

CS 447/647

init

Overview

What is init?

What are units?

Exercises

Linux SysOps Handbook

A study notes book for the common knowledge and tasks of a Linux system admin.

[GitBook](#).

▲ [rhythmshifter03](#) 3 hours ago | [prev](#) | [next](#) [-]

This is amazingly useful to a person learning Linux in a corporate environment where they've just been thrown to the wolves to figure it out. Thank you!!

[reply](#)

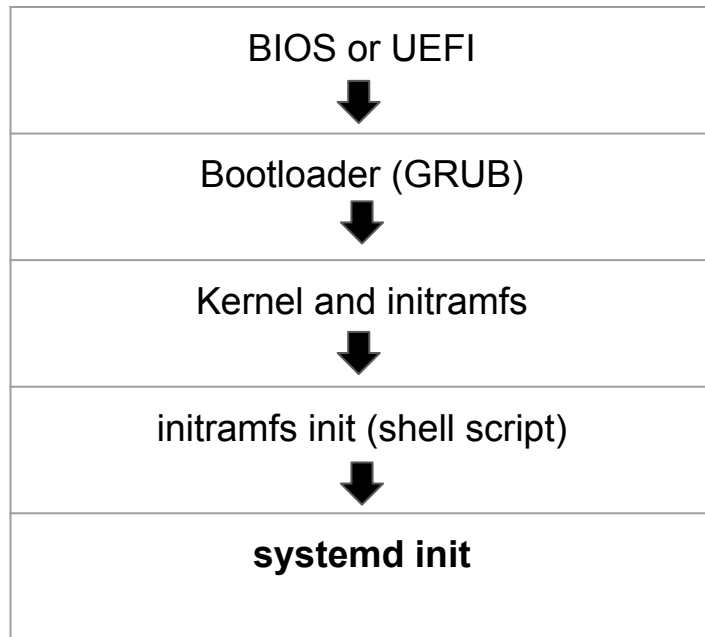
▲ [sp33der89](#) 3 hours ago | [parent](#) | [next](#) [-]

As somebody who's in this exact same position(well not really corporate), I feel the same way about this guide.

Thanks so much the author!

[reply](#)

Virtual Machine Boot (Recap)



What is init?

init

- Short for initialization
- First process to run
 - PID 1
 - Query processes with the ps(1) command: `ps -q 1`
 - `/sbin/init > /lib/systemd/systemd`
- Three types
 - SysV - System 5, Just a bunch of shell scripts (Legacy). 1983
 - BSD - Just like Sys5 but for the Berkeley Software Distribution
 - **systemd - Replaces SysV. used on nearly all modern Linux distributions**

Why you need to know SysV

- Legacy systems
 - CentOS 5 and older
- Debian used it before 2014
 - Ubuntu is downstream of Debian
- “The only reason why we still use it today is the cost of a migration.”

SysV init - Runlevels

Modes of operation

0 Shutdown

1 Single user mode - AKA Recovery

2-5 Normal multi-user mode - Most things run here, Networking, Graphics

6 Reboot


```
#!/bin/sh
# Start/stop the cron daemon.
#
### BEGIN INIT INFO
# Provides:          cron
# Required-Start:    $remote_fs $syslog $time
# Required-Stop:     $remote_fs $syslog $time
# Should-Start:      $network $named slapd autofs ypbind nsd nsld winbind
# Should-Stop:       $network $named slapd autofs ypbind nsd nsld winbind
# Default-Start:     2 3 4 5
# Default-Stop:
# Short-Description: Regular background program processing daemon
# Description:       cron is a standard UNIX program that runs user-specified
#                    programs at periodic scheduled times. vixie cron adds a
#                    number of features to the basic UNIX cron, including better
#                    security and more powerful configuration options.
### END INIT INFO

PATH=/bin:/usr/bin:/sbin:/usr/sbin
DESC="cron daemon"
NAME=cron
DAEMON=/usr/sbin/cron
PIDFILE=/var/run/crond.pid
SCRIPTNAME=/etc/init.d/"$NAME"

test -f $DAEMON || exit 0

. /lib/lsb/init-functions

[ -r /etc/default/cron ] && . /etc/default/cron
```

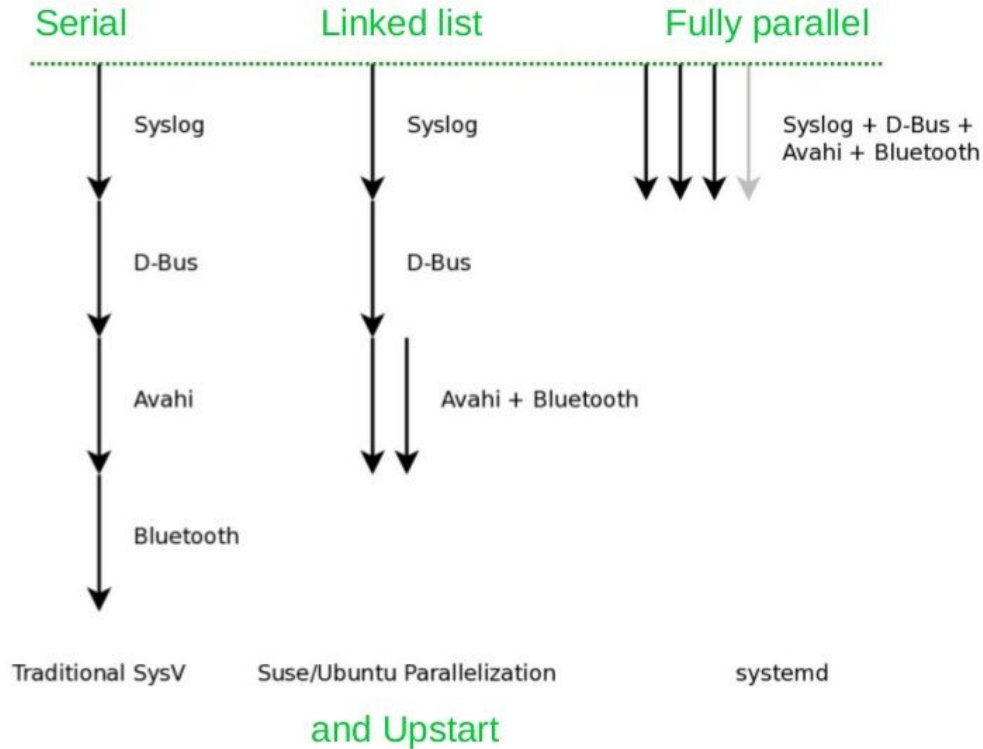
SysV init issues

- Not parallel
- Issues with dependencies
- Different conventions\code
- Difficult to maintain

systemd init

- Standardize system services management
- Collection of programs, daemons, libraries, technologies and kernel components
 - systemctl - Manages units
 - journalctl - Logging
 - networkd - Network Configuration
 - loginctl - Login manager
 - hostnamectl - Control the system hostname
- In short, it starts stuff.

Socket-based activation is key to systemd's fast boot



systemd.unit

- Unit - entity managed by systemd
 - **Service** - Most common
 - **Socket** - Interprocess Communication (IPC)
 - **Timer** - Time-based process
 - **Mount** - Filesystem mounting
 - **Target** - Group of Units
 - Scope - Group of processes
 - Slice - Resources for a group of processes
 - Path - A path monitored by systemd
 - Whatever else Lennart Poettering implements

`systemctl --type help`

`systemctl list-units` #Shows all units

Managing a systemd unit

vsftpd - lightweight, efficient FTP server written for security

1. Type **apt install -y vsftpd** to Install the Very Secure FTP Server
2. Type **systemctl start vsftpd** to activate the FTP server on your machine.
3. Type **systemctl status vsftpd**. You'll get output where you can see that the vsftpd service is currently operational.
4. Type **systemctl disable vsftpd** to stop the service from starting at boot.
5. Type **systemctl enable vsftpd** to automatically start the service after a restart.

systemd.unit

systemctl list-unit-files

UNIT FILE	STATE
proc-sys-fs-binfmt_misc.automount	static
-.mount	generated
data.mount	generated
dev-hugepages.mount	static
dev-mqueue.mount	static
proc-sys-fs-binfmt_misc.mount	static
sys-fs-fuse-connections.mount	static
sys-kernel-config.mount	static
sys-kernel-debug.mount	static
acpid.path	enabled
systemd-ask-password-console.path	static
systemd-ask-password-wall.path	static
watch-log.path	linked
session-376.scope	transient
session-38.scope	transient
acpid.service	disabled
apt-daily-upgrade.service	static
apt-daily.service	static
atftpd.service	generated
autovt@.service	enabled

systemd.unit states

State	Meaning
bad	Some kind of problem within systemd ; usually a bad unit file
disabled	Present, but not configured to start autonomously
enabled	Installed and runnable; will start autonomously
indirect	Disabled, but has peers in Also clauses that may be enabled
linked	Unit file available through a symlink
masked	Banished from the systemd world from a logical perspective
static	Depended upon by another unit; has no install requirements

systemd.target

- Similar to runlevels
- Groups units together
- Important targets
 - `multi-user.target`: day-to-day use server
 - `graphical.target`: day-to-day use desktop
 - `emergency.target`: used for recovery
 - `rescue.target`: used for single-user mode

`systemctl --type target` #Show all targets

`systemctl list-dependencies multi-user.target` #Show dependencies

`systemctl get-default` #Display the default target

`systemctl set-default multi-user.target` #Set the default target

`systemctl isolate reboot.target` #Start these units and stop others

Run level	Target	Description
0	poweroff.target	System halt
emergency	emergency.target	Bare-bones shell for system recovery
1, s, single	rescue.target	Single-user mode
2	multi-user.target ^a	Multiuser mode (command line)
3	multi-user.target ^a	Multiuser mode with networking
4	multi-user.target ^a	Not normally used by init
5	graphical.target	Multiuser mode with networking and GUI
6	reboot.target	System reboot

a. By default, **multi-user.target** maps to **runlevel3.target**, multiuser mode with networking.

Unit locations

- **/usr/lib/systemd/system**
 - a. Contains default unit files that have been installed from packages. You should never edit these files directly.
- **/etc/systemd/system** contains custom unit files.
 - a. It may also contain files that have been written by an administrator or generated by the **systemctl edit** command.
- **/run/systemd/system**
 - a. Contains unit files that have automatically been generated.
- **SYSTEMD_UNIT_PATH env variable**
 - a. Has to exist before /sbin/init is executed.
 - b. **\$SYSTEMD_UNIT_PATH** ends with an empty component (":"), the usual unit load path will be appended to the contents of the variable.


```
1  [Unit]
2  Description=The Apache HTTP Server
3  After=network.target remote-fs.target nss-lookup.target
4
5  [Service]
6  Type=forking
7  Environment=APACHE_STARTED_BY_SYSTEMD=true
8  ExecStart=/usr/sbin/apachectl start
9  ExecStop=/usr/sbin/apachectl stop
10 ExecReload=/usr/sbin/apachectl graceful
11 PrivateTmp=true
12 Restart=on-abort
13
14 [Install]
15 WantedBy=multi-user.target
```

man 5 systemd.service

Apache2 Ubuntu Default Page

localhost

Search



ubuntu

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/  
|-- apache2.conf  
|   |-- ports.conf  
|-- mods-enabled  
|   |-- *.load  
|   |-- *.conf  
|-- conf-enabled  
|   |-- *.conf  
|-- sites-enabled  
|   |-- *.conf
```

Changing a unit configuration

1. Type **apt install nginx-extras** to install the nginx web server package.
2. Type **systemctl cat nginx.service** to show the current configuration of the unit file that starts the nginx web server.
3. Type **systemctl show nginx.service** to get an overview of available configuration options for this unit file.
4. Type **systemctl edit nginx.service** to change the default configuration, and ensure that the **[Service]** section includes the lines **Restart=always** and **RestartSec=5s**.
5. Type **systemctl daemon-reload** to ensure that systemd picks up the new configuration.
6. Type **systemctl restart nginx** to restart the nginx service.
7. Type **systemctl status nginx** and then repeat after 5 seconds. You'll notice that the nginx process gets automatically restarted.

```
export SYSTEMD_EDITOR="/usr/bin/vim" #Changes the editor
```

Testing “Restart=always”

1. View the process

```
ps ax | grep nginx
```

```
#or
```

```
ps -fp $(pgrep -d, -x nginx) #pgrep finds the process by name
```

2. Kill the process with a signal

```
pkill -SIGTERM -f nginx
```

3. Wait 5 seconds... then check if it's running

```
ps ax | grep nginx
```

4. Check status

```
systemctl status nginx
```

service

```
systemctl --type=service --all
```

UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
apparmor.service	not-found	inactive	dead	apparmor.service
apt-daily-upgrade.service	loaded	inactive	dead	Daily apt upgrade and clean acti
apt-daily.service	loaded	inactive	dead	Daily apt download activities
auditd.service	not-found	inactive	dead	auditd.service
clamav-daemon.service	not-found	inactive	dead	clamav-daemon.service
console-screen.service	not-found	inactive	dead	console-screen.service
console-setup.service	loaded	active	exited	Set console font and keymap
cron.service	loaded	active	running	Regular background program proce
dbus.service	loaded	active	running	D-Bus System Message Bus
display-manager.service	not-found	inactive	dead	display-manager.service
emergency.service	loaded	inactive	dead	Emergency Shell
exim4.service	loaded	active	running	LSB: exim Mail Transport Agent
fail2ban.service	loaded	active	running	Fail2Ban Service
firewalld.service	not-found	inactive	dead	firewalld.service
getty-static.service	loaded	inactive	dead	getty on tty2-tty6 if dbus and 1
getty@tty1.service	loaded	active	running	Getty on tty1

Minecraft .service

```
apt install unzip
mkdir /srv/minecraft
cd /srv/minecraft/
wget https://minecraft.azureedge.net/bin-linux/bedrock-server-1.16.40.02.zip
unzip bedrock-server-1.16.40.02.zip
#Test it
./bedrock_server
```

.service file

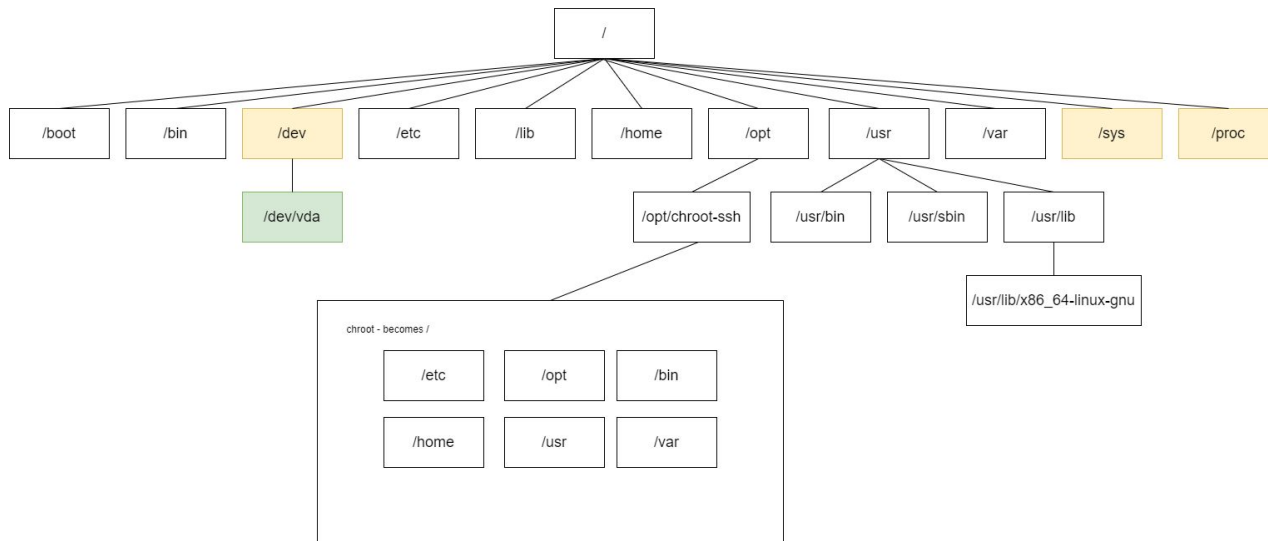
```
[Unit]
Description=Minecraft server
Requires=network.target local-fs.target
After=network.target local-fs.target

[Install]
WantedBy=multi-user.target

[Service]
Type=oneshot
KillMode=none
User=root
ExecStart=/usr/bin/tmux new-session -d -s minecraft-server -n minecraft-server -c /srv/minecraft '/srv/minecraft/bedrock_server'
ExecStop=/usr/bin/tmux send-keys -t minecraft-server:minecraft-server s t o p Enter
WorkingDirectory=/srv/minecraft/
RemainAfterExit=yes
```

chroot-ssh

- Create a separate SSH service
- Limit filesystem access
- Untrustworthy users
- Limit userland commands



chroot-ssh

```
lxc config device add c0 proxy22 proxy listen=tcp:0.0.0.0:2222  
connect=tcp:127.0.0.1:22
```

socket

- A unit file that ends in “.socket”
- Unix Socket or Network Socket
 - A unix socket is a file
 - Network socket is an Internet Protocol (IP) address and a port. 127.0.0.01:4444
 - Inter-Process Communication
- Needs three sections
 - [Unit]
 - [Install]
 - [Socket]

/etc/systemd/system/echo.socket

```
[Unit]
Description=Simple echo server

[Socket]
ListenStream=/run/echo
Accept=yes

[Install]
WantedBy=sockets.target
```

```
systemctl list-unit-files | grep socket
```

/etc/systemd/system/echo@.service

```
[Unit]
Description=Echo server service

[Service]
ExecStart=/usr/local/bin/echo.py
StandardInput=socket
```

systemctl list-unit-files | grep "echo@"

/usr/local/bin/echo.py

Create a file:

```
#!/usr/bin/env python3
import sys
data = sys.stdin.read()

output = "{0}\r\n".format(data.strip().upper())

sys.stdout.write(output)
```

```
apt install -y socat
chmod +x /usr/local/bin/echo.py
echo "test" | echo.py # Output: TEST
```


Let's use our echo.socket

1. `systemctl daemon-reload`
2. `systemctl list-unit-files | grep echo`
3. `systemctl start echo.socket`
4. `apt install -y socat` # Used for communicating with a Unix socket
5. `echo hello world | socat - unix-connect:/run/echo`

timer

- A unit file that ends in “.timer” that runs a service
- Built-in support for
 - Calendar Time, IE: OnCalendar=Mon *-*-* 00:00:00
 - Monotonic Time Events, IE: OnBootSec, OnStartupSec
 - Transient
 - `systemd-run --on-active=30 /bin/touch /tmp/foo`
- Needs three sections
 - [Unit]
 - [Install]
 - [Timer]

timer - OnCalendar Examples

DayOfWeek Year-Month-Day Hour:Minute:Second

minutely → `*-*-* *:*:00`

hourly → `*-*-* *:00:00`

daily → `*-*-* 00:00:00`

monthly → `*-*-01 00:00:00`

weekly → `Mon *-*-* 00:00:00`

yearly → `*-01-01 00:00:00`

quarterly → `*-01,04,07,10-01 00:00:00`

semiannually → `*-01,07-01 00:00:00`

`systemd-run -d --user --on-calendar '2022-02-22 23:59:00 PST' some-command`

timer - OnCalendar Examples

#First Saturday of each month

Sat *-*-1..7 18:00:00

#Monday through Friday 10:30PM

Mon..Fri 22:30

#Run the first 4 days of a Month only if Mon or Tue

Mon,Tue *-*-01..04 12:00:00

#Run these

systemd-analyze calendar '*-*-* 20:00:0'

systemd-analyze calendar 'Mon,Tue *-*-01..04 12:00:00'

/etc/systemd/system/backup@.service

[Unit]

Description=Performs a system backup

[Service]

Type=oneshot

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

/etc/systemd/system/backup@%i.timer

[Unit]

Description=Run a backup at %i interval

[Timer]

OnCalendar=%i

Persistent=true

[Install]

WantedBy=timers.target

`/usr/local/bin/backup.sh`

```
#!/bin/bash
```

```
DATE_STR=`date +%d_%m_%Y_%H_%M`  
echo "Backup Script ${DATE_STR}"
```

```
# Test in with bash..
```

```
chmod +x /usr/local/bin/backup.sh  
/usr/local/bin/backup.sh #Test it
```

timer

```
systemctl start backup@hourly.timer
```

```
systemctl status backup@hourly.timer
```

```
systemctl list-timers
```

```
systemctl start backup@hourly.service
```

```
systemctl status backup@hourly.service
```

#Transient Timer Example

```
systemd-run --on-active=30 /usr/local/bin/backup.sh
```


mount - Filesystem Mountpoint

`#/etc/systemd/system/cs447.mount`

`[Unit]`

`Description=Server CS447 directory`

`After=network.target`

`[Mount]`

`What=192.168.1.1:/cs447`

`Where=/cs447`

`Type=nfs`

`Options=_netdev,auto`

`[Install]`

`WantedBy=multi-user.target`

systemd logging

- `journal`
 - Handles logging from the kernel and all services from the early boot process (initramfs) to final shutdown
- Messages are stored in `/run`
 - `rsyslog` can forward them to `/var/log`

Common Commands

`SYSTEMD_LOG_LEVEL=debug` #Environmental variable for debugging

`journalctl -n 20` #Last 20 lines

`journalctl -u echo.socket`

`journalctl -f -u echo.socket` #Follow the log, similar to `tail -f`

`journalctl /usr/local/bin/echo.py` #All lines from `echo.py`

```
import logging
from systemd.journal import JournaldLogHandler

##### BEGIN LOGGING SETUP #####

# Get an instance of the logger object this module will use
logger = logging.getLogger(__name__)

# Instantiate the JournaldLogHandler to hook into systemd
journald_handler = JournaldLogHandler()

# Set a formatter to include the level name
journald_handler.setFormatter(logging.Formatter(
    '[%(levelname)s] %(message)s'
))

# Add the journald handler to the current logger
logger.addHandler(journald_handler)

# Set the logging level
logger.setLevel(logging.DEBUG)

##### END OF LOGGING SETUP #####

logger.info(__name__)
```

Exercise

Tar file inspecting socket

1. Pipe a tar file to the socket.
2. The socket sends the input to `list_tar.py`,
3. `list_tar.py` iterates over the `TarInfo` objects and prints their names.
4. Need `.socket` and `.service` files

```
cp /cs447/demos/systemd/list_tar-socket/list_tar.py.start ~/
```

```
cp /cs447/demos/systemd/list_tar-socket/troff_files.tar ~/
```

```
mv list_tar.py /usr/local/bin/list_tar.py
```

```
cat troff_files.tar | list_tar.py
```

```
cat troff_files.tar | socat - unix-connect:/run/list_tar #UDS
```