systemd is a Linux initialization system and service manager that includes features like on-demand starting of daemons, mount and automount point maintenance, snapshot support, and processes tracking using Linux control groups. systemd provides a logging daemon and other tools and utilities to help with common system administration tasks.

Lennart Pottering and Kay Sievers wrote systemd, inspired by macOS’s launchd and Upstart, with the goal of creating a modern and dynamic system. Notably, systemd provides aggressive parallelization capabilities and dependency-based service control logic, allowing for services to start in parallel and leading to a quicker boot time. These two aspects were present in Upstart, but improved upon by systemd.

systemd is the default init system for the major Linux distributions but is backwards compatible with SysV init scripts. SysVinit is an initialization system which predates systemd and uses a simplified approach to service startup. systemd not only manages system initialization, but also provides alternatives for other well known utilities, like cron and syslog. Because systemd does several things within the Linux user space, many have criticized it for violating the UNIX philosophy, which emphasizes simplicity and modularity.

## The Linux Boot Process and systemd

Linux requires an initialization system during its boot and startup process. At the end of the boot process, the Linux kernel loads systemd and passes control over to it and the startup process begins. During this step, the kernel initializes the first user space process, the systemd init process with process ID 1, and then goes idle unless called again. systemd prepares the user space and brings the Linux host into an operational state by starting all other processes on the system.

## systemd Units

systemd introduces the concept of systemd units and there are several types, such as a service unit, mount unit, socket unit and slice unit. Units are defined in unit configuration files, which include information about the unit type and its behavior.

For most distributions using systemd, unit files are stored in the following directories:

* The /usr/lib/systemd/user/ directory is the default location where unit files are installed by packages. Unit files in the default directory should not be altered.
* The /run/systemd/system/ directory is the runtime location for unit files.
* The /etc/systemd/system/ directory stores unit files that extend a service. This directory will take precedence over unit files located anywhere else in the system.

### Mount Units

A mount unit configuration file contains information about a file system mount point that is controlled and supervised by systemd. To find the existing mount units on your machine, use the systemctl tool to view a complete list:

systemctl list-units --type=mount

To view information about a specific mount unit, use the systemctl status command with the unit’s name:

systemctl status sys-fs-fuse-connections.mount

This example displays a single entry for the FUSE Control File System mount point:

The systemctl output provides the location of the mount unit configuration file, along with information on the state of the mount, location of the mount, links to documentation, running tasks, and its corresponding CGroup. Many of these details are defined in the mount unit’s configuration file.

FUSE is a file system framework that provides an interface for user space programs to export a virtual file system to the Linux kernel. It can also be used to provide data access with a file system directory structure and file operations to any object.

Inspect the contents of the FUSE control file system’s mount unit configuration file with systemctl:

systemctl cat sys-fs-fuse-connections.mount

All unit files must contain a [Unit] section that outlines generalized options, dependencies and conditions for the unit.

A mount unit file must contain a [Mount] section.

There are many other unit file types available in systemd.

## systemd Tools

systemd makes common system administration tasks easier to manage with its systemctl and journalctl commands. systemctl can be used to gather detailed information about the overall state of your server and any individual unit type. It can stop and start the server and modify the system state.

systemd’s journalctl tool provides a centralized process and system logging tool. This command allows you to query the systemd journal, which creates and maintains indexed journals from logging information that is pooled from different areas within the system; areas like standard output and standard error of service units, log messages via syslog, and kernel log messages. In this way, system administrators can use a single tool to monitor and debug a server.

Create a script or executable that the service will manage.

#!/bin/bash

echo "devops.service: ## Starting ##" | systemd-cat -p info

while :

do

TIMESTAMP=$(date '+%Y-%m-%d %H:%M:%S')

echo "devops.service: timestamp ${TIMESTAMP}" | systemd-cat -p info

sleep 60

done

The script doesn’t do a whole lot, but there are a few points worth noting.

* The two echo lines are piped through systemd-cat, a program that takes the output from a program and sends it to the journal. Entries to the journal are given a priority. We’re using the -p (priority) option to indicate that our messages are for information (info) only. They’re not important errors or warnings.
* There is an endless while loop.
* The TIMESTAMP variable is set to the current date and time. This is formatted into a message and sent to the journal.
* The script then sleeps for 60 seconds.
* After 60 seconds the loop is repeated. So, this script writes a timestamped message to the journal once per minute.

Then wee need to make script executable.

sudo chmod +x ./devops.sh

We’ll copy the script to the /usr/local/bin directory.

sudo cp devops.sh /usr/local/bin

Each program that is started by systemd has a definition file, called a service unit file. This holds certain attributes that systemd can use to locate and launch the program, and to define some of its behavior.

We need to create a unit file for our new service, but it is prudent to make sure none of the existing unit files have the name we want to give our new service

sudo systemctl list-unit-files --type=service

Our service is going to be called “devops.service”. No unit files have that name, so we can proceed and create our unit file.

[Unit]

Description=Devops Service Example

Wants=network.target

After=syslog.target network-online.target

[Service]

Type=simple

ExecStart=/usr/local/bin/devops.sh

Restart=on-failure

RestartSec=10

KillMode=process

MemoryMax=1M

[Install]

WantedBy=multi-user.target

Then we need copy this service to /etc/systemd/system .

sudo cp devops.service /etc/systemd/system/

The entries have these meanings. These are typical entries. Our simple service doesn’t actually need most of them, but including them allows us to explain them.

* **Description:** This is a text description of your service.
* **Wants:** Our service wants—but doesn’t require—the network to be up before our service is started.
* **After:** A list of unit names that should be started after this service has been successfully started, if they’re not already running.
* **Type:** Simple. systemd will consider this service started as soon as the process specified by ExecStart has been forked.
* **ExecStart:** The path to the process that should be started.
* **Restart:** When and if the service should be restarted. We have set it to “on-failure.”
* **RestartSec:** How long to wait before attempting to restart the service. This value is in seconds.
* **KillMode:** Defines how systemd should kill the process if we ask systemctl to stop the service. We have this set to “process.” This causes systemd to use the SIGTERM signal on the main process only. If our service was a non-trivial program instead of a simple script, we would set this to “mixed” to ensure that any spawned processes were also terminated.
* **WantedBy:** We have this set to “multi-user.target”, which means the service should be started as long as the system is in a state where multiple users can log in, whether or not a graphical user interface is available.

The unit file doesn’t need to be executable, but the permissions on the unit file should restrict who can edit it. You don’t want a malicious or mischievous user changing the unit file so that it executes a different program altogether.

This command will give the owner read and write permissions, and read permissions to the group. Others will have no permissions.

sudo chmod 640 /etc/systemd/system/devops.service

We can have systemctl check the syntax of our unit file for us, even if the service isn’t running yet. Any errors will be reported. (Actually, the “.service” part is optional for most commands.)

systemctl status devops.service

When you add a new unit file or edit an existing one, you must tell systemd to reload the unit file definitions.

sudo systemctl daemon-reload

If you want a service to be launched at startup you must enable it:

sudo systemctl enable devops

Enabling a service doesn’t start it, it only sets it to be launched at boot time. To start the service now, you must use systemctl with the start option.

sudo systemctl start devops

After manually starting the service or after rebooting the computer, we can verify that our service is running correctly.

sudo systemctl status devops.service

* The green dot means our service is up and running smoothly.
* The name of the service is “devops.service”, and the long description is the one that we provided in the unit file.
* We’re shown which unit file has been loaded “/etc/systemd/system/devops.service”.
* The service is active, and the time the service was launched is listed for us.
* Its PID is 7762.
* There are two tasks associated with the service.
* A total of 1.5 mb are being used by the service.
* The control group includes the “devops.sh” script and the sleep command, which has been launched by “devops.sh.” Most of the time, the sleep command will be doing the work for this service.

We’re also shown the last 10 journal entries produced by this service. Unsurprisingly, they are all one minute apart.