(https://u2
4.gov.ua/)

Systemd: Service File Examples

Posted on March 20, 2018 (/systemd-service-file-example/) by admin (/author/admin/)

(/systemd-service-file-example/)

Most Linux distributions use systemd as a system and service manager.

The `systemctl` is the main command in systemd, used to control services.

In this tutorial i will show how to create a systemd service file that will allow you to control your service using the `systemctl` command, how to restart systemd without reboot to reload unit files and how to enable your new service.

I will also show and describe the most important systemd service file options with the live examples of the systemd service files.

Create Systemd Service File

Create a systemd service file `/etc/systemd/system/foo-daemon.service` (replace the `foo-daemon` with your service name):

```
$ sudo touch /etc/systemd/system/foo-daemon.service
$ sudo chmod 664 /etc/systemd/system/foo-daemon.service
```

Open the `foo-daemon.service` file and add the minimal service configuration options that allow this service to be controlled via `systemctl` :

```
Description=Foo
[Service]
ExecStart=/usr/sbin/foo-daemon

[Install]
WantedBy=multi-user.target
```

> **Path To Daemon:** If you don't know the full path to a daemon, try `which foo-daemon` .

Once the service file is changed, it needs to reload systemd configuration:

```
$ sudo systemctl daemon-reload
```

Now you should be able to `start` , `stop` , `restart` and check the service `status`

```
$ sudo systemctl start foo-daemon
$ sudo systemctl stop foo-daemon
$ sudo systemctl restart foo-daemon
$ systemctl status foo-daemon
```

To configure a service to start automatically on boot, you need to `enable` it:

```
$ sudo systemctl enable foo-daemon
```

To check the service logs, run:

```
$ journalctl -u foo-daemon
```

Systemd Service File Options

Systemd service files typically consist of three sections.

The common configuration items are configured in the generic `[Unit]` and `[Install]` sections.

The service specific configuration options are configured in the `[Service]` section.

Important `[Unit]` Section Options

Option	Description
`Description`	A short description of the unit.
`Documentation`	A list of URIs referencing documentation.
`Before` , `After`	The order in which units are started.
`Requires`	If this unit gets activated, the units listed here will be activated as well. If one of the other units gets deactivated or fails, this unit will be deactivated.
`Wants`	Configures weaker dependencies than `Requires` . If any of the listed units does not start successfully, it has no impact on the unit activation. This is the recommended way to establish custom unit dependencies.

`Conflicts`	If a unit has a `Conflicts` setting on another unit, starting the former will stop the latter and vice versa.

A complete list of `[Unit]` section options:

```
$ man systemd.unit
```

Important `[Install]` Section Options

Option	Description
`Alias`	A space-separated list of additional names for the unit. Most `systemctl` commands, excluding `systemctl enable`, can use aliases instead of the actual unit name.
`RequiredBy`, `WantedBy`	The current service will be started when the listed services are started. See the description of `Wants` and `Requires` in the `[Unit]` section for details.
`Also`	Specifies a list of units to be enabled or disabled along with this unit when a user runs `systemctl enable` or `systemctl disable`.

A complete list of `[Install]` section options:

```
$ man systemd.unit
```

Important `[Service]` Section Options

Option	Description
`Type`	Configures the process start-up type. One of: `simple` (default) – starts the service immediately. It is expected that the main process of the service is defined in `ExecStart`. `forking` – considers the service started up once the process forks and the parent has exited. `oneshot` – similar to `simple`, but it is expected that the process has to exit before systemd starts follow-up units (useful for scripts that do a single job and then exit). You may want to set `RemainAfterExit=yes` as well so that systemd still considers the service as active after the process has exited. `dbus` – similar to `simple`, but considers the service started up when the main process gains a D-Bus name. `notify` – similar to `simple`, but considers the service started up only after it sends a special signal to systemd. `idle` – similar to `simple`, but the actual execution of the service binary is delayed until all jobs are finished.
`ExecStart`	Commands with arguments to execute when the service is started. `Type=oneshot` enables specifying multiple custom commands that are then executed sequentially. `ExecStartPre` and `ExecStartPost` specify custom commands to be executed before and after `ExecStart`.
`ExecStop`	Commands to execute to stop the service started via `ExecStart`.
`ExecReload`	Commands to execute to trigger a configuration reload in the service.
`Restart`	With this option enabled, the service shall be restarted when the service process exits, is killed, or a timeout is reached with the exception of a normal stop by the `systemctl stop` command.
`RemainAfterExit`	If set to `True`, the service is considered active even when all its processes exited. Useful with `Type=oneshot`. Default value is `False`.

(https://pl2 A complete list of `[Service]` section options:

4.gov.ua/)

```
$ man systemd.service
```

Systemd Service File Examples

```
[Unit]
Description=The NGINX HTTP and reverse proxy server
After=syslog.target network.target remote-fs.target nss-lookup.target

[Service]
Type=forking
PIDFile=/run/nginx.pid
ExecStartPre=/usr/sbin/nginx -t
ExecStart=/usr/sbin/nginx
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s QUIT $MAINPID
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

```
[Unit]
Description=The Apache HTTP Server
After=network.target remote-fs.target nss-lookup.target

[Service]
Type=notify
EnvironmentFile=/etc/sysconfig/httpd
ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND
ExecReload=/usr/sbin/httpd $OPTIONS -k graceful
ExecStop=/bin/kill -WINCH ${MAINPID}
KillSignal=SIGCONT
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

```
[Unit]
Description=Redis persistent key-value database
After=network.target

[Service]
ExecStart=/usr/bin/redis-server /etc/redis.conf --daemonize no
ExecStop=/usr/bin/redis-shutdown
User=redis
Group=redis

[Install]
WantedBy=multi-user.target
```

For more examples, check the systemd.service (https://www.freedesktop.org/software/systemd/man/systemd.service.html) and systemd.unit (https://www.freedesktop.org/software/systemd/man/systemd.unit.html) man pages.

Was it useful? Share this post with the world!