systemd

Ross Starritt

March 1, 2019

Mines Linux Users Group

Basics

What is `systemd`?

`systemd` builds your OS and then manages your daemons

Includes dependency management

And most of the people here at LUG use it

It is the default init program in most major distros

Runs as PID 1 when used as init

Composed of 69 individual binaries

Log everything that happens

Runs as a daemon itself

What is `systemd`?

`systemd` builds your OS and then manages your daemons
Includes dependency management

And most of the people here at LUG use it

It is the default init program in most major distros

Runs as PID 1 when used as init

Composed of 69 individual binaries

Log everything that happens

Runs as a daemon itself

What is `systemd`?

`systemd` builds your OS and then manages your daemons
Includes dependency management

And most of the people here at LUG use it

It is the default init program in most major distros

Runs as PID 1 when used as init

Composed of 69 individual binaries

Log everything that happens

Runs as a daemon itself

What is `systemd`?

`systemd` builds your OS and then manages your daemons
Includes dependency management

And most of the people here at LUG use it

It is the default init program in most major distros

Runs as PID 1 when used as init

Composed of 69 individual binaries

Log everything that happens

Runs as a daemon itself

What is `systemd`?

`systemd` builds your OS and then manages your daemons
Includes dependency management

And most of the people here at LUG use it

It is the default init program in most major distros

Runs as PID 1 when used as init

Composed of 69 individual binaries

Log everything that happens

Runs as a daemon itself

What is `systemd`?

`systemd` builds your OS and then manages your daemons
Includes dependency management

And most of the people here at LUG use it

It is the default init program in most major distros

Runs as PID 1 when used as init

Composed of 69 individual binaries

Log everything that happens

Runs as a daemon itself

What is `systemd`?

`systemd` builds your OS and then manages your daemons
Includes dependency management

And most of the people here at LUG use it

It is the default init program in most major distros

Runs as PID 1 when used as init

Composed of 69 individual binaries

Log everything that happens

Runs as a daemon itself

Philosophy of `systemd`

Every system service is a daemon encapsulated within a unit

System manager, not just service manager

Bridge gap between userspace and kernelspace

Distro agnostic, but Linux kernel specific

Handle system events

Philosophy of `systemd`

Every system service is a daemon encapsulated within a unit

System manager, not just service manager

Bridge gap between userspace and kernelspace

Distro agnostic, but Linux kernel specific

Handle system events

Philosophy of `systemd`

Every system service is a daemon encapsulated within a unit

System manager, not just service manager

Bridge gap between userspace and kernelspace

Distro agnostic, but Linux kernel specific

Handle system events

Philosophy of `systemd`

Every system service is a daemon encapsulated within a unit

System manager, not just service manager

Bridge gap between userspace and kernelspace

Distro agnostic, but Linux kernel specific

Handle system events

Philosophy of `systemd`

Every system service is a daemon encapsulated within a unit

System manager, not just service manager

Bridge gap between userspace and kernelspace

Distro agnostic, but Linux kernel specific

Handle system events

Ancestry

`systemd` was created to address perceived deficiencies in SysV

The old system ran a series of bash scripts , which were written by distro maintainers

Windows had SVChost and Apple had launchd
Both were successful programs based on services

Created in 2010, and saw wide adoption by 2015

Ancestry

`systemd` was created to address perceived deficiencies in SysV

The old system ran a series of bash scripts , which were written by distro maintainers

Windows had SVChost and Apple had launchd
Both were successful programs based on services

Created in 2010, and saw wide adoption by 2015

Ancestry

`systemd` was created to address perceived deficiencies in SysV

The old system ran a series of bash scripts , which were written by distro maintainers

Windows had SVChost and Apple had launchd
Both were successful programs based on services

Created in 2010, and saw wide adoption by 2015

Ancestry

`systemd` was created to address perceived deficiencies in SysV

The old system ran a series of bash scripts , which were written by distro maintainers

Windows had SVChost and Apple had launchd
Both were successful programs based on services

Created in 2010, and saw wide adoption by 2015

Structure

Daemons wrapped in units

Track units with cgroups

Units grouped into targets e.g. graphical.target launches everything needed to run a GUI

Communicate with sockets

Units can request state change of other units with jobs

Run transaction on jobs before running the job

Structure

Daemons wrapped in units

Track units with cgroups

Units grouped into targets e.g. graphical.target launches everything needed to run a GUI

Communicate with sockets

Units can request state change of other units with jobs

Run transaction on jobs before running the job

Structure

Daemons wrapped in units

Track units with cgroups

Units grouped into targets e.g. graphical.target launches everything needed to run a GUI

Communicate with sockets

Units can request state change of other units with jobs

Run transaction on jobs before running the job

Structure

Daemons wrapped in units

Track units with cgroups

Units grouped into targets e.g. graphical.target launches everything needed to run a GUI

Communicate with sockets

Units can request state change of other units with jobs

Run transaction on jobs before running the job

Structure

Daemons wrapped in units

Track units with cgroups

Units grouped into targets e.g. graphical.target launches everything needed to run a GUI

Communicate with sockets

Units can request state change of other units with jobs

Run transaction on jobs before running the job

Structure

Daemons wrapped in units

Track units with cgroups

Units grouped into targets e.g. graphical.target launches everything needed to run a GUI

Communicate with sockets

Units can request state change of other units with jobs

Run transaction on jobs before running the job

Transactions

Don't destroy the current state

1. Check for requested state change
2. Check for internal conflict and loops
3. Check with conflicts with the preexisting job queue
4. Merge with the job queue

`systemd` will attempt to solve any of the above issues
Jobs are only declined if resolution is impossible.

Transactions

Don't destroy the current state

1. Check for requested state change
2. Check for internal conflict and loops
3. Check with conflicts with the preexisting job queue
4. Merge with the job queue

`systemd` will attempt to solve any of the above issues
Jobs are only declined if resolution is impossible.

Transactions

Don't destroy the current state

1. Check for requested state change
2. Check for internal conflict and loops
3. Check with conflicts with the preexisting job queue
4. Merge with the job queue

`systemd` will attempt to solve any of the above issues
Jobs are only declined if resolution is impossible.

Transactions

Don’t destroy the current state

1. Check for requested state change
2. Check for internal conflict and loops
3. Check with conflicts with the preexisting job queue
4. Merge with the job queue

`systemd` will attempt to solve any of the above issues
Jobs are only declined if resolution is impossible.

Transactions

Don't destroy the current state

1. Check for requested state change
2. Check for internal conflict and loops
3. Check with conflicts with the preexisting job queue
4. Merge with the job queue

`systemd` will attempt to solve any of the above issues
Jobs are only declined if resolution is impossible.

Transactions

Don’t destroy the current state

1. Check for requested state change
2. Check for internal conflict and loops
3. Check with conflicts with the preexisting job queue
4. Merge with the job queue

`systemd` will attempt to solve any of the above issues
Jobs are only declined if resolution is impossible.

Unit Types

1. service - start and control daemons and their processes
2. socket - deal with IPC and networking sockets and socket-based activation
3. target - group of units
4. device - expose kernel devices
5. moint - control file system mount points
6. automount - allows parallelized boot and on-demand filesystem mounting
7. timer - trigger other units based on timers – replace cron
8. swap - encapsulate memory swap
9. path - activate a service when an object is changed on file system
10. slice - group units to manage processes in a hierarchical tree (for resource management)
11. scope - manage foreign processes (no starting)

Unit Types

1. service - start and control daemons and their processes
2. socket - deal with IPC and networking sockets and socket-based activation
3. target - group of units
4. device - expose kernel devices
5. moint - control file system mount points
6. automount - allows parallelized boot and on-demand filesystem mounting
7. timer - trigger other units based on timers – replace cron
8. swap - encapsulate memory swap
9. path - activate a service when an object is changed on file system
10. slice - group units to manage processes in a hierarchical tree (for resource management)
11. scope - manage foreign processes (no starting)

Unit Types

1. service - start and control daemons and their processes
2. socket - deal with IPC and networking sockets and socket-based activation
3. target - group of units
4. device - expose kernel devices
5. moint - control file system mount points
6. automount - allows parallelized boot and on-demand filesystem mounting
7. timer - trigger other units based on timers – replace cron
8. swap - encapsulate memory swap
9. path - activate a service when an object is changed on file system
10. slice - group units to manage processes in a hierarchical tree (for resource management)
11. scope - manage foreign processes (no starting)

Unit Types

1. service - start and control daemons and their processes
2. socket - deal with IPC and networking sockets and socket-based activation
3. target - group of units
4. device - expose kernel devices
5. moint - control file system mount points
6. automount - allows parallelized boot and on-demand filesystem mounting
7. timer - trigger other units based on timers – replace cron
8. swap - encapsulate memory swap
9. path - activate a service when an object is changed on file system
10. slice - group units to manage processes in a hierarchical tree (for resource management)
11. scope - manage foreign processes (no starting)

Unit Types

1. service - start and control daemons and their processes
2. socket - deal with IPC and networking sockets and socket-based activation
3. target - group of units
4. device - expose kernel devices
5. moint - control file system mount points
6. automount - allows parallelized boot and on-demand filesystem mounting
7. timer - trigger other units based on timers – replace cron
8. swap - encapsulate memory swap
9. path - activate a service when an object is changed on file system
10. slice - group units to manage processes in a hierarchical tree (for resource management)
11. scope - manage foreign processes (no starting)

Unit Types

1. service - start and control daemons and their processes
2. socket - deal with IPC and networking sockets and socket-based activation
3. target - group of units
4. device - expose kernel devices
5. moint - control file system mount points
6. automount - allows parallelized boot and on-demand filesystem mounting
7. timer - trigger other units based on timers – replace cron
8. swap - encapsulate memory swap
9. path - activate a service when an object is changed on file system
10. slice - group units to manage processes in a hierarchical tree (for resource management)
11. scope - manage foreign processes (no starting)

Unit Types

1. service - start and control daemons and their processes
2. socket - deal with IPC and networking sockets and socket-based activation
3. target - group of units
4. device - expose kernel devices
5. moint - control file system mount points
6. automount - allows parallelized boot and on-demand filesystem mounting
7. timer - trigger other units based on timers – replace cron
8. swap - encapsulate memory swap
9. path - activate a service when an object is changed on file system
10. slice - group units to manage processes in a hierarchical tree (for resource management)
11. scope - manage foreign processes (no starting)

Unit Types

1. service - start and control daemons and their processes
2. socket - deal with IPC and networking sockets and socket-based activation
3. target - group of units
4. device - expose kernel devices
5. moint - control file system mount points
6. automount - allows parallelized boot and on-demand filesystem mounting
7. timer - trigger other units based on timers – replace cron
8. swap - encapsulate memory swap
9. path - activate a service when an object is changed on file system
10. slice - group units to manage processes in a hierarchical tree (for resource management)
11. scope - manage foreign processes (no starting)

Unit Types

1. service - start and control daemons and their processes
2. socket - deal with IPC and networking sockets and socket-based activation
3. target - group of units
4. device - expose kernel devices
5. moint - control file system mount points
6. automount - allows parallelized boot and on-demand filesystem mounting
7. timer - trigger other units based on timers – replace cron
8. swap - encapsulate memory swap
9. path - activate a service when an object is changed on file system
10. slice - group units to manage processes in a hierarchical tree (for resource management)
11. scope - manage foreign processes (no starting)

Unit Types

1. service - start and control daemons and their processes
2. socket - deal with IPC and networking sockets and socket-based activation
3. target - group of units
4. device - expose kernel devices
5. moint - control file system mount points
6. automount - allows parallelized boot and on-demand filesystem mounting
7. timer - trigger other units based on timers – replace cron
8. swap - encapsulate memory swap
9. path - activate a service when an object is changed on file system
10. slice - group units to manage processes in a hierarchical tree (for resource management)
11. scope - manage foreign processes (no starting)

Unit Types

1. service - start and control daemons and their processes
2. socket - deal with IPC and networking sockets and socket-based activation
3. target - group of units
4. device - expose kernel devices
5. moint - control file system mount points
6. automount - allows parallelized boot and on-demand filesystem mounting
7. timer - trigger other units based on timers – replace cron
8. swap - encapsulate memory swap
9. path - activate a service when an object is changed on file system
10. slice - group units to manage processes in a hierarchical tree (for resource management)
11. scope - manage foreign processes (no starting)

Usage

Daemon Management

With `systemctl`, units can be manipulated

`systemctl status myservice` will tell you the state of myservice

Units can be started and stopped for the current session

Replace ”status” with ”start” or ”stop”

To start a unit on every boot or prevent a unit from starting

Replace with ”enable” or ”disable”

Daemon Management

With `systemctl`, units can be manipulated

`systemctl status myservice` will tell you the state of myservice

Units can be started and stopped for the current session

Replace ”status” with ”start” or ”stop”

To start a unit on every boot or prevent a unit from starting

Replace with ”enable” or ”disable”

Daemon Management

With `systemctl`, units can be manipulated

`systemctl status myservice` will tell you the state of myservice

Units can be started and stopped for the current session

Replace ”status” with ”start” or ”stop”

To start a unit on every boot or prevent a unit from starting

Replace with ”enable” or ”disable”

Daemon Management

With `systemctl`, units can be manipulated

`systemctl status myservice` will tell you the state of myservice

Units can be started and stopped for the current session

Replace ”status” with ”start” or ”stop”

To start a unit on every boot or prevent a unit from starting

Replace with ”enable” or ”disable”

journalctl

Read the logs!

`systemd` uses append only logging. Logs are persistent by default

`journalctl -b` returns the current boot

`journalctl -b -1` returns the previous boot

`journalctl -D /mnt/var/log/journal -xe` allows you to read logs from another system, mounted at /mnt

logging options set by environment variables

journalctl

Read the logs!

`systemd` uses append only logging. Logs are persistent by default

`journalctl -b` returns the current boot

`journalctl -b -1` returns the previous boot

`journalctl -D /mnt/var/log/journal -xe` allows you to read logs from another system, mounted at /mnt

logging options set by environment variables

`journalctl`

Read the logs!

`systemd` uses append only logging. Logs are persistent by default

`journalctl -b` returns the current boot

`journalctl -b -1` returns the previous boot

`journalctl -D /mnt/var/log/journal -xe` allows you to read logs from another system, mounted at /mnt

logging options set by environment variables

`journalctl`

Read the logs!

`systemd` uses append only logging. Logs are persistent by default

`journalctl -b` returns the current boot

`journalctl -b -1` returns the previous boot

`journalctl -D /mnt/var/log/journal -xe` allows you to read logs from another system, mounted at /mnt

logging options set by environment variables

`journalctl`

Read the logs!

`systemd` uses append only logging. Logs are persistent by default

`journalctl -b` returns the current boot

`journalctl -b -1` returns the previous boot

`journalctl -D /mnt/var/log/journal -xe` allows you to read logs from another system, mounted at /mnt

logging options set by environment variables

`journalctl`

Read the logs!

`systemd` uses append only logging. Logs are persistent by default

`journalctl -b` returns the current boot

`journalctl -b -1` returns the previous boot

`journalctl -D /mnt/var/log/journal -xe` allows you to read logs from another system, mounted at /mnt

logging options set by environment variables

Power Management

`systemd` provides power management targets.
They are invoked with `systemctl`

- poweroff
- reboot
- suspend
- hibernate
- hybrid-sleep

Power Management

`systemd` provides power management targets.
They are invoked with `systemctl`

- poweroff
- reboot
- suspend
- hibernate
- hybrid-sleep

Timers

Realtime and monotonic timers are supported

Archwiki has examples!

`systemd-run` allows arbitrary commands to be ran after a timer

As a cron replacement:

Pros:

+ logging

+ systemd benefits

Cons:

- more complicated setup

- MAILTO functionality missing. Can be shoehorned in

Programs exist to translate from crontabs to `systemd` timers

Timers

Realtime and monotonic timers are supported

Archwiki has examples!

`systemd-run` allows arbitrary commands to be ran after a timer

As a cron replacement:

Pros:

+ logging

+ systemd benefits

Cons:

- more complicated setup

- MAILTO functionality missing. Can be shoehorned in

Programs exist to translate from crontabs to `systemd` timers

Timers

Realtime and monotonic timers are supported

Archwiki has examples!

`systemd-run` allows arbitrary commands to be ran after a timer

As a cron replacement:

Pros:

+ logging

+ systemd benefits

Cons:

- more complicated setup

- MAILTO functionality missing. Can be shoehorned in

Programs exist to translate from crontabs to `systemd` timers

Timers

Realtime and monotonic timers are supported

Archwiki has examples!

`systemd-run` allows arbitrary commands to be ran after a timer

As a cron replacement:

Pros:

+ logging

+ systemd benefits

Cons:

- more complicated setup

- MAILTO functionality missing. Can be shoehorned in

Programs exist to translate from crontabs to `systemd` timers

Timers

Realtime and monotonic timers are supported

Archwiki has examples!

`systemd-run` allows arbitrary commands to be ran after a timer

As a cron replacement:

Pros:

+ logging

+ systemd benefits

Cons:

- more complicated setup

- MAILTO functionality missing. Can be shoehorned in

Programs exist to translate from crontabs to `systemd` timers

Timers

Realtime and monotonic timers are supported

Archwiki has examples!

`systemd-run` allows arbitrary commands to be ran after a timer

As a cron replacement:

Pros:

+ logging

+ systemd benefits

Cons:

- more complicated setup

- MAILTO functionality missing. Can be shoehorned in

Programs exist to translate from crontabs to `systemd` timers

Drama

Debian:

In 2014, the Debian mailing list was subjected to heated discussion about whether or not to change over to `systemd`

Four developers, including the systemd package maintainer quit because of the stress

Lennart Poettering is a dick. Allegedly

Community and developers don't get along well

Significantly different from previous systems in Linux

Change is scary

Large repository pulling in many smaller projects

e.g. gummiboot was taken in to become systemd boot

Drama

Debian:

In 2014, the Debian mailing list was subjected to heated discussion about whether or not to change over to `systemd`

Four developers, including the systemd package maintainer quit because of the stress

Lennart Poettering is a dick. Allegedly

Community and developers don't get along well

Significantly different from previous systems in Linux

Change is scary

Large repository pulling in many smaller projects

e.g. gummiboot was taken in to become systemd boot

Drama

Debian:

In 2014, the Debian mailing list was subjected to heated discussion about whether or not to change over to `systemd`

Four developers, including the systemd package maintainer quit because of the stress

Lennart Poettering is a dick. Allegedly

Community and developers don't get along well

Significantly different from previous systems in Linux

Change is scary

Large repository pulling in many smaller projects

e.g. gummiboot was taken in to become systemd boot

Drama

Debian:

In 2014, the Debian mailing list was subjected to heated discussion about whether or not to change over to `systemd`

Four developers, including the systemd package maintainer quit because of the stress

Lennart Poettering is a dick. Allegedly

Community and developers don't get along well

Significantly different from previous systems in Linux

Change is scary

Large repository pulling in many smaller projects

e.g. gummiboot was taken in to become systemd boot

Drama

Debian:

In 2014, the Debian mailing list was subjected to heated discussion about whether or not to change over to `systemd`

Four developers, including the systemd package maintainer quit because of the stress

Lennart Poettering is a dick. Allegedly

Community and developers don't get along well

Significantly different from previous systems in Linux

Change is scary

Large repository pulling in many smaller projects

e.g. gummiboot was taken in to become systemd boot

Drama

Debian:

In 2014, the Debian mailing list was subjected to heated discussion about whether or not to change over to `systemd`

Four developers, including the systemd package maintainer quit because of the stress

Lennart Poettering is a dick. Allegedly

Community and developers don't get along well

Significantly different from previous systems in Linux

Change is scary

Large repository pulling in many smaller projects

e.g. gummiboot was taken in to become systemd boot

Drama

Debian:

In 2014, the Debian mailing list was subjected to heated discussion about whether or not to change over to `systemd`

Four developers, including the systemd package maintainer quit because of the stress

Lennart Poettering is a dick. Allegedly

Community and developers don't get along well

Significantly different from previous systems in Linux

Change is scary

Large repository pulling in many smaller projects

e.g. gummiboot was taken in to become systemd boot

Drama

Debian:

In 2014, the Debian mailing list was subjected to heated discussion about whether or not to change over to `systemd`

Four developers, including the systemd package maintainer quit because of the stress

Lennart Poettering is a dick. Allegedly

Community and developers don't get along well

Significantly different from previous systems in Linux

Change is scary

Large repository pulling in many smaller projects

e.g. gummiboot was taken in to become systemd boot

Questions