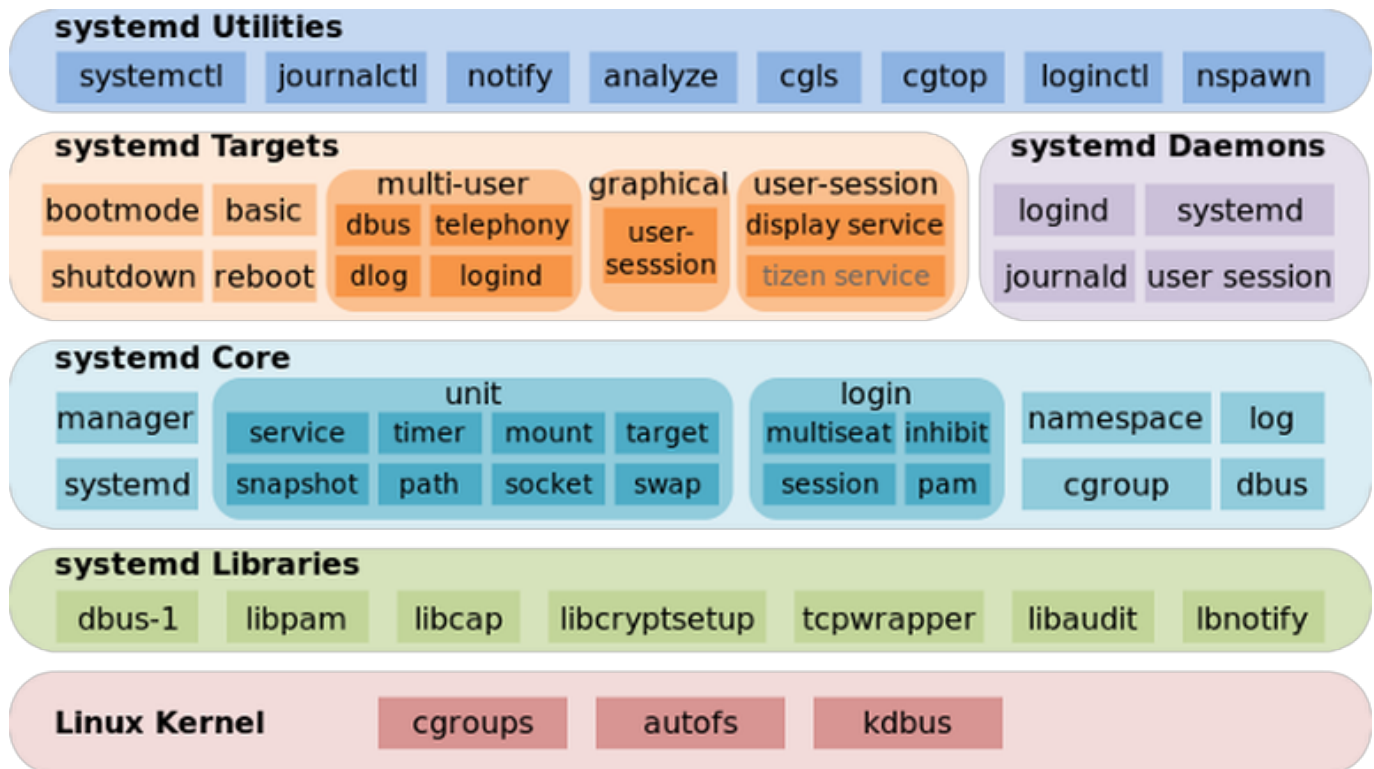


How to start / Stop and Analyze system services and improve Linux system boot time performance

Author : admin



This post is going to be a very short one and to walk through shortly to System V basic start / stop remove service old way and the new ways introduced over the last 10 years or so with the introduction of systemd on mass base across Linux distributions.

Finally I'll give you few hints on how to check (analyze) the boot time performance on a modern GNU / Linux system that is using systemd enabled services.

1. System V and the old days few classic used ways to stop / start / restart services (runlevels and common wrapper scripts)

The old fashioned days when Linux was using SystemV / e.g. no SystemD used way was to just go through all the running services with following the run script logic inside the runlevel the system was booting, e.g. to check runlevel and then optimize each and every run script via the respective location of the bash service init scripts:

```
root@noah:/home/hipo# /sbin/runlevel
N 5
```

Or on some RPM based distros like Fedora / RHEL / SUSE Enterprise Linux to use **chkconfig** command, e.g. list services:

```
~]# chkconfig --list
```

```
etworkManager 0:off 1:off 2:on 3:on 4:on 5:on 6:off
abrttd         0:off 1:off 2:off 3:on 4:off 5:on 6:off
acpid          0:off 1:off 2:on 3:on 4:on 5:on 6:off
anamon         0:off 1:off 2:off 3:off 4:off 5:off 6:off
atd            0:off 1:off 2:off 3:on 4:on 5:on 6:off
auditd         0:off 1:off 2:on 3:on 4:on 5:on 6:off
avahi-daemon   0:off 1:off 2:off 3:on 4:on 5:on 6:off
```

And to start stop the service into (default runlevel) or respective runlevel:

```
~]# chkconfig httpd on
```

```
~]# chkconfig --list httpd
httpd        0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

~]# chkconfig service_name on --level runlevels

Debian / Ubuntu and other .deb based distributions with System V (which executes scripts without single order but one by one) are not having natively chkconfig but instead are famous for **update-rc.d** *init script wrapper*, here is few basic use of it:

update-rc.d defaults
update-rc.d start 20 3 4 5
update-rc.d -f remove

Here defaults means default set boot runtime for system and numbers are just whether service is started or stopped for respective runlevels. To check what is your default one simply run **/sbin/runlevel**

Other useful tool to stop / start services and analyze what service is running and which not in real time (but without modifying boot time set for a service) - more universal nowadays is to use the **service** command.

```
root@noah:/home/hipo# service --status-all
[ + ] acpid
[ - ] alsa-utils
[ - ] anacron
[ + ] apache-htcacheclean
[ - ] apache2
[ + ] atd
[ + ] aumix
```

root@noah:/home/hipo# **service cron restart/usr/sbin/service** command is just a simple wrapper bash shell script that takes care about start / stop etc. operations of scripts found under **/etc/init.d**

For those who don't want to tamper with too much typing and manual configuration there is an [all distribution system V compatible ncurses interface text itnerface sysv-rc-conf](#) which could make your life easier on configuring services on non-systemd (old) Linux-es.

To install on Debian distros:

```
debian:~# apt-get install sysv-rc-conf
```

```
debian:~# sysv-rc-conf
```

SysV Runlevel Config -: stop service =/+: start service h: help q: quit								
service	1	2	3	4	5	0	6	S
acpid	[]	[X]	[X]	[X]	[X]	[]	[]	[]
alsa-utils	[]	[]	[]	[]	[]	[]	[]	[X]
anacron	[]	[X]	[X]	[X]	[X]	[]	[]	[]
apache-ht\$	[]	[]	[]	[]	[]	[]	[]	[]
apache2	[]	[]	[X]	[X]	[X]	[]	[]	[]
atd	[]	[X]	[X]	[X]	[X]	[]	[]	[]
aumix	[]	[X]	[X]	[X]	[X]	[]	[]	[]
avahi-daes\$	[]	[X]	[X]	[X]	[X]	[]	[]	[]
binfmt-su\$	[]	[X]	[X]	[X]	[X]	[]	[]	[]
bluetooth	[]	[X]	[X]	[X]	[X]	[]	[]	[]
bootlogs	[X]	[X]	[X]	[X]	[X]	[]	[]	[]
cgmanager	[]	[X]	[X]	[X]	[]	[]	[]	[]
cgproxy	[]	[X]	[X]	[X]	[X]	[]	[]	[]
clamav-da\$	[]	[]	[]	[]	[]	[]	[]	[]
clamav-fr\$	[]	[]	[X]	[X]	[X]	[]	[]	[]
cpufrequt\$	[]	[X]	[X]	[X]	[X]	[]	[]	[]
cron	[]	[X]	[X]	[X]	[X]	[]	[]	[]
cups	[]	[X]	[X]	[X]	[X]	[]	[]	[]
cups-brow\$	[]	[X]	[X]	[X]	[X]	[]	[]	[]
dbus	[]	[X]	[X]	[X]	[X]	[]	[]	[]
dnsmasq	[]	[X]	[X]	[X]	[]	[]	[]	[]
exim4	[]	[X]	[X]	[X]	[X]	[]	[]	[]

Use the arrow keys or mouse to move around. ^n: next pg ^p: prev pg
space: toggle service on / off

2. SystemD basic use Start / stop check service and a little bit of information for the novice

As most Linux kernel based distributions except some like **Slackware** and few others see [the full list of Linux distributions without systemd](#) (and aha yes slackw. users loves rc.local so much - we all do :) migrated and are nowadays using actively SystemD, to start / stop analyze running system running services / processes

systemctl - Control the systemd system and service manager

To check whether a service is enabled

systemctl is-active application.service

To check whether a unit is in a failed state

systemctl is-failed application.service

To get a status of running application via systemctl messaging

systemctl status sshd

```
? ssh.service - OpenBSD Secure Shell server Loaded: loaded (/lib/systemd/system/ssh.service;  
enabled; vendor preset: enabled) Active: active (running) since Sat 2019-07-06 20:01:02 EEST;  
2h 3min ago Main PID: 1335 (sshd) Tasks: 1 (limit: 4915) CGroup: /system.slice/ssh.service  
??1335 /usr/sbin/sshd -D ??? 06 20:01:00 noah systemd[1]: Starting OpenBSD Secure Shell  
server... ??? 06 20:01:02 noah sshd[1335]: Server listening on 0.0.0.0 port 22. ??? 06 20:01:02  
noah sshd[1335]: Server listening on :: port 22. ??? 06 20:01:02 noah systemd[1]: Started  
OpenBSD Secure Shell server.
```

To enable / disable application with systemctl **systemctl enable application.service**

systemctl disable application.service

To stop / start given application **systemctl stop sshd**

systemctl stop tor

To reload running application

systemctl reload sshd

Some applications does not have the right functionality in systemd script to reload configuration without fully restarting the app if this is the case use **systemctl reload-or-restart application.service**

systemctl list-unit-files

Then to view the content of a single service unit file:

```
:~# systemctl cat apache2.service
# /lib/systemd/system/apache2.service
[Unit]
Description=The Apache HTTP Server
After=network.target remote-fs.target nss-lookup.target

[Service]
Type=forking
Environment=APACHE_STARTED_BY_SYSTEMD=true
ExecStart=/usr/sbin/apachectl start
ExecStop=/usr/sbin/apachectl stop
ExecReload=/usr/sbin/apachectl graceful
PrivateTmp=true
Restart=on-abort

[Install]
WantedBy=multi-user.target
```

```
#!/bin/bash
# Start the ABRT daemon
#
# chkconfig: 35 82 16
# description: Saves segfault data, kernel oopses, fatal exceptions
# processname: abrt
# pidfile: /var/run/abrt.pid
### BEGIN INIT INFO
# Provides: abrt
# Required-Start: $syslog $local_fs messagebus
# Required-Stop: $syslog $local_fs
# Default-Stop: 0 1 2 6
# Default-Start: 3 5
# Short-Description: Saves segfault data, kernel oopses, fatal exceptions
# Description: Saves segfault data, kernel oopses, fatal exceptions
### END INIT INFO

# Source function library.
. /etc/rc.d/init.d/functions
ABRT_BIN="/usr/sbin/abrt"
LOCK="/var/lock/subsys/abrt"
RETVAL=0
```

```
[Unit]
Description=Saves segfault data, kernel oopses, fatal exceptions
After=syslog.target

[Service]
ExecStart=/usr/sbin/abrt
Type=forking
PIDFile=/var/run/abrt.pid

[Install]
WantedBy=multi-user.target
```

systemd's advancement over normal SystemV services it is able to track and show dependencies of a single run service for proper operation on other services

```
~# systemctl list-dependencies sshd.service
```

```
? ??system.slice
? ??sysinit.target
? ??dev-hugepages.mount
? ??dev-mqueue.mount
? ??keyboard-setup.service
? ??kmod-static-nodes.service
? ??proc-sys-fs-binfmt_misc.automount
? ??sys-fs-fuse-connections.mount
```



```
? ??sys-kernel-config.mount
? ??sys-kernel-debug.mount
? ??systemd-ask-password-console.path
? ??systemd-binfmt.service
....
...
.
```

You can also **mask** / **unmask** service e.g. make it temporary unavailable via systemd with

```
sudo systemctl mask nginx.service
```

it will then appear as masked if you do list-unit-files

If you want to change something on a systemd unit file this is done with

```
systemctl edit --full nginx.service
```

In case if some modificatgion was done to systemd service files e.g. lets say to /etc/systemd/system/apache2.service or even you've made a Linux system Upgrade recently that added extra systemd service config files it will be necessary to reload all files present in /etc/systemd/system/* with:

```
systemctl daemon-reload
```

Systemd has a target states which are pretty similar to the runlevel concept (e.g. runlevel 5 means

graphical etc.), for example to check the default target for a system:

One very helpful feature is to restart **systemd** but it seems this is not well documented as of now and though this might work after some system package upgrade roll-outs it is always better to reboot the system, but you can give it a try if restart can't be done due to application criticality.

To restart systemd and its spawned subprocesses do:

systemctl daemon-reexec

```
root@noah:/home/hipo# systemctl get-default  
graphical.target
```

to check all targets possible targets

```
root@noah:/home/hipo# systemctl list-unit-files --type=target  
UNIT FILE          STATE  
basic.target       static  
bluetooth.target   static  
busnames.target    static  
cryptsetup-pre.target static  
cryptsetup.target   static  
ctrl-alt-del.target disabled  
default.target     static  
emergency.target    static  
exit.target         disabled  
final.target        static  
getty.target        static  
graphical.target    static  
...
```

you can put the system in Single user mode if you like without running the good old well known command:

/sbin/init 1

command with

systemctl rescue

You can even shutdown / poweroff / reboot system via systemctl (though I never did that and I don't recommend) :)

To do so use:

systemctl halt
systemctl poweroff
systemctl reboot

For the lazy ones that don't want to type all the time like crazy to configure and manage simple systemctl set services take a look at [chkservice - an ncurses text based menu systemctl management interface](#)

As chkservice is relatively new it is still not present in stable Stretch Debian repositories but it is in **current testing Debian unstable Buster / Sid - Testing / Unstable distribution** and has installable package for **Ubuntu / Arch Linux and Fedora**

```
[x] > accounts-daemon.service      Accounts Service
[ ] > acpid.service               ACPI event daemon
[s] = alsa-restore.service        Save/Restore Sound Card State
[s] = alsa-state.service          Manage Sound Card State (restore and store)
-m- alsa-utils.service            /lib/systemd/system/alsa-utils.service
[x] = anacron-resume.service       /lib/systemd/system/anacron-resume.service
[x] = anacron.service             Run anacron jobs
[x] > apache-htcacheclean.service  Disk Cache Cleaning Daemon for Apache HTTP Server
[ ] = apache-htcacheclean@.service /lib/systemd/system/apache-htcacheclean@.service
[x] = apache2.service             The Apache HTTP Server
[ ] = apache2@.service            /lib/systemd/system/apache2@.service
-m- apparmor.service              apparmor.service
[s] = apt-daily.service            Daily apt activities
-m- auditd.service                auditd.service
[x] = autovt@.service             /lib/systemd/system/autovt@.service
[x] > avahi-daemon.service         Avahi mDNS/DNS-SD Stack
[x] = binfmt-support.service       Enable support for additional executable binary for
[x] > bluetooth.service           Bluetooth service
-m- bootlogd.service              /lib/systemd/system/bootlogd.service
-m- bootlogs.service              /lib/systemd/system/bootlogs.service
-m- bootmisc.service              /lib/systemd/system/bootmisc.service
[x] = brltty.service              Braille Device Support
[x] = casper.service              Shuts down the "live" preinstalled system cleanly
[s] = certbot.service             Certbot
[x] > cgmanager.service            Cgroup management daemon
[x] = cgproxy.service             Cgroup management proxy
-m- checkfs.service               /lib/systemd/system/checkfs.service
-m- checkroot-bootclean.service   /lib/systemd/system/checkroot-bootclean.service

20/527                                     ? - help
```

Picture Source Tecmint.com

```
[x]                                chkconfig 0.10
[ ]                                em/dbus-
[x]                                us-org.f
[x]                                em/dbus-
[s]                                us-org.f
[s]                                bus-org.
[ ]                                ll.servi
[ ]                                anager.s
[x] = Up/k - move cursor up. Down/j - move cursor down. S-up
[s] = PgUp/b - move page up. PgDown/f - move page down.
[s] =
-m- Action keys:
[x] = r - reload/update. q - exit. ice
[s] = Space - enable/disable. s - start/stop unit. line-upd
[s] > License: cleanup.
```

3. Analyzing and fix performance boot slowness issues due to a service taking long to boot

The first very useful thing is to know how long exactly all daemons / services got booted on your GNU / Linux OS.

```
linux-server:~# systemd-analyze
```

Startup finished in 4.135s (kernel) + 3min 47.863s (userspace) = 3min 51.998s

As you can see it reports both the kernel boot time and userspace (surrounding services that had to boot for the system to be considered fully booted).

Once you have the system properly booted you have a console or / ssh access

```
root@pcfreak:/home/hipo# systemd-analyze blame
```

2min 14.172s tor@default.service

1min 40.455s docker.service

1min 3.649s fail2ban.service

58.806s nmbd.service

53.992s rc-local.service

51.458s systemd-tmpfiles-setup.service

50.495s mariadb.service

46.348s snort.service

34.910s ModemManager.service

33.748s squid.service

32.226s ejabberd.service

28.207s certbot.service

28.104s networking.service

23.639s munin-node.service

20.917s smbd.service

20.261s tinyproxy.service

19.981s accounts-daemon.service

18.501s loadcpufreq.service

16.756s stunnel4.service

15.575s oidentd.service

15.376s dev-sda1.device

15.368s courier-authdaemon.service

15.301s sysstat.service

15.154s gpm.service

13.276s systemd-logind.service

13.251s rsyslog.service

13.240s lpd.service

13.237s pppd-dns.service

12.904s NetworkManager-wait-online.service

12.540s lm-sensors.service


12.525s watchdog.service
12.515s inetd.service

...

As you can see you get a list of services time took to boot in secs and you can further debug each of it to find out why it boots so slow (network / DNS / configuration issuse whatever).

On a servers it is useful to look up for some processes slowing it down like **gdm.service** etc.

Close up words rant on SystemD vs SysemV

	init	systemd
Статус на всички процеси	<code>service --status-all</code>	<code>systemctl service --state active</code>
Старт/Стоп на процес	<code>service httpd stop</code> <code>service httpd start</code>	<code>systemctl start httpd.service</code> <code>systemctl stop httpd.service</code>
Добавяне на процес при старт	<code>chkconfig httpd on</code> <code>chkconfig httpd off</code>	<code>systemctl enable httpd.service</code> <code>systemctl disable httpd.service</code>
Статус на процес	<code>service httpd status</code>	<code>systemctl status httpd.service</code>
Проверка дали е добавен процес за пускане при старт	<code>checkconfig httpd --list</code>	<code>systemctl is-enabled httpd</code>
Смяна на runlevel	<code>init 3</code>	<code>systemctl isolate runlevel13.target</code>
Рестарт/Стоп на системата	<code>shutdown</code> <code>reboot</code> <code>halt</code>	<code>systemctl shutdown</code> <code>systemctl reboot</code> <code>systemctl halt</code>

A lot could be ranted on what is better systemd or systemV. I personally hated systemd since day since I saw it being introduced first in Fedora / CentOS linuxes and a bit later in my beloved desktop used Debian Linux.

I still remember the bugs and headaches with systemd's introduction as it is with all new the early adoption of technology makes a lot of pain in the ass.

Eventually systemd has become a standard and with my employment as a contractor through **Itelligence GmbH** for **SAP AG** I now am forced to work with systemd daily on SLES 12 based Linuces and I was forced to get used to it.

But still there is my personal preference to SystemV even though the critics of slow boot etc.but for managing a multitude of Linux preinstalled servers like Virtual Machines and trying to standardize a Data Center with Tens of Thousands of Linuxes running on different Hypervisors VMWare / OpenXen + physical hosts etc. systemd brings a bit of more standardization that makes it a winner.