Автоматизация

Systemd менежира сервизите в CentOS както и в много други дистрибуции.
Отговаря също за много други обекти като:
– устройства;
– системни таймери;
– targets -> това са еквивалентите на Runlevels*

**"Нивото на изпълнение е състояние на init и цялата система, която определя какви системни услуги работят. Нивата на изпълнение се идентифицират с числа. Някои с нива на изпълнение, за да определят кои подсистеми работят, напр. Дали X работи, дали мрежата работи и т.н."*

***Systemd objects** се наричат още **units** за всеки **unit** има **unitfile** за конфигурация.*

Това което трябва да знаем по тази тема са само:

Service units** и **Service unit files.

*Командата отговорна за тях е **systemctl**.*

*За да видим нашите service **unit files** напишете в конзолата:*

systemctl list-unit-files –at service

Изхода за на тази команда ще изглежда ето така:

UNIT FILE	STATE	VENDOR PRESET
accounts-daemon.service	enabled	enabled
acpid.service	disabled	enabled
alsa-restore.service	static	enabled
alsa-state.service	static	enabled
alsa-utils.service	masked	enabled
anacron.service	enabled	enabled
apparmor.service	enabled	enabled
apport-autoreport.service	static	enabled
apport-forward@.service	static	enabled
apport.service	generated	enabled
apt-daily-upgrade.service	static	enabled
apt-daily.service	static	enabled
autovt@.service	enabled	enabled
avahi-daemon.service	enabled	enabled
bluetooth.service	enabled	enabled
bolt.service	static	enabled
brltty-udev.service	static	enabled
brltty.service	disabled	enabled
clean-mount-point@.service	static	enabled
colord.service	static	enabled
configure-printer@.service	static	enabled
console-getty.service	disabled	disabled
console-setup.service	enabled	enabled
container-getty@.service	static	enabled
cron.service	enabled	enabled
cryptdisks-early.service	masked	enabled
cryptdisks.service	masked	enabled
cups-browsed.service	enabled	enabled
cups.service	enabled	enabled
dbus-fi.w1.wpa_supplicant1.service	enabled	enabled
dbus-org.bluez.service	enabled	enabled
dbus-org.freedesktop.Avahi.service	enabled	enabled
dbus-org.freedesktop.hostname1.service	static	enabled
dbus-org.freedesktop.locale1.service	static	enabled
dbus-org.freedesktop.login1.service	static	enabled
dbus-org.freedesktop.ModemManager1.service	enabled	enabled
dbus-org.freedesktop.nm-dispatcher.service	enabled	enabled
dbus-org.freedesktop.resolve1.service	enabled	enabled
dbus-org.freedesktop.thermald.service	enabled	enabled
dbus-org.freedesktop.timedate1.service	static	enabled
dbus-org.freedesktop.timesync1.service	enabled	enabled
dbus.service	static	enabled
debug-shell.service	disabled	disabled
display-manager.service	static	enabled
dmesg.service	enabled	enabled
e2scrub@.service	static	enabled
e2scrub_all.service	static	enabled
e2scrub_fail@.service	static	enabled
e2scrub_reap.service	enabled	enabled
emergency.service	static	enabled
fprintd.service	static	enabled

Обикновено **systemctl list-unit-files -t service** ще ви покаже всички сервизи които са конфигурирани да се включват сами.

Добавяйки **-a** всъщност сортира всичките --> включени и изключени.
systemctl list-unit-files имайте в предвид, че тази команда само покзва статуса на сервизите.

Enabled	Сервиза ще се включи автоматично по време на старт
Disabled	Сервиза няма да стартира автоматично
Static	Сервиза не е включен

Някой сервизи не са имани в предвид да стартират автоматично.

sudo systemctl list-units -at service

Първата колона е със името на сервиза, втората колона показава дали юнит файла е зареден, третата колона показва о
сервиза,

четвъртата колона показва по детайлна информация за статута на сервиза - дали работи или не.

Можем да ги сортираме както си искаме например:
sudo systemctl list-units -t service --state running

Можем да cat-нем сервизите за да видим конфигурациите им:

systemctl cat rsyslog.service

```
nick@sportsmodule-xmachine:~$ systemctl cat rsyslog.service
# /lib/systemd/system/rsyslog.service
[Unit]
Description=System Logging Service
Requires=syslog.socket
Documentation=man:rsyslogd(8)
Documentation=https://www.rsyslog.com/doc/

[Service]
Type=notify
ExecStart=/usr/sbin/rsyslogd -n -iNONE
StandardOutput=null
Restart=on-failure

# Increase the default a bit in order to allow many simultaneous
# files to be monitored, we might need a lot of fds.
LimitNOFILE=16384

[Install]
WantedBy=multi-user.target
Alias=syslog.service
```

Това, което виждаме тук, е файлът на сервизната единица,
който определя зависимостите на услугата,
каква команда се изпълнява и
какво трябва да направи услугата, ако не успее.

Можем да взем статуса на единична услуга:
systemctl status rsyslog

```
nick@sportsmodule-xmachine:~$ systemctl status rsyslog
● rsyslog.service - System Logging Service
     Loaded: loaded (/lib/systemd/system/rsyslog.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2020-11-09 13:35:58 EET; 4h 30min ago
TriggeredBy: ● syslog.socket
       Docs: man:rsyslogd(8)
             https://www.rsyslog.com/doc/
   Main PID: 608 (rsyslogd)
      Tasks: 4 (limit: 16582)
     Memory: 4.4M
     CGroup: /system.slice/rsyslog.service
             └─608 /usr/sbin/rsyslogd -n -iNONE

ное 09 13:35:58 sportsmodule-xmachine systemd[1]: Starting System Logging Service...
ное 09 13:35:58 sportsmodule-xmachine rsyslogd[608]: imuxsock: Acquired UNIX socket '/run/systemd/journal/syslog' (fd 3) from systemd.  [v8.2001.0]
ное 09 13:35:58 sportsmodule-xmachine rsyslogd[608]: rsyslogd's groupid changed to 110
ное 09 13:35:58 sportsmodule-xmachine rsyslogd[608]: rsyslogd's userid changed to 104
ное 09 13:35:58 sportsmodule-xmachine rsyslogd[608]: [origin software="rsyslogd" swVersion="8.2001.0" x-pid="608" x-info="https://www.rsyslog.com"] start
ное 09 13:35:58 sportsmodule-xmachine systemd[1]: Started System Logging Service.
```

Системните сервизи са процеси стартирани от операционната система и са менежирани с командата systemctl и с нея
systemctl list-unit-files –at service

```
nick@sportsmodule-xmachine:~$ sudo systemctl list-unit-files -at service
UNIT FILE STATE VENDOR PRESET

0 unit files listed.
```

Нека спрем atd **cron.service** ще ви трябват високи привилегии.
sudo systemctl stop cron

Нека проверим статута му:
sudo systemctl status cron

```
nick@sportsmodule-xmachine:~$ sudo systemctl status cron
● cron.service - Regular background program processing daemon
     Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabled)
     Active: inactive (dead) since Mon 2020-11-09 18:09:48 EET; 7s ago
       Docs: man:cron(8)
    Process: 583 ExecStart=/usr/sbin/cron -f $EXTRA_OPTS (code=killed, signal=TERM)
   Main PID: 583 (code=killed, signal=TERM)

ное 09 16:30:01 sportsmodule-xmachine CRON[39934]: (root) CMD ([ -x /etc/init.d/anacron ] && if [ ! -d /run/systemd/system ]; then /usr/sbin/invoke-rc.d anacron start >/dev/null; fi)
ное 09 16:30:01 sportsmodule-xmachine CRON[39933]: pam_unix(cron:session): session closed for user root
ное 09 17:17:01 sportsmodule-xmachine CRON[41636]: pam_unix(cron:session): session opened for user root by (uid=0)
ное 09 17:17:01 sportsmodule-xmachine CRON[41637]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
ное 09 17:17:01 sportsmodule-xmachine CRON[41636]: pam_unix(cron:session): session closed for user root
ное 09 17:30:01 sportsmodule-xmachine CRON[41850]: pam_unix(cron:session): session opened for user root by (uid=0)
ное 09 17:30:01 sportsmodule-xmachine CRON[41850]: pam_unix(cron:session): session closed for user root
ное 09 18:09:48 sportsmodule-xmachine systemd[1]: Stopping Regular background program processing daemon...
ное 09 18:09:48 sportsmodule-xmachine systemd[1]: cron.service: Succeeded.
ное 09 18:09:48 sportsmodule-xmachine systemd[1]: Stopped Regular background program processing daemon.
```

Нека го стартираме:
systemctl start cron

Верифицирайте, че пак работи с:
sudo systemctl status cron

```
nick@sportsmodule-xmachine:~$ sudo systemctl status cron
● cron.service - Regular background program processing daemon
     Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2020-11-09 18:10:31 EET; 5s ago
       Docs: man:cron(8)
   Main PID: 45910 (cron)
      Tasks: 1 (limit: 16582)
     Memory: 328.0K
     CGroup: /system.slice/cron.service
             └─45910 /usr/sbin/cron -f

ное 09 18:10:31 sportsmodule-xmachine systemd[1]: Started Regular background program processing daemon.
ное 09 18:10:31 sportsmodule-xmachine cron[45910]: (CRON) INFO (pidfile fd = 3)
ное 09 18:10:31 sportsmodule-xmachine cron[45910]: (CRON) INFO (Skipping @reboot jobs -- not system startup)
```

За да рестартираме ползваме **systemctl restart <service name>.**

Ако искаме просто да разберем дали някой сервиз просто работи може да ползваме:

systemctl is-active <service name>

systemctl is-active cron

```
nick@sportsmodule-xmachine:~$ systemctl is-active cron
active
```

Ако искаме да ограничиме сервиз от това да бъде активен или сам да се включва можем да го маскираме.

systemctl mask cron

```
nick@sportsmodule-xmachine:~$ systemctl mask cron
Created symlink /etc/systemd/system/cron.service → /dev/null.
```

```
nick@sportsmodule-xmachine:~$ sudo systemctl status cron
● cron.service
     Loaded: masked (Reason: Unit cron.service is masked.)
     Active: active (running) since Mon 2020-11-09 18:10:31 EET; 1min 34s ago
   Main PID: 45910 (cron)
      CGroup: /system.slice/cron.service
              └─45910 /usr/sbin/cron -f

Ное 09 18:10:31 sportsmodule-xmachine systemd[1]: Started Regular background program processing daemon.
Ное 09 18:10:31 sportsmodule-xmachine cron[45910]: (CRON) INFO (pidfile fd = 3)
Ное 09 18:10:31 sportsmodule-xmachine cron[45910]: (CRON) INFO (Skipping @reboot jobs -- not system startup)
Ное 09 18:11:37 sportsmodule-xmachine systemd[1]: cron.service: Current command vanished from the unit file, execution of the command list won't be resumed.
```

systemctl unmask cron

```
nick@sportsmodule-xmachine:~$ systemctl unmask cron
Removed /etc/systemd/system/cron.service.
```

Можем да ги **enable** и **disable**

sudo systemctl disable cron

```
nick@sportsmodule-xmachine:~$ sudo systemctl disable cron
Synchronizing state of cron.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable cron
Removed /etc/systemd/system/multi-user.target.wants/cron.service.
nick@sportsmodule-xmachine:~$
```

```
nick@sportsmodule-xmachine:~$ systemctl is-enabled cron.service
disabled
```

sudo systemctl enable cron

```
nick@sportsmodule-xmachine:~$ sudo systemctl enable cron
Synchronizing state of cron.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable cron
Created symlink /etc/systemd/system/multi-user.target.wants/cron.service → /lib/systemd/system/cron.service.
nick@sportsmodule-xmachine:~$ sudo systemctl is-enabled cron
enabled
nick@sportsmodule-xmachine:~$
```

Проверката е:

systemctl is-enabled cron

systemd ни позволява да дефинираме таймерни единици:

systemd timers те са много по силна и алтернативна версия на **cron jobs**

Имаме два тима **timer-units**:

Real-time timers;
• Активират се на календарни събития.
• Подобни на cron jobs.
• Стартират базирано на датата и часа.
Можем да сложим таймер на 31 декември с цел да ни каже Happy New Year или може да сложим таймер с цел да направи цял backup на системата.

Monotonic timers;
• Активират се на времеви интервал спрямо начална точка .
• Например ако таймер трябва да се активира 5 минути след буут процеса.
• Или 30 секунди след като се логнем.

Ще използваме монотен таймер.

Защо е по добре да ползваме таймери вместо cron джобове е, защото всеки джоб ще си има свой сървис файл и свой таймер, средата с която работи е доста по разбираема от **conjob**.

• Могат да работят в тяхната си среда.
• Могат да се добавят в **cgroups**. Могат да имат зависимости спрямо други **systemd** юнити.
• Например мрежи, примерно може да имаме джоб който ще тръгва в момента в който имаме връзка с интернет.
• Също джобовете ще бъдат логнати в **systemd journal**.

Cron jobs

https://crontab.guru/

vim /etc/crontab

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

• Лесни за създаването и почти винаги са еднолиниини.
• Имат съпорт за имейли базирано на резултата от джоба.

Тази функционалност може да бъде заменена с таймери от systemd.

Файловете crontab са мястото, където се съхраняват списъците със задачи и други инструкции към демона cron.

crontab -e

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
```

Един елементарен cron е:

@daily root apt update -y && apt upgrade –y

Променяте файла crontab и може да впишете вашите автоматични дейности.

За да проверите дали cron job работи отворете:
cat /var/log/syslog | grep cron

Ако не работи отворете и махнете # пред кронджоба ви.
/etc/rsyslog.d/50-default.conf

Systemd timers имат **.timer** extension и сервиз със **.service** екстеншън.

Можем да имаме **backup.timer** и **backup.service** файл където таймера ще активира сървиса.

След това сервиза може да пусне скрипт със всякакво име като например **backup.sh.**
systemctl list-timers

Изхода показва кога за последно раймера е ръннал за последно и кога пак.

Казва ни името на таймера и сервиза.

Примерен **сървис** файл:

```
[Unit]
Description=System Update
[Service]
Type=simple
ExecStart=/root/bin/script.sh
[Install]
WantedBy=muti-user.target
```

Примерен **timer** файл:

```
[Unit]
Description=System updates automatically after it is booted
[Timer]
OnStartupSec=1 min
Unit=update_job.service
[Install]
WantedBy=multi-user.target
```

Monotonic timers:
Те са спецефични до стартираща точка.

Monotonic timers:
Те са спецефични до стартираща точка.

OnActiveSec	Това определя таймер спрямо момента, в който таймерът е активиран.
OnBootSec	Това дефинира таймер по отношение на момента на зареждане на маши ната.
OnStartupSec	Това дефинира таймер спрямо времето, когато мениджърът на услуги за първи път стартира. За системните таймерни единици това е много подо бно на OnBootSec =, тъй като системният мениджър на услуги обикнове но започва много рано при стартиране. Това е предимно полезно, когато е конфигурирано в единици, работещи в мениджъра на услуги за всеки потребител, тъй като мениджърът на потребителски услуги обикновено започва само при първо влизане, а не по време на зареждане.
OnUnitActiveSec	Това определя таймер по отношение на времето, когато таймерът, който трябва да се активира, е бил активиран за последно.
OnUnitInactiveSec	Това определя таймер по отношение на времето, когато таймерът, който трябва да се активира, е бил деактивиран за последно.