1/7 2025/10/29 17:59 Systemd

Systemd



Hauseigenes Apt-Repo: https://apt.iteas.at 🔤 🕬 📧

Ein Service bearbeiten und personalisieren.

```
systemctl edit --full rc-local
```

Um den Defaulteditor VI von SystemD zu überschreiben bedient man sich diesem Befehl:

```
EDITOR=nano systemctl edit --full rc-local
```

Hier wird das komplette Service kopiert und wird von Updates des Systems nicht weiterhin berührt. Dies kann je nachdem zu irgendwann auch zu Problemen führen. Deshalb gibt es auch eine andere Variante wo die Files virtuell verschmolzen werden:

```
EDITOR=nano systemctl edit apache2
```

Oder wenn es ein komplett neues Unitfile ist:

```
systemctl edit -f -l rc-local
```

Man könnte auch Dienste direkt in /etc/systemd/system/blabla-custom.service kopieren. Von dem wird abgeraten. Vor allem da viele Services erst von anderen Systemddiensten nur temporär angelegt werden. Das Kommando oben extrahiert die richtigen Files automatisch.

Beispiel Einbau von Sleep beim Start eines Services

[Unit] Description=Puppet agent Requires=network.target [Service] Type=forking EnvironmentFile=-/etc/default/puppet PIDFile=/run/puppet/agent.pid ExecStartPre=/bin/sleep 15 ExecStart=/usr/bin/puppet agent \$DAEMON_OPTS [Install] WantedBy=multi-user.target

Timeout beim Beenden eines Services

Sehr nützlich wenn durch z.B. nicht mehr erreichen von Services wie NB's - WLAN Dienste ihr maximales Timeout erreichen würden.

update: 2023/08/17 server_und_serverdienste:systemd https://www.deepdoc.at/dokuwiki/doku.php?id=server_und_serverdienste:systemd&rev=1692302845

[Unit]

Description=Make remote CUPS printers available locally

After=cups.service avahi-daemon.service

Wants=cups.service avahi-daemon.service

[Service]

TimeoutStopSec=2

ExecStart=/usr/sbin/cups-browsed

[Install]

WantedBy=multi-user.target

Autologin systemd auf der Konsole ohne Displaymanager

Hierzu legt man sich folgendes File an:

nano /etc/systemd/system/getty@tty1.service.d/override.conf

Mit folgenden Inhalt

[Service]
ExecStart=
ExecStart=-/sbin/agetty --autologin xbmc --noclear %I 38400 linux

Hier wird z.B. der xbmc Benutzer automatisch eingeloggt. Danach werden natürlich .zshrc .bashrc und auch die .xinitrc beachtet.

Systemdservices über Remote ausführen

Mit Systemd ist es sehr bequem möglich Dinge zu organisieren ohne das man direkt am Host ist. z.b.

systemctl -H root@myhost.supertux.bla status apache2

NFS-Client

systemctl enable nfs-client.target
systemctl enable rpc-statd.service
systemctl enable rpcbind.service

Mounten mit Systemd - FSTAB ruhe in Frieden

2025/10/29 17:59 3/7 Systemd

Systemd Mount mit NFS

Testsystem: Debian 10/Proxmox 6.3

Die FSTAB ist mehr oder weniger überholt. Benötigt wird sie vom System wohl nur mehr für den Root Mount. Generell wird bereits bei jedem manuellen FSTAB-Eintrag ein Systemd-Unit-File generiert dass auf die FSTAB verweist. Daher wird auch empfohlen statt der FSTAB nur mehr Systemd zu verwenden, was sehr viele Vorteile mit sich bringt. Z.B. hat man damit die Möglichkeit auch Abhängigkeiten von anderen Programmen und States anzugeben. Ein praktischen nerviges Beispiel wäre wenn ein Netzwerkmount nicht immer automatisch in der FSTAB gemountet wird, weil das Netzwerk vielleicht nicht immer gleich schnell verfügbar ist. Und obwohl man in der FSTAB die Option gesetzt hat dass, das Netzwerk verfügbar sein muss, funktioniert es trotzdem doch immer nicht. Systemd schafft hier für dich Abhilfe. Auch Proxmox verwendet den Systemd-Mounter als Default.

Hier als Beispiel ein einfacher Mount einer lokalen HDD. Als erstes legst du ein sogenanntes Unit-File an. Der Mountpoint wird dabei automatisch erstellt.

```
systemctl edit -f -l "/mnt/datastore/HDD-extern-OSIT"
```

Um den Defaulteditor VI von SystemD zu überschreiben bedient man sich diesem Befehl:

```
EDITOR=nano systemctl edit -f -l "/mnt/datastore/HDD-extern-OSIT"
```

Wie du siehst muss der Name der exakte Mountpoint sein. Nun befüllst du das File mit diesem Inhalt:

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

[Unit]
Description=Mount datatstore 'sicherung-osit-extern' under
'/mnt/datastore/HDD-extern-OSIT'

[Mount]
Options=defaults
Type=ext4
What=/dev/disk/by-uuid/d6b3aa86-aa6c-4b41-b6b2-16457820169629
Where=/mnt/datastore/HDD-extern-OSIT
```

Und NFS4:

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

[Unit]
Description=Mount datatstore under /home/mydata
Requires=network.target
Requires=NetworkManager.service
Requires=network-online.target

[Mount]
```

 $\frac{\text{upuate.}}{2023/08/17} server_und_server dienste: systemd \ \text{https://www.deepdoc.at/dokuwiki/doku.php?id=server_und_server dienste: systemd \& rev=1692302845223/08/17$

```
Options=rw,_netdev,auto,acl,exec,intr,bg,nfsvers=4,minorversion=2,x-
systemd.device-timeout=60,x-systemd.mount-timeout=60
Type=nfs4
What=myhostserver.lan:/ssd-pool/mydatastore
Where=/home/mydata
```

Mit dem nächsten Befehl hast eine tolle Übersicht für alle Mountpoints die es gibt, und ob diese im Autostart sind oder nicht.

systemctl list-unit-files -t mount

UNIT FILEmount boot-efi.mount dev-hugepages.mount dev-mqueue.mount mnt-datastore-HDD\x2dextern\x2d0SIT.mount proc-fs-nfsd.mount proc-sys-fs-binfmt_misc.mount run-rpc_pipefs.mount sys-fs-fuse-connections.mount sys-kernel-debug.mount	STATE generated generated static static disabled static static static static static static static
---	---

In den Autostart damit:

```
systemctl enable "mnt-datastore-HDD\x2dextern\x2dOSIT.mount"
```

Und mounten:

```
systemctl start mnt-datastore-HDD\\x2dextern\\x2dOSIT.mount
```

Bei der Mountübersicht sieht das ganze nun so aus:

```
UNIT FILE
                                           STATE
-.mount
                                           generated
boot-efi.mount
                                           generated
dev-hugepages.mount
                                           static
dev-mqueue.mount
                                            static
mnt-datastore-HDD\x2dextern\x2d0SIT.mount enabled
proc-fs-nfsd.mount
                                           static
proc-sys-fs-binfmt misc.mount
                                           static
run-rpc pipefs.mount
                                           static
sys-fs-fuse-connections.mount
                                           static
sys-kernel-config.mount
                                           static
sys-kernel-debug.mount
                                           static
```

Bestehende Unitfiles kann mit dem folgenden Befehlen editieren:

2025/10/29 17:59 5/7 Systemd

systemctl edit -l mnt-datastore-HDD\\x2dextern\\x2dOSIT.mount

oder auch:

systemctl edit -l "/mnt/datastore/HDD-extern-OSIT"

Systemd Mount mit Samba

Testsystem: Ubuntu 20.04, 22.03LTS auf Proxmox LXC

Dies gestaltet sich sehr ähnlich wie NFS. Lediglich ein Paket und die Authentifizierung kommt dazu.

```
apt install cifs-utils -dy
```

Danach erstellen wir unser Unitfile, und aktivieren es:

```
EDITOR=nano systemctl edit -f -l "/data-docs"
```

```
[Unit]
```

Description=samba mount for sambafiles Requires=systemd-networkd.service After=network-online.target Wants=network-online.target

[Mount]

What=//yourserver.lan/data-docs

Where=/media-kodi

Options = credentials = /root/.smbcredentials, auto, vers = 3.0, uid = 2344, gid = 2344, find the continuous continuous

le_mode=0777,dir_mode=0777

Type=cifs

TimeoutSec=30

[Install]

WantedBy=multi-user.target

Nun noch SystemD reloaden und den Mount aktivieren:

```
systemctl daemon-reload
systemctl enable "data\x2ddocs.mount"
```

Nun kann das Systemd-Service getartet werden, und somit wird auch das Laufwerk eingehängt.

```
systemctl start "data\x2ddocs.mount"
```

Für die Erweiterung deines Unitfiles empfehle ich diesen Artikel und auch diesen auf Ubuntuusers.

Systemd Autostart

Hier ein Beispiel für ein WOL Script das beim Boot ausgeführt wird, aber erst wenn der Server online ist.

```
systemctl edit -f -l wol-at-boot.service
```

Inhalt:

```
[Unit]
Description=execute Wake-up on LAN

Wants=network.target
After=syslog.target network-online.target

[Service]
Type=oneshot
ExecStart=/etc/cron.hourly/wol.sh

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

```
systemctl enable wol-at-boot.service
systemctl daemon-reload
```

Debuging

Um z.B. Zeiten beim Systemstart ansehen zu können gibt es zwei nette Befehle:

```
systemd-analyze plot > bootchart.svg
systemd-analyze blame
```

Weitere nützliche Systemd-Befehle

```
systemctl reset-failed
systemctl --failed
```

Links

- Hersteller|Dokumentation Systemd
- Hersteller|Dokumentation Systemd Mount Unit
- https://wiki.ubuntuusers.de/systemd/systemctl/

2025/10/29 17:59 7/7 Systemd

• https://wiki.ubuntuusers.de/systemd/Mount_Units/

From

https://www.deepdoc.at/dokuwiki/ - DEEPDOC.AT - enjoy your brain

Permanent link:

https://www.deepdoc.at/dokuwiki/doku.php?id=server_und_serverdienste:systemd&rev=1692302845

Last update: 2023/08/17 20:07

