< Linux

Systemd

- **Szerző:** Sallai András

- Web: http://szit.hu

A systemd-ről

A systemd egy **rendszer-előkészítő** és **rendszermenedzselő** szoftver, amely egy új (2018) szabvány a Linuxos rendszerek számára. Bár jelentős vita van arról, hogy jó vagy nem jó, a a legtöbb Linux terjesztés készítő elkezdte lecserélni a SysV rendszerelőkészítőt systemd-re.

A Debian GNU/Linux 8 verzióval vezzette be ez az új démonkezelési rendszert.

A systemd gyorsabb rendszerindítást tesz lehetővé. Minden démon egy egyszerű konfigurációs scripttel indul. Ha leáll egy démon, automatikusan újraindításra kerül. Minden processz saját cgroupban fut, alapértelmezetten, így azok egymástól jól elszeparálva futnak. A systemdnek saját naplózó rendszere van. Lehetővé teszi konkténerek kezelését.

Áttekintjük, a systemctl parancs használatát, amellyel kezelhetjük a szolgáltatásokat, megtekinthetjük vagy megváltoztathatjuk állapotukat, dolgozhatunk a konfigurációs fájlokkal.

Ne felejtsük el, hogy a systemd-t sok Linux terjesztés beépíti saját rendszerébe, de nem mind.

Szolgáltatás menedzsment

Az rendszerelőkészítő feladata, hogy a Linux kernel indulása után előkészítse az induló szolgáltatásokat. A rendszerelőkészítővel kezeljük is szolgáltatásokat, démonokat a rendszer futása közben.

A systemd rendszerben a legtöbb művelet az egységekhez (Unit) kapcsolódik. Az egységek kategorizálva vannak az általuk képviselt erőforrás típusa szerint. Az egység típusát a fájl kiterjesztése mutatja számunkra.

A szolgáltatások számára egy .service kiterjesztésű fájlt használunk. A szolgáltatások kezelése során, a .service kiterjesztés megadása általában elhagyható.

Szolgáltatások indítása, leállítása

Indítás:

```
systemctl start alkalmazas.service
```

Indítás a .service kiterjesztés elhagyásával:

```
systemctl start alkalmazas
```

Szolgáltatás leállítása:

```
systemctl stop alkalmazás.
```

Újraindítás, újratöltés

```
systemctl restart alkalmazas.service
```

Konfiguráció újratöltése:

```
systemctl reload alkalmazas.service
```

Ha nem vagyunk benne biztosak, hogy a konfiguráció újratölthető-e az adott alkalmazás esetén, akkor használjuk a reload-or-restart parancsot:

```
systemctl reload-or-restart alkalmazas.service
```

Szolgáltatás engedélyezése, tiltása

A start és stop parancsok a rendszer újraindítása után érvénytelenek, csak az adott munkamenetre érvényesek. Ha a rendszer indulásával együtt szeretnénk egy szolgáltatást elindítani, akkor azt a következő paranccsal tehetjük meg:

```
systemctl enable alkalmazas.service
```

Ez a parancs egy szimbolikus linket hoz létre, rendszerint a következő könyvtárból

```
/lib/systemd/system/
```

A szimbolikus link a következő helyen jön létre:

```
/etc/systemd/system
```

A szolgáltatás tiltása, a rendszer indulásával együtt:

```
systemctl disable alkalmazas.service
```

Ez a parancs törli az „enable” paranccsal létrehozott szimbolikus linket.

Ezek a parancsok az aktuális munkamenetben nem indítják el, vagy nem állítják le az adott szolgáltatást.

A szolgáltatás ellenőrzése

A szolgáltatás állapotának ellenőrzéséhez használjuk a következő parancsot:

```
systemctl status alkalmazas.service
```

Legyen például egy apache2 webhely, amelynek az állapota ehhez hasonló lehet:

```
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor
preset:
   Active: active (running) since Wed 2018-07-18 08:04:12 CEST; 6h ago
  Process: 4875 ExecReload=/usr/sbin/apachectl graceful (code=exited,
status=0/
  Process: 931 ExecStart=/usr/sbin/apachectl start (code=exited,
status=0/SUCCE
 Main PID: 1171 (apache2)
    Tasks: 6 (limit: 4915)
   Memory: 37.9M
      CPU: 1.060s
   CGroup: /system.slice/apache2.service
           ├─1171 /usr/sbin/apache2 -k start
           ├─4881 /usr/sbin/apache2 -k start
           ├─4882 /usr/sbin/apache2 -k start
           ├─4883 /usr/sbin/apache2 -k start
           ├─4884 /usr/sbin/apache2 -k start
           └─4885 /usr/sbin/apache2 -k start

júl 18 08:04:05 tatami systemd[1]: Starting The Apache HTTP Server...
júl 18 08:04:12 tatami systemd[1]: Started The Apache HTTP Server.
júl 18 10:13:18 tatami systemd[1]: Reloading The Apache HTTP Server.
```

Ha csak szeretnénk ellenőrizni, hogy a szolgáltatás aktív-e, futtassuk a következő parancsot:

```
systemctl is-active alkalmazas.service
```

A parancs a képernyőre írja az active vagy a inactive szót. A parancs visszatérési értéke 0, ha a szolgáltatás aktív. Ha nem aktív nullától eltérő szám.

Azt is lekérdezhetjük, hogy rendszerindításkor engedélyezve, vagy tiltva van:

```
systemctl is-enabled alkalmazas.service
```

A válasz enabled vagy disabled. A parancs visszatérési értéke 0 vagy 1.

Ellenőrizhetjük, hogy nem hibás-e a szolgáltatásunk:

```
systemctl is-failed alkalmazas.service
```

Ez a parancs active vagy failed szavakkal tér vissza, attól függően, hogy fut-e a szolgáltatás. A

szolgáltatás le van állítva, akkor unknown vagy inactive szavakkal is visszatérhet. Ha hiba történt, akkor a parans 0-val tér vissza, minden más esetben 1-el.

A rendszer állapotának áttekintése

Egységek (Unit) áttekintése

```
systemctl list-units
```

A kimenetben a következő oszlopokat látjuk:

- UNIT – az egység (Unit) neve
- LOAD – az egyég a memóriában van-e
- ACTIVE – az egység aktív-e
- SUB – alacsonyabb szintű állapot
- DESCRIPTION – rövid leírás

Ugyanezt az eredményt kapjuk, ha csak simán kiadjuk a systemctl parancsot:

```
systemctl
```

Plusz kapcsolóval még több információ nyerhető:

```
systemctl list-units --all
```

A --all kapcsoló azokat az egységeket is megmutatja, amelyek nem aktívak.

Esetleg konkrétan megadhatunk egy állapotot:

```
systemctl list-units --all --state=inactive
```

Szűrhetünk típusra:

```
systemctl list-units --all --type=service
```

Az összes egységfájl listázása

A list-units parancs csak azokat az egységeket mutatja meg számunkra, amelyeket a systemd megpróbál értelmezni és memóriába tölteni. A systemd csak a szükséges egységeket tölti be, de nem az összest. Az összes egység listázása:

```
systemctl list-unit-files
```

Csak azok az erőforrások jelennek meg, amelyeket a systemd egységfájlokból ismer. A kimenetben két oszlop jelenik meg:

- UNIT FILE

- STATE

```
UNIT FILE                                    STATE
proc-sys-fs-binfmt_misc.automount            static
-,mount                                      generated
```

Az állapotok rendszerint „enabled”, „disabled”, „static”, „masked”. Ebben a környezetben a static állapot azt jelenti az egységfájl nem tartalmaz „install” szekciót, amelyet az engedélyezéshez használunk. Ezek szerint ez az egység nem engedélyezhető. Ez lehet azért mert egy egyszeri művelet végrehajtásáról van szó, vagy azért mert egy másik egység függőségeként használjuk, vagyis nem fut önmagától.

Szolgáltatások

Milyen ismert szolgáltatások vannak?:

```
systemctl
```

Milyen szolgáltatások futnak?

```
systemctl -t service
```

A lehetséges kimenetet itt látjuk:

```
  UNIT                      LOAD   ACTIVE SUB     DESCRIPTION
  acpid.service             loaded active running ACPI event daemon
  apache2.service           loaded active running LSB: Apache2 web server
  atd.service               loaded active running Deferred execution
scheduler
  console-setup.service     loaded active exited  LSB: Set console font
and ke
  cron.service              loaded active running Regular background
program p
  dbus.service              loaded active running D-Bus System Message Bus
  exim4.service             loaded active exited  LSB: exim Mail Transport
Age
  getty@tty1.service        loaded active running Getty on tty1
  ifup@eth0.service         loaded active exited  ifup for eth0
● isc-dhcp-server.service   loaded failed failed  LSB: DHCP server
  kbd.service               loaded active exited  LSB: Prepare console
  keyboard-setup.service    loaded active exited  LSB: Set preliminary
keymap
  kmod-static-nodes.service loaded active exited  Create list of required
stat
  mysql.service             loaded active running LSB: Start and stop the
mysq
  networking.service        loaded active exited  LSB: Raise network
interface
  nfs-common.service        loaded active running LSB: NFS support files
```

```
commo
  nmbd.service                loaded active running LSB: start Samba NetBIOS
nam
  ntp.service                 loaded active running LSB: Start NTP daemon
  postfix.service             loaded active running LSB: Postfix Mail
Transport
  rc-local.service            loaded active exited  /etc/rc.local
Compatibility
  rpcbind.service             loaded active running LSB: RPC portmapper
replacem
  rsyslog.service             loaded active running System Logging Service
  samba-ad-dc.service         loaded active exited  LSB: start Samba daemons
for
  smbd.service                loaded active running LSB: start Samba
SMB/CIFS da
  ssh.service                 loaded active running OpenBSD Secure Shell
server
  systemd-journald.service    loaded active running Journal Service
  systemd-logind.service      loaded active running Login Service
  systemd-modules-load.service loaded active exited  Load Kernel Modules
  systemd-random-seed.service loaded active exited  Load/Save Random Seed
  systemd-remount-fs.service  loaded active exited  Remount Root and Kernel
File
  systemd-setup-dgram-qlen.service loaded active exited  Increase datagram
queue
  systemd-sysctl.service      loaded active exited  Apply Kernel Variables
  systemd-tmpfiles-setup-dev.service loaded active exited  Create Static
Device
  systemd-tmpfiles-setup.service loaded active exited  Create Volatile Files
and
  systemd-udev-trigger.service loaded active exited  udev Coldplug all
Devices
  systemd-udevd.service       loaded active running udev Kernel Device
Manager
  systemd-update-utmp.service loaded active exited  Update UTMP about System
Boo
  systemd-user-sessions.service loaded active exited  Permit User Sessions
  udev-finish.service         loaded active exited  Copy rules generated
while t

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.

39 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

Egy szolgáltatás állapota

```
$ systemctl status apache2
```

A lehetséges kimenet:

```
 systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor
preset:
   Active: active (running) since Sun 2017-09-17 07:33:37 CEST; 2h 56min ago
  Process: 3054 ExecReload=/usr/sbin/apachectl graceful (code=exited,
status=0/S
  Process: 867 ExecStart=/usr/sbin/apachectl start (code=exited,
status=0/SUCCES
 Main PID: 1076 (apache2)
    Tasks: 6 (limit: 4915)
   Memory: 35.6M
      CPU: 579ms
   CGroup: /system.slice/apache2.service
           ├─1076 /usr/sbin/apache2 -k start
           ├─3060 /usr/sbin/apache2 -k start
           ├─3061 /usr/sbin/apache2 -k start
           ├─3062 /usr/sbin/apache2 -k start
           ├─3063 /usr/sbin/apache2 -k start
           └─3064 /usr/sbin/apache2 -k start
```

Egységmenedzsment

Eddig dolgoztunk szolgáltatásokkal és azok megjelenítésével foglalkoztunk. Most néhány plusz lehetőséget nézünk meg.

Egységfájlok megjelenítése

A systemd által betöltött egységek megjeleníthetők a cat paranccsal.

```
systemctl cat ssh.service
```

```
# /lib/systemd/system/ssh.service
[Unit]
Description=OpenBSD Secure Shell server
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/sshd_not_to_be_run

[Service]
EnvironmentFile=-/etc/default/ssh
ExecStartPre=/usr/sbin/sshd -t
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/usr/sbin/sshd -t
```

```
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartPreventExitStatus=255
Type=notify

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

A függőségek megjelenítése

A függőségeket a list-dependecies paranccsal lehetséges.

```
systemctl list-dependecies sshd.service
```

```
sshd.service
● ├─system.slice
● └─sysinit.target
●   ├─dev-hugepages.mount
●   ├─dev-mqueue.mount
●   ├─keyboard-setup.service
●   ├─kmod-static-nodes.service
●   ├─lvm2-lvmetad.socket
...
```

A rekurzív függőségek csak a .target egységek számára jelennek meg. Ha az összes rekurzív függőséget szeretnénk megjeleníteni, akkor használjuk a --all kapcsolóval.

A fordított függőségek megjelenítéséhez használjuk a --reverse kapcsolót.

Használhatjuk még a --before és a --after kapcsolókat, amelyek a megmutatják mitől függ és mi függ az aktuális egységtől.

Egység tulajdonságok ellenőrzése

```
systemctl show sshd.service
```

Lehetséges kimenet:

```
Type=notify
Restart=on-failure
NotifyAccess=main
RestartUSec=100ms
TimeoutStartUSec=1min 30s
TimeoutStopUSec=1min 30s
RuntimeMaxUSec=infinity
...
```

Ha egyetlen tulajdonságot szeretnénk megjeleníteni a -p kapcsolóval tehetjük meg:

```
systemctl show sshd.service -p Restart
Restart=on-failure
```

Egységek maszkolása

Az egységeket láttuk, hogy indíthatók automatikusan vagy kézzel. Az indítás azonban teljesen letiltható maszkolással:

```
systemctl mask alkalmazas.service
```

Az eredmény a list-unit-files kimenetében látható:

```
systemctl list-unit-files
```

```
systemctl mask apache2
Created symlink /etc/systemd/system/apache2.service → /dev/null.
```

```
systemctl list-unit-files
...
apache2.service                                   masked
...
```

A maszkolás megszüntetése:

```
systemctl unmask apache2
```

Egységfájlok szerkesztése

Szerkesztés:

```
systemctl edit alkalmazas.service
```

Az alkalmazás számára ezzel egy kiegészítő egységfájl beállító állományt hozunk létre.

Például:

```
systemctl edit apache2.service
```

Egy üres állomány fog megnyílni. Ez lehetőséget ad plusz direktívákat adjunk az egységfájlhoz. Egy könyvtár fog létrejönni a /etc/systemd/system könyvtárban, amely tartalmazza az egység nevét és „.d” utótagot. Az apache esetén apache.service.d jön létre.

A fenti könyvtárban egy override.conf állomány jön létre. Az egység betöltésekor az eredeti beállítások is betöltődnek, majd az override.conf fájlban megadott részletek felülírják a megadott részeket. Az eredeti egységfájlt kiegészítettük.

Ha az egész egységfájlt felül szeretnénk írni, használjuk a --full kapcsolót.

```
systemctl edit --full apache2.service
```

Ez betölti az eredeti teljes egységfájlt. Ha elkészültünk a szerkesztéssel, a fájl a /etc/systemd/system könyvtárba íródik ki. Ez fájl elsőbbséget fog élvezni a rendszer egységállományával szemben, amely a /lib/systemd/system könyvtárban található.

A kiegészítés törlése a /etc/systemd/system könyvtárban létrejött könyvtár törlésével oldható meg:

```
rm -r /etc/systemd/system/apache2.service.d
```

A teljes változtatás is így törölhető:

```
rm /etc/systemd/system/apache2.service
```

A fájl vagy a könyvtár törlése után újra kell tölteni a systemd démont:

```
systemctl daemon-reload
```

Szabályozás célokkal

A futási szintek a systemd esetén nem léteznek, de ezeket helyettesíti a systemd esetén a cél (target). A célok speciális egységfájlok, amelyek a rendszer állapotát és a szinkronizációs pontokat írják le. A célfájlok .target kiterjesztést kapnak, hasonlóan a normál egységfájlokhoz. A célok valójában nem tesznek semmit, helyette más egységeket csoportosítanak.

Ez a rendszer arra használható, hogy bizonyos állapotokat vegyen fel egy rendszer, más előkészítő (init) rendszerek futási szintjeihez hasonlóan. Ezeket arra használjuk, hogy bizonyos funkciók elérhetők legyenek, ahelyett, hogy az egyes egységeket külön-külön állítanám be.

Például van egy network.target. Ha ez be van töltve, azt jelzi, hogy a hálózat készen áll a működésre. Ez a cél függőségként használható más célok beállításaiban. Megadható milyen viszonyban van az egyik cél a másikkal. Olyan beállításokra gondolunk mint:

- WantedBy=
- RequiredBy=

A fenti direktívák után több cél is megadható szóközzel tagolva.

Ha egy beállításban a WantedBy= beállítás után írjuk például a network.target célt, akkor a konfigurált cél a network.target céllal fog együtt indulni. Ha network.target cél még sem indul el, ez nem érinti az éppen konfigurált célt működését.

```
WantedBy=network.target
```

Ha a RequiredBy= után írjuk a network.target célt, a és a network nem indul el, akkor a konfigurált cél is kikapcsolásra kerül.

```
RequiredBy=network.target
```

Az alapértelmezett cél

A systemd rendelkezik egy alapértelmezett céllal, amelyet akkor használ, amikor a rendszer indul (boot).

Az alapértelmezett cél lekérdezése:

```
systemctl get-default
```

Lehetséges kimenet például:

```
graphical.target
```

vagy:

```
multi-user.target
```

Az új alapértelmezett cél beállítása set-default paranccsal lehetséges:

```
systemctl set-default multi-user.target
```

Az elérhető célok

```
systemctl list-unit-files --type=target
```

A futási szintekkel ellentétben több cél is lehet egyszerre aktív.

Az összes aktív cél megjelenítése:

```
systemctl list-units --type=target
```

Célok elkülönítése

Lehetőség van a egy egység (unit) minden függőségének elindítására, és minden más leállítására. Erre használható a isolate parancs. Ez hasonlít a futási szintváltáshoz.

Ha például a graphical.target aktív, de közben szeretnénk minden grafikus rendszert leállítani, a multi-user.target elkülöníthető. A graphical.target függ a multi-user.target-től, de fordítva nem igaz.

Az elkülönítés előtt nézzük meg, milyen függőségei vannak a multi-user.target-nek:

```
systemctl list-dependecies multi-user.target
```

Az elkülönítés:

```
systemctl isolate multi-user.target
```

Fontos események parancsai

A systemctl meghatároz néhány speciális parancsot.

Ha a rendszer rescue, azaz egyfelhasználós módba szertnénk léptetni, a következőt tesszük:

```
systemctl isolate rescue.target
```

Ez rövidíthető így:

```
systemctl rescue
```

Az eseményről történő figyelmeztetés is lehetővé válik:

```
systemctl halt
```

Teljes kikapcsolás:

```
systemctl poweroff
```

Újraindítás:

```
systemctl reboot
```

A legtöbb gépen azért használható a rövid változat:

```
poweroff
reboot
```

Ha megnézzük ezeket a parancsokat az „ls -l” paranccsal, láthatjuk, hogy azok a /bin/systemctl parancsra mutatnak.

Naplózás

Ha a systemctl paranccsal újraindítunk egy szolgáltatást, az alapértelmezett kimenetre nem ír semmit. A kimenetet a naplózásra kerül (journal), de nem a syslogba.

Az állapot lekérdezésével a naplóeredményeket lekérdezhetjük, a következő módon:

```
systemctl status apache2
```

Ebben a formában olyan szolgáltatások státuszát is lekérdezhetjük, amelyet a systemd-vel nem is kezelünk. Ilyen a cron:

```
systemctl status cron
```

Ezek az adatok azonban nem tartósak. A rendszer újraindítása után elvesznek. Ennek következménye is, hogy csak rendszergazdaként olvashatjuk.

A systemd, tulajdonképpen fel van készítve, hogy könyvtárba is maradandóan naplózzon. Csak létre kell hozni a könyvtárat számára.

Könyvtár létrehozása:

```
mkdir /var/log/journal
```

A könyvtár csoportja legyen a systemd-journal:

```
chgrp systemd-journal /var/log/journal
```

Jogosultságok beállítása:

```
chmod g+rwx /var/log/journal
```

Ha azt szeretnénk, hogy a „janos” nevű felhasználó képes legyen megnézni a naplót:

```
usermod -a -G systemd-journal janos
```

Indítsuk újra a rendszert:

```
systemctl reboot
```

Ezek után az apache2 eseményeit így nézhetjük meg:

```
journalctl -u apache2
```

A Ctrl + Alt + Del

A Ctrl + Alt + Del billentyűkombináció alapértelmezetten újraindítja a rendszert. Ezt megváltoztathatjuk a következő módon:

Létre kell hozni egy ctrl-alt-del.target-et. Előbb nézzük meg a link jelenleg célját:

```
# ls -ls /lib/systemd/system/ctrl-alt-del.target
```

A cél jelenleg:

- reboot.target

Változtassuk meg a célt:

```
# ln -s /lib/systemd/system/poweroff.target \
/etc/systemd/system/ctrl-alt-del.target
```

Indítsuk újra a systemd menedzser konfigurálót:

```
# systemctl daemon-reload
```

Szolgáltatás beüzemelése

Készítsük el magát a szolgáltatást:

```
nano /usr/local/bin/pelda.sh
```

A pelda.sh tartalma legyen:

```
#!/bin/bash
while true
do
    echo Az aktuális idő $(date)
    sleep 1
done
```

A systemd-ben felveszem a szolgáltatást:

```
cd /etc/systemd/system
nano pelda.service
```

A fájl tartalma legyen:

```
[Service]
ExecStart=/usr/local/bin/pelda.sh
```

Indítsuk el a szolgáltatást:

```
systemctl start pelda
```

Ellenőrizzük:

```
systemctl status pelda
```

```
  pelda.service
   Loaded: loaded (/etc/systemd/system/pelda.service; static; vendor preset:
   Active: active (running) since Wed 2018-07-04 17:20:56 CEST; 12min ago
 Main PID: 556 (pelda.sh)
    Tasks: 2 (limit: 4915)
   CGroup: /system.slice/pelda.service
...
```

Ellenőrizzük a naplóban a syslog futását:

```
tail /var/log/syslog
```

Ellenőrizzük a systemd naplót:

```
journalctl -u pelda
```

```
journalctl -u pelda -f
```

Nézzük meg a futó folyamatok között:

```
ps axf
```

További beállítások:

```
[Service]
ExecStart=/usr/local/bin/pelda.sh
Restart=always
```

A „Restart” opció újraindítja a szolgáltatás, ha az valamiért leáll. Az összes lehetséges opció: no, always, on-success, on-failure, on-abnormal, on-abort, on-wathcdog.

További hasznos beállítások:

```
[Service]
ExecStart=/usr/local/bin/pelda.sh
Restart=always
WorkingDirectory=/usr/local/bin
User=pelda
Group=pelda
Environment=SAVEDIR=/usr/local/pelda
```

Unit szekció készítése:

```
[Unit]
Description=A pelda szervizem
After=network.target

[Service]
ExecStart=/usr/local/bin/pelda.sh
Restart=always
WorkingDirectory=/usr/local/bin
User=pelda
Group=pelda
Environment=SAVEDIR=/usr/local/pelda
```

Készíthetünk egy leírást. Megadhatjuk milyen szolgáltatás után induljon a mi szolgáltatásunk.

Ha azt szeretnénk, hogy a szolgáltatásunk a rendszer újraindulása után is engedélyezve legyen, akkor szükségünk lesz egy „Install” szekcióra is.

```
[Unit]
Description=A pelda szervizem
After=network.target
```

```
[Service]
ExecStart=/usr/local/bin/pelda.sh
Restart=always
WorkingDirectory=/usr/local/bin
User=pelda
Group=pelda
Environment=SAVEDIR=/usr/local/pelda

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

Így már engedélyezhető a szolgáltatásunk tartósan:

```
systemctl enable pelda
```

Ha utólag szerkesztjük a szervizállományt, akkor a változtatás után szükséges:

```
systemctl daemon-reload
```

Persze lehet így is:

```
systemctl restart pelda
```

A szolgáltatásfájl szerkeszthető a systemctl paranccsal is:

```
systemctl edit pelda --full
```

Ellenőrzések

systemd-analyze

A systemd-analyze parancs segítségével értékelhetjük az utolsó rendszerindulási időket.

```
systemd-analyze
```

Paramétere nélkül, lehetséges kimenet:

```
Startup finished in 1.968s (kernel) + 23.141s (userspace) = 25.110s
```

blame

```
systemd-analyze blame
```

Lehetséges kimenet:

```
          9.311s apt-daily.service
```

```
          6.900s NetworkManager-wait-online.service
          5.066s nmbd.service
          4.281s systemd-udev-settle.service
          1.883s mariadb.service
          1.780s apt-daily-upgrade.service
           710ms wicd.service
           680ms vboxdrv.service
           496ms apache2.service
           445ms loadcpufreq.service
           427ms ModemManager.service
           339ms accounts-daemon.service
           328ms dev-sda1.device
           282ms systemd-logind.service
           272ms speech-dispatcher.service
           270ms lm-sensors.service
           265ms rsyslog.service
           259ms zfs-share.service
           253ms pppd-dns.service
           252ms alsa-restore.service
```

critical-chain

```
systemd-analyze critical-chain
```

Az egység indulásának, aktiválásának kezdetének ideje a @ karakter után van kiírva. Az egység elinduláshoz szükséges idő a + karakter után jelenik meg.

```
graphical.target @17.107s
└─multi-user.target @17.107s
  └─nmbd.service @12.040s +5.066s
    └─network-online.target @12.038s
      └─NetworkManager-wait-online.service @5.137s +6.900s
        └─NetworkManager.service @4.969s +153ms
          └─dbus.service @4.738s
            └─basic.target @4.707s
              └─sockets.target @4.707s
                └─avahi-daemon.socket @4.707s
                  └─sysinit.target @4.702s
                    └─sys-fs-fuse-connections.mount @6.804s +1ms
                      └─systemd-modules-load.service @141ms +46ms
                        └─systemd-journald.socket @139ms
                          └─-.mount @117ms
                            └─system.slice @139ms
                              └─-.slice @117ms
```

Függelék

A Debian GNU/Linux démonkezelése

A Debian GNU/Linux rendszereken továbbra is használhatók a invoke-rc.d és a service parancsok is:

```
invoke-rc.d apache2 stop
invoke-rc.d apache2 start
invoke-rc.d apache2 restart
```

```
service apache2 stop
service apache2 start
service apache2 restart
```

Démon készítés

A „Szolgáltatás beüzemelése” fejezetben egy egyszerű Bash script a szolgáltatás. C nyelven a következő helyen találunk egy egyszerű linuxos démon forráskódot:

- Démon programozása

Forrás

- https://www.digitalocean.com/ 2018
- man systed.service (Debian 9)
- man systemd.unit (Debian 9)
- https://medium.com/@johannes_gehrs/getting-started-with-systemd-on-debian-jessie-e024758ca63d (2018)
- https://wiki.debian.org/systemd (2018)
- https://www.freedesktop.org/wiki/Software/systemd/ (2018)
    - https://www.freedesktop.org/software/systemd/man/index.html (2018)
- Konténerek:
    - https://wiki.debian.org/nspawn (2018)
    - https://wiki.debian.org/FreedomBox (2018)
    - https://wiki.debian.org/Debootstrap (2018)
    - https://blog.selectel.com/systemd-containers-introduction-systemd-nspawn/ (2018)
    - http://trentsonlinedocs.xyz/debian_nspawn_container_on_arch_for_testing_apache_configurations/ (2018)