Lab 03.02
systemd
IN719 Systems Administration

1 Introduction

Unix and similar systems have long relied upon init systems to bootstrap and manage resources as the server starts up and shuts down, and more generally as the host transitions between various states. Since about 2015, the primary init system used on Linux has been *systemd*. The adoption of systemd has been a contentious process and the ensuing controversy is well worth discussing at a pub after about two beers. In the meantime, however, we need to familiarise ourselves with it so that we can effectively do our work.

On one of your servers, run the command `pstree -p` to see a tree diagram to see how your host's processes are started either by systemd, or indirectly started by a process that was started by systemd.

2 Key Concepts

Units

The primary abstraction in systemd is the *unit*. A unit is a system resource that systemd is able to manage. systemd can manage 12 different types of units:

- device
- mount
- path
- scope
- service
- slice
- snapshot
- socket
- swap
- target
- timer

We will focus primarily on service units, which represent services running on a host, like a web server or a database server.

More concretely, a unit is something defined in a *unit file*. Unit files conform to a specific format and syntax that we will see shortly. These files may be placed in four directories:

1. `/etc/systemd/system`
2. `/run/systemd/system`
3. `/lib/systemd/system`
4. `/usr/lib/systemd/system`

If you look in some of these directories you will find unit files of various types. Their filenames indicate what sorts of units they are. Service unit files end in `.service`, device unit files end in `.device`, and so on.

systemd will check these directories in this order, taking the first matching unit file found. Locally defined units should be placed in `/etc/systemd/system`, so any unit files we write will be placed there.

Targets

Targets are special systemd units that are themselves collections of other units that specify a desired overall system state. For example, a host will have a default target that identifies a set of units that should be active when the host boots up. See all the targets defined on your server with the command

```
systemctl list-unit-files --type=target
```

3 An Example

Let's see how some of these ideas work by creating a small example. First, write a shell script in your home directory like the following:

```
#!/bin/bash
while $(sleep 30);
do
    echo "Hello, systemd world"
done
```

Name your script `hello.sh` and make it executable.

To turn this into a systemd service, we need to write a unit file for it. Write a unit file named `hello.service` with the following contents:

```
[Unit]
Description=Hello world service
After=systemd-user-sessions.service
[Service]
SyslogIdentifier=HelloWorldService
Type=simple
ExecStart=/home/<your home dir>/hello.sh
```

Copy this file to `/etc/systemd/system`. Verify that systemd recognises your unit with the command

```
systemctl list-unit-files | grep hello.service
```

Our unit file has two sections: `[Unit]` and `[Service]`. Some of the entries are self explanatory. The `After` directive identifies what units should be started before this one is. The `Type` of our service is `simple`, but more commonly we will see `forking` services for daemons.

Start the new service with the command

```
systemctl start hello.service
```

and check that it is running with the command

```
systemctl status hello.service
```

Interestingly, our very simple service writes logs. We can see the logged output with the command

```
journalctl -u hello -e
```

The `-u` option tells `journalctl` that we only want to see the log entries for the `hello` service and the `-e` says to show the entires from the most recent entry and work backwards. Note that many services write their own log files in addition to what they write to systemd's journal.

4 `systemd` Quick Reference

To wrap up we will review some systemd commands that we will find useful this semester and beyond. We will use the `cron` service as an example. Some of these commands require `sudo` privileges.

`systemctl start cron.service`

Starts the service[1].

`systemctl status cron.service`

Prints a report on the service's status, including recent journal entries

`systemctl stop cron.service`

Stops the service

`systemctl cat cron.service`

Prints the services unit file.

`systemctl edit cron.service --full`

Opens the services unit file in the default editor. Note that this and the cat command above finds the effective unit file in the directories we listed above. We don't need to look around to find the file ourselves.

Note also that cron's unit file has an `[Install]` section, which means that this service can be *enabled* to start at boot.

`systemctl enable cron.service`

Identifies the service to be started automatically on boot.

`systemctl disable cron.service`

Removes the service from the list of those started at boot.

`journalctl -u nginx.service`

Show journal entries for the service. Note that we often want to add the `-e` option to see the newest entries.

[1] Duh.