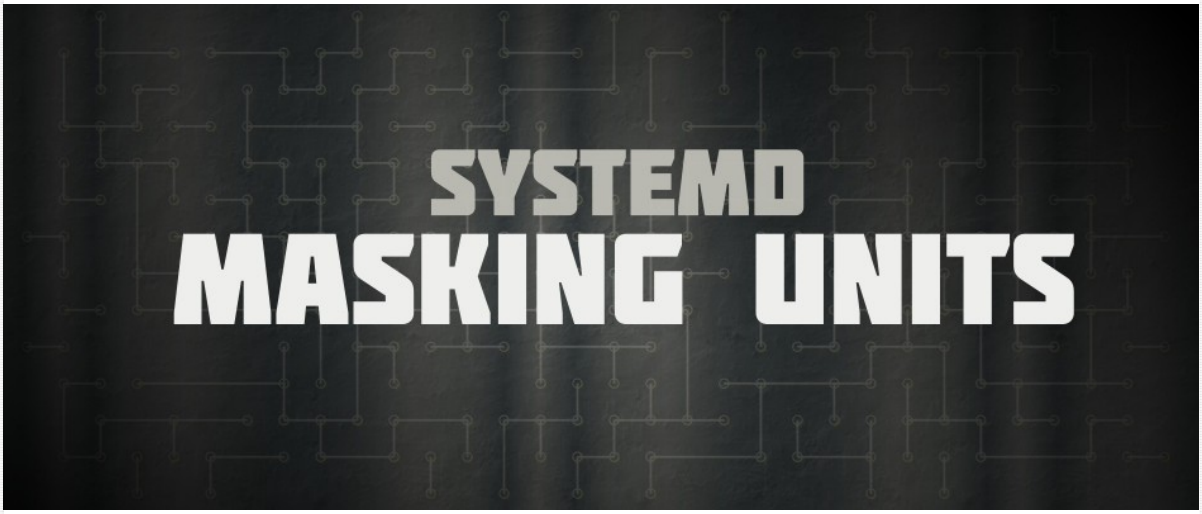
systemd: Masking units



Welcome back to our continuing series on systemd. In an earlier article, we discussed simple commands to manage the services on your Fedora system. Here we are going to discuss a subtlety of systemd that makes it very powerful: masking. Before we get to masking, though, let's review other levels of "off" for a service.

The first two levels of "off"

You may know these common *systemctl* commands:

```
systemctl start httpd.service
systemctl stop httpd.service
```

These commands take effect immediately to start or stop the web server (httpd) based on its unit file. You can think of *stop* as the **first level of "off"** for a unit in systemd.

You might also recall these common commands:

```
systemctl enable httpd.service
systemctl disable httpd.service
```

These commands do not take effect immediately. They ensure the service will be started (or not) after the next system boot. You can think of *disable* as the **second level of "off"** for systemd units.

You probably can tell where we're going with this. Masking is a third level, less often used but quite powerful. But before we look at masking, we need to understand why it exists: dependencies.

Unit dependencies

It's important to remember, though, that systemd is more powerful than the legacy *init* startup system. The old *init* system would start or stop services in the order determined by their file names. These files were usually links placed in a directory named after the *runlevel* in which they should operate. (If you need to refresh your memory on runlevels, refer to this earlier systemd article.) For instance, consider this old initscript link:

```
/etc/rc3.d/S40myservice -> /etc/init.d/myservice
```

This link exists in the */etc/rc3.d/* folder. The link determined what happened when the system entered runlevel 3, or multi-user mode. The *S* in the name means the script will be run in start mode. A *K* instead would indicate the script would be stopped (killed). The number *40* is an indicator of ordering. Lower numbered scripts run before this one, and higher numbered scripts run later. The actual script itself lived in the */etc/init.d/* folder.

Compared to systemd's capabilities, this seems a bit primitive! Recall that systemd can understand dependencies between units. This means systemd can tell when one unit must be started to allow another to run.

Furthermore, systemd uses this information to start a required service, **even if disabled,** to satisfy dependencies. As an example, let's look at some the dependencies of *httpd.service:*

```
$ systemctl list-dependencies httpd.service
httpd.service
● ├─-.mount
● ├─system.slice
● ├─tmp.mount
● ├─var.mount
● └─basic.target
●   └─-.mount
●   ├─alsa-restore.service
```

```
...
```

This graph shows us all the units on which the web server unit depends. Notice the graph is recursive, meaning it also shows dependencies of dependencies.

Let's look at the effect of disabling a dependency. Assume that *httpd.service* is enabled in systemd. Now we disable the system.slice service with the following command:

```
systemctl disable system.slice
```

By the way, disabling this unit isn't a great idea. It could theoretically affect every service that systemd starts! As it happens, though, it doesn't matter if we do. Because *httpd.service* depends on *system.slice*, disabling has no effect. systemd understands the requirements of *httpd.service* and therefore it starts *system.slice* regardless.

Now you can hopefully start to understand why masking exists. It allows the administrator to force systemd to do as instructed, even if it's illogical.

Masking: the third level

You may recall that systemd uses several locations to store unit information. If not, feel free to refer to our previous article on unit files to refresh your memory. To record unit masking, systemd uses the local system configuration files in */etc/systemd/*. It writes a link file pointing to */dev/null*, the famous "nothing" file in UNIX and Linux. So for instance, masking *httpd.service* would result in this link:

```
$ sudo systemctl mask httpd.service
Created symlink from /etc/systemd/system/httpd.service to /dev/null.
```

Note this is equivalent to running this command:

```
$ sudo ln -s /dev/null /etc/systemd/system/httpd.service
```

Now attempt to start the web server manually:

```
systemctl start httpd.service
```

You'll see the following error message:

```
Failed to start httpd.service: Unit httpd.service is masked
```

This is the **third level of "off"** in systemd. If you boot with a unit masked, it will not

run even to satisfy dependencies. Masking is powerful for this reason. But like other powerful commands, you should use it carefully. If you mask an important unit (like the aforementioned *system.slice*), you could stop your system from booting normally. To unmask the unit, use this command:

```
systemctl unmask httpd.service
```

Hopefully you can now appreciate even more the power of systemd units and dependencies.For some additional explanation of masking, refer to this post from the "systemd for administrators" blog series. See you soon for our next systemd installment!

Ashutosh Sudhakar Bhakare

November 18, 2015

For System Administrators, For Users

Previous post

Next post

2 Comments

ADD YOURS

Aditya Konarde
November 20, 2015 at 11:02

Nice and informative article.. Particularly helped clear my confusion between enable and start. Keep the good work going.

Westerj
November 24, 2015 at 23:12

Thanks for shedding a little light on the parallelisms and distinctions between Systemd and Traditional Init. This will help admins navigate the changes if the need arises.

Leave a Reply

Your email address will not be published.

Name

Email

Website

Post Comment

Notify me of follow-up comments by email.

Notify me of new posts by email.