Lesson #2 Managing and troubleshooting system boot and services

Advanced Linux Administration

Aleš Zelinka

Introduction

38.5 min

Anatomy of Linux boot

Boot is a series of steps in which small and simple programs start increasingly bigger and more complex programs, ending in full featured operating system.

Q: What are the major phases of booting to Linux OS?

Anatomy of Linux boot

Boot is a series of steps in which small and simple programs start increasingly bigger and more complex programs, ending in full featured operating system.

1. hardware
2. firmware (BIOS/UEFI)
3. bootloader
4. OS kernel (linux) + root filesystem from initial ramdisk
5. init process (systemd)
6. userspace
7. user sessions

Anatomy of Linux boot

Demo on a virt machine

1. hardware
2. firmware (BIOS/UEFI)
    - Not interesting … to us … now
3. Bootloader
4. OS kernel (linux) + root filesystem from initial ramdisk
5. init process (systemd)
6. userspace
7. user sessions

Anatomy of Linux boot

1. hardware
2. firmware (BIOS/UEFI)
3. bootloader
   - The goal of the bootloader is to let user start one of the installed OSes or start the default one.
   - **grub2**, bootmgr (windows), elilo, zipl,...
4. OS kernel (linux) + root filesystem from initial ramdisk
5. init process (systemd)
6. userspace
7. user sessions

Anatomy of Linux boot

1. hardware
2. firmware (BIOS/UEFI)
3. bootloader
4. OS kernel (linux) + root filesystem from initial ramdisk
   - Kernel (The Linux) extracts the initrd archive to memory and runs whatever is inside it
   - The Initial ramdisk is a specialized mini-OS with single purpose: make the real root filesystem available ~ mount it
   - Demo: `$ lsinitrd`
   - Hint: press ESC to show logs during boot
5. init process (systemd)
6. userspace
7. user sessions

Anatomy of Linux boot

1. hardware
2. firmware (BIOS/UEFI)
3. bootloader
4. OS kernel (linux) + root filesystem from initial ramdisk
5. init process (systemd)
   - Process (pid) 1, started by kernel. The so called init process, parent of all other userland processes, itself with no parent
   - Responsible for starting everything that comprise an OS and keeping that running
   - Demo: $ `pstree -p`
   - Systemd is the most used implementation of an init system, there are alternatives
   - **Q**: Why is it called systemd?
6. userspace
7. user sessions

Anatomy of Linux boot

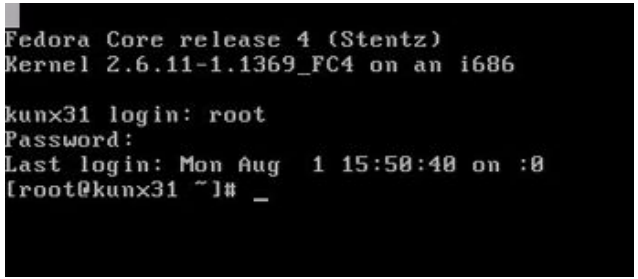
1. hardware
2. firmware (BIOS/UEFI)
3. bootloader
4. OS kernel (linux) + root filesystem from initial ramdisk
5. init process (systemd)
6. Userspace
   - Userspace is a term for anything that is not kernel, userland processes run in user mode = have restricted access to HW and memory (mostly through kernel).
   - Services (daemons) get started automatically (after boot / on demand)
   - **Q**: What is a service?
7. user sessions

Anatomy of Linux boot

1. hardware
2. firmware (BIOS/UEFI)
3. bootloader
4. OS kernel (linux) + root filesystem from initial ramdisk
5. init process (systemd)
6. Userspace
   - Userspace is a term for anything that is not kernel, userland processes run in user mode = have restricted access to HW and memory (mostly through kernel).
   - Services (daemons) get started
   - **Q**: What is a service?
   - **A**: A background processes, not under the direct control of an interactive user, usually serving requests repeatedly
     - E.g. web server, mail server, automatic updates,...
7. user sessions

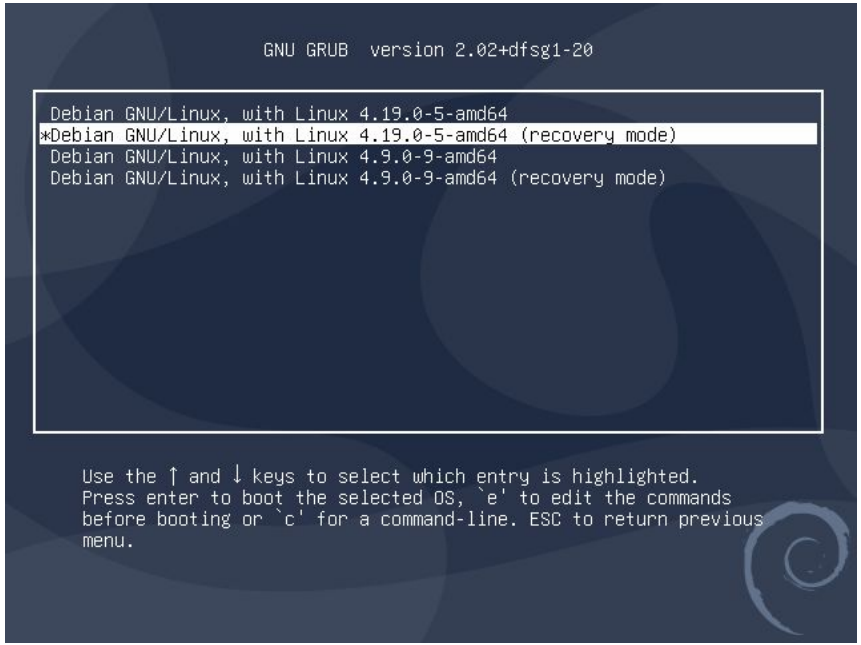
Anatomy of Linux boot

1. hardware
2. firmware (BIOS/UEFI)
3. bootloader
4. OS kernel (linux) + root filesystem from initial ramdisk
5. init process (systemd)
6. Userspace
7. user sessions - processes started as a consequence of user login
   - console or graphical
   - Interactive
   - `$ ps au`



Bootloader: grub2

Grub can be configured from within interactively, allows you to use a shell-like scripting language to control what it does.



Bootloader: grub2

```
load_video
set gfx_payload=keep
insmod gzio
linux ($root)/vmlinuz-5.0.9-301.fc30.x86_64 root=/dev/mapper/fedora_localhost-\
-live-root ro resume=/dev/mapper/fedora_localhost--live-swap rd.lvm.lv=fedora_\
localhost-live/root rd.lvm.lv=fedora_localhost-live/swap rhgb quiet
initrd ($root)/initramfs-5.0.9-301.fc30.x86_64.img



      Press Ctrl-x to start, Ctrl-c for a command prompt or Escape to
      discard edits and return to the menu. Pressing Tab lists
      possible completions.
```

Bootloader: grub2

```
linux <path_to_kernel> kernel_parameters ...

initrd <path_to_ramdisk>
```

Few notable kernel parameters

- For debugging remove kernel parameters ‘rhgb’/’splash’ and ‘quiet’
- To boot into simple singleuser system with filesystems and services:
  - `systemd.unit=rescue.target` (`rescue`, `single` or 1)
- To boot to a very limited system without filesystems and services (and RO /)
  - `systemd.unit=emergency.target` (or simply `emergency`)
- **Q**: Why kernel parameter 1 means rescue mode?

Systemd: terminology

- Unit = any resource that the system knows how to operate on and manage
  - `$ man systemd.unit`
  - A unit can depend on another unit, systemd makes sure all are properly started and in the correct order
- Service = service-type unit, e.g. httpd.service for web server
  - `$ man systemd.service`
- Target = target-type unit, group of other units and a synchronization point
  - e.g. graphical.target represents all units that have to run to have graphical desktop
  - `$ man systemd.target`

Systemd: systemctl: Managing services

- `$ systemctl` status
- `$ systemctl [start|stop] some_service`
  - Will update runtime configuration
- `$ systemctl [enable|disable|mask] some_service`
  - Will change behaviour persistently, e.g. after boot

```
zelial@chuchvalec:~
File  Edit  View  Search  Terminal  Help
[chuch ~]$ sudo systemctl status cups
● cups.service - CUPS Scheduler
   Loaded: loaded (/usr/lib/systemd/system/cups.service; enabled; vendor preset: disabled)
   Active: active (running) since Wed 2019-09-18 19:43:12 CEST; 40min ago
     Docs: man:cupsd(8)
 Main PID: 1070 (cupsd)
   Status: "Scheduler is running..."
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/cups.service
           └─1070 /usr/sbin/cupsd -l

Sep 18 19:43:12 chuchvalec cupsd[1070]: Loading from cache...
Sep 18 19:43:12 chuchvalec systemd[1]: Started CUPS Scheduler.
Sep 18 19:43:12 chuchvalec cupsd[1070]: REQUEST localhost - - "POST / HTTP/1.1" 200 349 Create-Printer-Subscription
Sep 18 19:43:13 chuchvalec cupsd[1070]: REQUEST localhost - - "POST / HTTP/1.1" 200 176 Create-Printer-Subscription
Sep 18 19:43:26 chuchvalec cupsd[1070]: REQUEST localhost - - "POST / HTTP/1.1" 200 362 Create-Printer-Subscription
Sep 18 20:22:37 chuchvalec cupsd[1070]: REQUEST localhost - - "POST / HTTP/1.1" 401 123 Cancel-Subscription success
Sep 18 20:22:37 chuchvalec cupsd[1070]: REQUEST localhost - root "POST / HTTP/1.1" 200 123 Cancel-Subscription succ
Sep 18 20:22:37 chuchvalec cupsd[1070]: REQUEST localhost - - "POST / HTTP/1.1" 200 152 Cancel-Subscription success
Sep 18 20:22:37 chuchvalec cupsd[1070]: REQUEST localhost - - "POST / HTTP/1.1" 200 349 Create-Printer-Subscription
Sep 18 20:22:37 chuchvalec cupsd[1070]: REQUEST localhost - - "POST / HTTP/1.1" 200 176 Create-Printer-Subscription
lines 1-20/20 (END)
```

Systemd: systemctl: Managing targets

Target = target-type unit, group of other units and a synchronization point

- `$ systemctl start unit.target`
  - will start all enabled, unmasked dependencies
- `$ man systemd.special` for list of special-purpose targets
  - basic.target, multi-user.target graphical.target
  - reboot.target, poweroff.target, suspend.target

```
[chuch ~]$ systemctl list-dependencies graphical.target
graphical.target
● ├accounts-daemon.service
● ├gdm.service
● ├rtkit-daemon.service
● ├switcheroo-control.service
● ├systemd-readahead-collect.service
● ├systemd-readahead-replay.service
● ├systemd-update-utmp-runlevel.service
● ├udisks2.service
● └multi-user.target
●   ├abrt-journal-core.service
●   ├abrt-oops.service
●   ├abrt-vmcore.service
●   ├abrt-xorg.service
●   ├atd.service
●   ├auditd.service
●   ├avahi-daemon.service
●   ├chronyd.service
●   ├crond.service
●   ├cups-browsed.service
●   ├cups.path
●   ├dbus.service
●   ├dbxtool.service
●   ├ipmievd.service
●   ├irqbalance.service
●   ├libvirtd.service
```

Systemd: systemctl: Managing targets

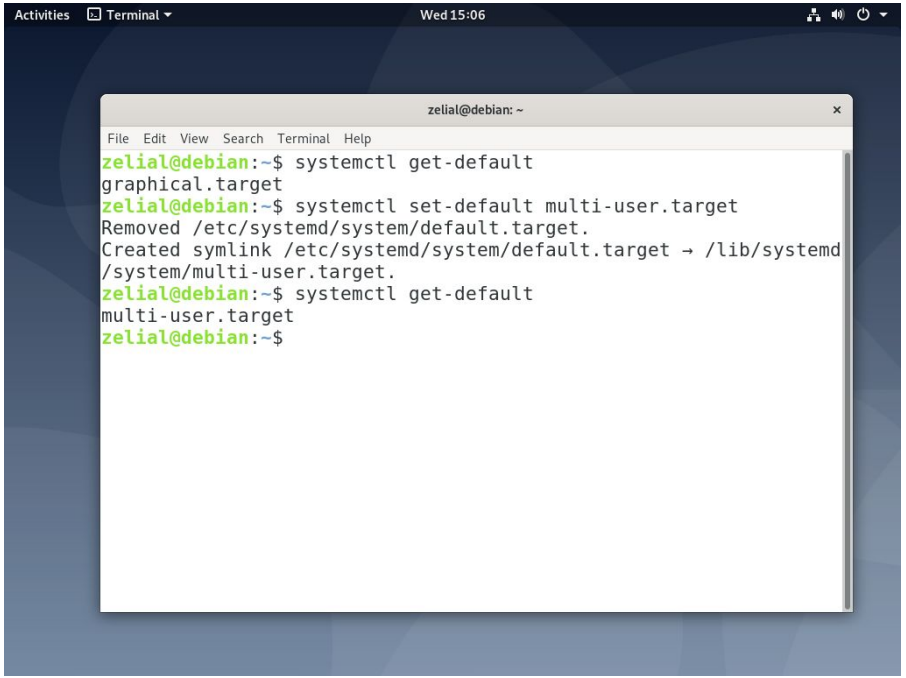
Default target - target the OS will boot to

```
$ systemctl [get-default|set-default]
```

Q: what will happen if you do this by mistake?

$ systemctl set-default poweroff.target

Q: How to fix it?



Logging

Logging is essential to any system administrator, through logs the admin can see what is/was happening.

Everything produces logs

- Kernel
- Systemd
- Services
- Applications

Logs are sensitive, available to root only.

Logging: files

Traditionally logs are stored as text files in /var/log/

```
$ less /var/log/some_log_file
```

- Browse through the log file

```
$ tail /var/log/some_log_file
```

- See latest log messages

```
$ tail -F /var/log/some_log_file
```

- Continuously show log messages as they are being logged

```
[chuch ~]$ ls -al /var/log/
total 39132
drwxr-xr-x. 25 root    root              12288 Sep 18 20:14 .
drwxr-xr-x. 23 root    root               4096 Sep 27  2018 ..
drwxr-xr-x.  2 root    root               4096 Jun  9  2014 anaconda
drwxr-xr-x.  2 root    root               4096 Jan  2  2018 atop
drwx------.  2 root    root               4096 Jul 26 11:56 audit
drwxr-xr-x.  2 root    root               4096 Sep 19  2017 blivet-gui
-rw-r--r--.  1 root    root               6198 Sep 18 19:43 boot.log
-rw-------.  1 root    utmp                  0 Sep 18 20:14 btmp
-rw-------.  1 root    utmp                  0 Sep  2 20:25 btmp-20190918
drwxr-xr-x.  2 chrony  chrony             4096 Apr  4  2018 chrony
drwxr-xr-x.  2 root    root               4096 Apr 12  2018 cluster
-rw-r--r--.  1 root    root               1245 Sep 18 21:01 cron
-rw-r--r--.  1 root    root               3174 Jun 18  2014 cron-20140618.gz
-rw-r--r--.  1 root    root               1708 Jun 23  2014 cron-20140623.gz
-rw-r--r--.  1 root    root               3502 Jun 30  2014 cron-20140630.gz
-rw-r--r--.  1 root    root               3191 Jul  8  2014 cron-20140708.gz
-rw-r--r--.  1 root    root               2649 Jul 14  2014 cron-20140714.gz
```

Logging: journal

Systemd stores logs in structured and indexed binary database.

```
$ journalctl -f
```

- Continuously show log messages as they are being logged

```
$ journalctl -u service
```

- Show logs from a particular service(unit)

```
$ journalctl -k
```

- Show only kernel log messages

```
$ journalctl /usr/bin/some_daemon
```

- Show messages logged by a particular program

```
$ journalctl --since="2019-09-17 21:30:00" --until="2019-09-17 23:00:00"
```

- Messages logged during a specific timeframe

```
$ journalctl --since=yesterday
$ journalctl --since=-2hours
```

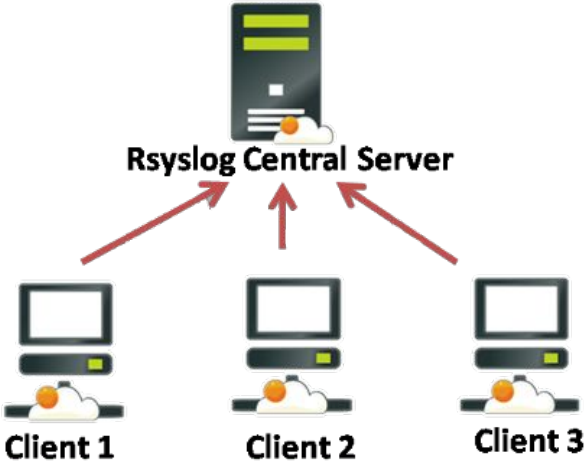
- Relative time specification: messages logged since yesterday; since 2 hours ago

Logging: rsyslog

Service with advanced logging features.

Not installed on all systems by default (`$ dnf install rsyslog`)

- send/receive logs over network to aggregate logs from many computers at one place
- Runs on _many_ OSes, not just Linux
- Creates `/var/log/messages` on Fedora which is a nice to have catch-all log file :))



Workshop

52.25 min

Lab 1 - Verbose boot [10 min]

- Reboot, enter interactive grub mode
- Enable verbose logging
- Boot, observe how boot progresses and is logged to screen
- Log in and using `journalctl` find the very first boot log message after the reboot

Lab 2 - troubleshooting boot [15 min]

- Boot into rescue mode via interactive grub, editing kernel parameters
  - `rescue.target`
- Login as root, count running processes
- Boot into emergency mode via interactive grub, editing kernel parameters
  - `emergency.target`
- Login as root, count running processes

Hints:

- `ps x |wc -l`
- To change anything on /, remount it read-write
  - `mount -o rw,remount /`

Lab 3 - Default target [15 min]

- Boot and login into your VM
- Set default target to `multi-user.target`
- Reboot, login
- Set default target back to `graphical.target`
- Reboot

Lab 4 - Logging a message [10 min]

- Login as root to be able to see logs
- Log a message “Hi from Lab4”
- Find the message in text log file `/var/log/messages`
- Find the message in journal

Hint: `$ man logger`

Lab 5 - failed service [10 min]

- List all failed services
- Show logs from one of the failed services
- If you have no failing service in your system
  - Congrats ;)
  - List logs of rsyslog.service

Hint: search ‘failed’ in `$ man systemctl`

Links and Resources

- Man bootup (7)
- Kernel command line parameters
- Systemd manual pages
  - systemctl
  - journalctl