
# A Brief Introduction to Systemd

Peter Ashford

Ashford Computer Consulting Service

4/12/2018

# What We'll Cover

This presentation covers the Linux systemd startup system.

The intended audience for this presentation is end-users and System Administrators.

This presentation is designed to provide enough information to allow a System Administrator to get their job done.

# What We Won't Cover

This presentation won't cover:

- All the possible commands and options. That's what the man page is for.
- Tips and Tricks for service developers. That's beyond the scope of this presentation.
- Security enhancements and Resource Management (quotas). While I'd like to cover that, there's enough there for another presentation.

# History

In Research Unix (starting in the 1970's), the entire startup was done by the "`/etc/rc`" script. Local modifications were made directly to that script.

BSD Unix added "`/etc/rc.local`" to the startup sequence, to allow simple customization of the startup sequence without a need to modify "`/etc/rc`".

# History

The sysVinit system that we're all used to was released in the early 1980's. This provided a framework for system startup and shutdown that is adequate, although sometimes barely, to handle even the most complex modern systems.

Why are we looking at replacements? Startup time is too long, due to the single-threaded nature of sysVinit. Also, some people believe that there's a "better" way to manage the state of a system.

# History

The Ubuntu project released Upstart in 2006. This was claimed to be a complete replacement for sysVinit. It was functional, but had its share of limitations and problems, which limited adoption.

Ubuntu migrated to systemd in 2015.

At this time, the only Linux distribution that still uses Upstart is ChromeOS.

# What is Systemd?

Systemd is a set of utilities that are used in place of sysVinit to manage services, including system startup and shutdown.

Systemd uses Targets and Service Groups to allow a Linux system to be brought to any defined state.

Systemd was developed by RedHat, and is currently a *de facto* requirement for GNOME.

# Issues

Systemd is viewed by many people as being excessively complex. This complexity is seen by some people as a violation of the Unix philosophy.

In addition, some Linux distributions are seen as having been forced to adopt systemd due to dependencies of other software, primarily GNOME.

There is also concern that systemd will push the Linux community towards being more uniform.

# Why Use Systemd?

Systemd provides aggressive parallelization capabilities during system startup and shutdown.

Systemd limits the services it starts to those that are actually needed.  As an example, the *bluetoothd* service doesn't start unless there's a bluetooth interface in the system, or something requests the status of bluetooth.

# Remember

- In systemd, everything is a unit
- Units have:
  - A Type
  - A State
  - May have a Status
- A unit has a base-name that is usually unique
- The base-name is combined with the Type to create a complete unit name

# Dependencies and Ordering

- Systemd uses dependencies and ordering to determine when it starts and stops units.
- Dependencies can be soft ("`Wants`") or hard ("`Requires`").
- Ordering is done with "`After`" and/or "`Before`".

Without "After" or "Before", services start at the same time. This is likely to cause problems.

# Using Systemd

The "`systemctl`" command is used to query and manage the real-time state of systemd.

Usage:

systemctl [options] (command) [unit ...]

unit - the name(s) of the systemd unit being queried or managed.

# Using Systemd

Some commands for "`systemctl`" are:

- `systemctl` (no arguments)
- `systemctl list-unit-files` [query]
- `systemctl status` (unit)
- `systemctl start` (unit)
- `systemctl stop` (unit)
- `systemctl restart` (unit)
- `systemctl reload` (unit)
- `systemctl reboot`

# Using Systemd

- `systemctl` (no arguments)

This command will display the status of all units that systemd has loaded. This includes many units that aren't services. This is the same as the "`systemctl list-units`" command.

Note: "`systemctl --failed`" lists the units that failed when started.

# Using Systemd

- `systemctl list-unit-files [query]`

This command will list all units that systemd has files for and their state. Some common states are:

- static - Can not be enabled
- enabled - Can be started (could already be started)
- disabled - Can’t be started but can be triggered from another unit
- masked - Can’t be enabled or triggered from another unit

# Using Systemd

- `systemctl status` [unit]
- `service` (unit) `status`

This command will display the detailed status of a specific unit that systemd monitors, or a brief status of all loaded units if no unit is supplied.

# Using Systemd

- `systemctl start` (unit)
- `service` (unit) `start`

This command will start the specified unit. If this unit has a startup dependency on other units, and they're not running, they will be started as well.

# Using Systemd

- `systemctl stop` (unit)
- `service` (unit) `stop`

This command will stop the specified unit. If other units have shutdown dependencies on this unit, and they're running, they will be stopped as well.

# Using Systemd

- `systemctl restart` (unit)
- `service` (unit) `restart`

This command will stop then start the specified unit. If other units have shutdown dependencies on this unit, and they're running, they will be stopped as well. If this unit has a startup dependency on other units, and they're not running, they will be started as well.

# Using Systemd

- `systemctl reload` (unit)
- `service` (unit) `reload`

This command will request the specified unit to reload its configuration file.

# Using Systemd

- `systemctl reboot`

This command will reboot the system and come up to the default target. This is equivalent to the sysVinit command "`init 6`" or "`shutdown -r`".

# Using Systemd

Some more commands for "`systemctl`" are:

- `systemctl enable` (unit)
- `systemctl disable` (unit)
- `systemctl reenable` (unit)
- `systemctl isolate` (target)
- `systemctl set-default` (target)
- `systemctl list-units` [query]
- `systemctl cat` (unit)
- `systemctl edit` (unit)

# Using Systemd

- `systemctl enable` (unit)
- `systemctl disable` (unit)

This command will allow the specified unit to be started (enable), or disallow that unit from being started (disable). This will not start or stop the service unless the "`--now`" option is used.

# Using Systemd

- `systemctl reenable` (unit)

This command will disable then enable the specified unit. The unit will not be stopped or started. This is used to reset the symbolic links to the defaults, as specified in the "`[Install]`" section of the unit.

# Using Systemd

- `systemctl isolate` (target)

This command will stop all running units not specified for the target and start non-running units that are specified in for the target. This is equivalent to using the "`init`" command to change the run level with sysVinit.

# Using Systemd

- `systemctl set-default` (target)

This command will set the specified target to be the default. When the system is booted, this is the target to which the system will be brought. This is equivalent to editing the "`/etc/inittab`" file (not used with systemd) with sysVinit.

Note: "`systemctl get-default`" returns the current default target.

# Using Systemd

- `systemctl list-units [query]`

This command will list the units that meet the query criteria.

# Using Systemd

- `systemctl cat` (unit)

This command will display the files for the specified unit.

# Using Systemd

- `systemctl edit` (unit)

This command will bring up an editor on the specified unit. This simplifies porting complex services from sysVinit to systemd. Several options are available to modify what this command does. Please see the "`systemctl`" man page for details.

Note: Use "`--full`" to create a copy of the original file.

# Using Systemd

Still more commands for "`systemctl`" are:

- `systemctl mask` (unit)
- `systemctl unmask` (unit)
- `systemctl add-wants` (unit) (units)
- `systemctl add-requires` (unit) (units)
- `systemctl snapshot` (name)
- `systemctl delete` (name)

# Using Systemd

- `systemctl mask` (unit)

This command will disable the specified unit and keep it from being enabled. In addition, the unit won't be able to be started by another unit. The unit will not be stopped unless the "`--now`" option is used.

# Using Systemd

- `systemctl unmask` (unit)

This command will allow the specified unit to be enabled but will not enable or start the unit. The unit will be available to be started from other units.

# Using Systemd

- `systemctl add-wants` (unit) (units)
- `systemctl add-requires` (unit) (units)

This command will add the provided unit(s) to the "`Wants`" or "`Requires`" list of the specified unit.

Note: For proper startup order, the "`After`" and/or "`Before`" should be used.

# Using Systemd

- `systemctl snapshot` (name)

This command will create a named snapshot (target) of the systemd state. This snapshot can be used to return to the current state by the use of "`systemctl isolate`" command.

Snapshots are dynamic and are lost on reboot.

# Using Systemd

- `systemctl delete` (name)

This command will delete a systemd snapshot.

# Using Systemd

Some other systemd commands:

- `journalctl`
- `systemd-cgls`

# Using Systemd

- `journalctl` [query]

This command will display entries from the systemd log. Please see the man page for more information.

Note: The systemd log files are binary, not text.

# Using Systemd

- `systemd-cgls`

This command will display the systemd process tree, grouped by systemd unit.

# Using Systemd

Options

- `-H [user@](hostname)[:container]`
- `--host [user@](hostname)[:container]`
- `-m (container)`

These options will run the `systemctl` command on a remote host or on a container within the system.

SSH is used to connect to remote systems.

# Compatibility

Traditional sysVinit scripts will continue to function on a system running systemd. During system startup, “fake” service stubs are created to map the systemd functionality into the sysVinit scripts. This provides backwards compatibility with software designed for sysVinit.

It should be noted that sysVinit scripts are all executed in parallel. If startup timing is important, the script should be ported to systemd.

# Porting

Most traditional SysVinit scripts can be ported to systemd fairly easily. For an excellent walkthrough of this process, please see this web page:

http://0pointer.de/blog/projects/systemd-for-admins-3.html

# Systemd

Digging

In

Deeper

# Daemons

The following are some of the system daemons that are part of the systemd toolkit:

- systemd - The main systemd process. This is the replacement for "`/etc/init`".
- journald - Responsible for event logging.
- networkd - Manages network configuration.
- logind - Manages user logins.
- udevd - Manages devices.

# Systemd Configuration Files

Systemd uses a number of configuration files to perform its functions. When running as “init” (PID = 1), the following files are used to define its actions:

- /etc/systemd/system.conf
- /etc/systemd/system.conf.d/*.conf
- /run/systemd/system.conf.d/*.conf
- /usr/lib/systemd/system.conf.d/*.conf

# Systemd Configuration Files

Systemd uses a number of configuration files to perform its functions. When run as a user process, (PID != 1), the following files are used to define its actions:

- /etc/systemd/user.conf
- /etc/systemd/user.conf.d/*.conf
- /run/systemd/user.conf.d/*.conf
- /usr/lib/systemd/user.conf.d/*.conf

# Systemd Configuration Directories

Systemd uses three directory trees to store configuration files. These directories, and their use (by convention) are as follows:

- `/etc/systemd` - Local configuration information
- `/run/systemd` - Temporary files - usually for testing
- `/usr/lib/systemd` - Vendors and Upstream providers

# Systemd Unit Types

Systemd supports actions and monitoring through what it calls “units”. There are twelve different types of unit that systemd supports:

| | | | |
|---|---|---|---|
| Service | Mount | Automount | Target |
| Swap | Socket | Device | Snapshot |
| Timer | Path | Slice | Scope |

# Systemd Unit Files

Each unit has a file in at least one of:

- /etc/systemd/system
- /usr/lib/systemd/system

Files in "`/etc`" have precedence.

# Systemd Unit Files

Some units also have directories containing “drop-in” files in:

- /etc/systemd/system/*unit*.d/*.conf
- /run/systemd/system/*unit*.d/*.conf
- /usr/lib/systemd/system/*unit*.d/*.conf

Files in “`/etc`” have precedence over files in “`/run`”, which have precedence over files in “`/usr/lib`”. Files are processed in lexicographic order, regardless of the directory.

# Systemd Unit Files

Each systemd unit is described and configured by a file. Files for all types of unit have a "`[Unit]`" section. This section describes the unit, and provides some general information on the unit. This section also contains information on when the unit may be started and what other units are related to it.

For more information, see "`man systemd.unit`".

# Systemd Unit Files

In addition, the files for all types of unit have an "`[Install]`" section. This section is used when a unit is enabled or disabled.

For more information, see "`man systemd.unit`".

# Systemd `Service` Unit

A system service under systemd is represented as an instance of a Service Unit.

The configuration information is contained in the "`[Service]`" section of the unit file.

For more information, see "`man systemd.service`".

# Systemd `Mount` Unit

Mount units control mount points in the file-system. Mount units must be named after the mount-point directories they control. Alternatively, mounts can be listed in "`/etc/fstab`".

The configuration information is contained in the "`[Mount]`" section of the unit file.

For more information, see "`man systemd.mount`".

# Systemd `Automount` Unit

Automount units provide automount capabilities. Automount units must be named after the mount-point directories they control, and require matching mount units which will be activated as needed.

The configuration information is contained in the "`[Automount]`" section of the unit file.

For more information, see "`man systemd.automount`".

# Systemd `Target` Unit

Target units are useful to group other units, or provide well-known synchronization points during boot-up.

The configuration information is contained in the "`[Target]`" section of the unit file.

For more information, see "`man systemd.target`".

# Systemd Targets

The targets that are normally used for default, and the sysVinit run levels that they correspond to, are as follows:

| Run Level | Target |
|---|---|
| 0 | poweroff.target |
| 1 | rescue.target |
| 2, 3, 4 | multi-user.target |
| 5 | graphical.target |
| 6 | reboot.target |

# Systemd Swap Unit

Swap units are very similar to mount units and describes swap partitions or files of the operating system. Swap units must be named after the devices or files they control. Alternatively, swap files/partitions can be listed in "`/etc/fstab`".

The configuration information is contained in the "`[Swap]`" section of the unit file.

For more information, see "`man systemd.swap`".

# Systemd `Socket` Unit

Socket units are useful for socket-based activation. A socket unit can be used to start a service when a connection comes in on the specified port, similar to "`inetd`".

The configuration information is contained in the "`[Socket]`" section of the unit file.

For more information, see "`man systemd.socket`".

# Systemd `Device` Unit

Device units expose kernel devices in systemd and may be used to implement device-based activation. Device units must be named after the "`/dev`" or "`/sys`" paths they control.

There is no "`[Device]`" section.

For more information, see "`man systemd.device`".

# Systemd `Snapshot` Unit

Snapshot units can be used to temporarily save the state of the set of systemd units, which may later be restored by activating the saved snapshot unit. These temporary units have no file and are lost at reboot.

This is useful to preserve a functional state before making changes to the systemd configuration.

For more information, see "`man systemd.snapshot`".

# Systemd `Timer` Unit

Timer units are useful for triggering activation of other units based on timers. Each timer unit requires either a service unit with a matching name or a unit (any type) with the name specified in the configuration. Performs similar functions to "`cron`".

The configuration information is contained in the "`[Timer]`" section of the unit file.

For more information, see "`man systemd.timer`".

# Systemd `Path` Unit

Path units may be used to activate other services when file-system objects change or are modified. Each path unit requires either a service unit with a matching name or a unit (any type) with the name specified in the configuration.

The configuration information is contained in the "`[Path]`" section of the unit file.

For more information, see "`man systemd.path`".

# Systemd `Slice` Unit

Slice units may be used to group units which manage system processes (such as service and scope units) in a hierarchical tree for resource management purposes.

The configuration information is contained in the "`[Slice]`" section of the unit file.

For more information, see "`man systemd.slice`".

# Systemd Scope Unit

Scope units are similar to service units, but manage foreign processes instead of starting them. A foreign process is one not started by systemd.

These units have no file and are dynamically created by systemd.

For more information, see "`man systemd.scope`".

# Systemd Templates

Systemd can use template files to create multiple instances of the same unit from a single configuration file.

The command "`systemctl start getty@tty3`" could be configured with a "`getty@.service`" file.

# Systemd User Management

Systemd has the ability to manage System users through direct manipulation of the "`/etc/group`" and "`/etc/password`" files. These users are defined under "`/usr/lib/sysusers.d/*.conf`".

Accounts created with this ability will not be able to login. This is often used for service isolation.

For more information, see "`man systemd-sysusers`".

# References

- http://0pointer.de/blog/projects/systemd.html
- https://www.freedesktop.org/wiki/Software/systemd/
- http://0pointer.de/public/systemd-man/systemctl.html
- http://0pointer.de/public/systemd-man/systemd.unit.html

# Feedback

My goal is to improve this presentation.  To that end, I would appreciate feedback to:

- ashford@accs.com
- www.linkedin.com/in/peterashford