Systemd

- Aufbewahrungszeit Journal von Systemd konfigurieren
- Einfache Kommandos beim Systemstart ausführen
- Kernel-Modul beim Systemstart laden und Parameter setzen
- NTP-Client in systemd konfigurieren
- Verzeichnis für PID-File in /run durch systemd anlegen lassen
- Kernel-Setting Transparent Hugepage konfigurieren
- Limits setzen/überschreiben mit systemd

Aufbewahrungszeit Journal von Systemd konfigurieren

> Getestet mit Arch Linux am 25.09.2020

Mit der Zeit wird das Journal (der /var/log/syslog Ersatz von Systemd) immer länger. Es läßt sich aber auch eine Retention konfigurieren, die z.B. die Logs nach einer Woche löscht. Dazu muss die Konfigurationsdatei wie folgt editiert werden:

```
MaxRetentionSec=1week
```

Aus der Manpage zu journald.conf:

```
MaxRetentionSec=
  The maximum time to store journal entries. This controls whether journal files containing entr
  should not be required as size-based deletion with options such as SystemMaxUse= should be suf
  policies, it might make sense to change this value from the default of 0 (which turns off this
  "week", "day", "h" or " m" to override the default time unit of seconds.
```

Es gibt noch ein paar andere Möglichkeiten, z.B. nach einer bestimmten Größe zu rotieren. Diese sind in der Manpage dokumentiert: `man journald.conf`

Einfache Kommandos beim Systemstart ausführen

```
[Unit]
Description=Disable segment offloading
Wants=network-online.target
After=network-online.target

[Service]
Type=oneshot
RemainAfterExit = yes
ExecStart=/usr/sbin/ethtool -K eno1 tso off gso off
ExecStart=/usr/sbin/ethtool -K vmbr0 tso off gso off

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

Beim Typ „oneshot“ können mehrere ExecStart Befehle konfiguriert werden.

Kernel-Modul beim Systemstart laden und Parameter setzen

Nur ein kurzes Beispiel, früher™ wurde das über die rc.local gemacht.

```
[Unit]
Description = Required settings to mount unencrypted cifs shares
After = NetworkManager-wait-online.service network.target network-online.target dbus.service
Wants = NetworkManager-wait-online.service network-online.target

[Service]
Type = oneshot
RemainAfterExit = yes
ExecStart = /bin/bash -c "modprobe cifs; echo 0x27 > /proc/fs/cifs/SecurityFlags"

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

NTP-Client in systemd konfigurieren

Die NTP-Server werden mit Leerzeichen getrennt in der timesyncd.conf hinterlegt.

/etc/systemd/timesyncd.conf

```
[Time]
Servers=0.ntp.wavecon.de 1.ntp.wavecon.de
```

Der Service muss noch aktiviert und gestartet werden

```
systemctl enable systemd-timesyncd.service
systemctl start systemd-timesyncd.service
```

Über `systemctl status systemd-timesyncd.service` kann man dann auch sehen ob er tut was er soll.

Verzeichnis für PID-File in /run durch systemd anlegen lassen

Systemd kann sich um das Anlegen eines Verzeichnisses in /run kümmern, wenn die zu startende Applikation dazu selbst keine Berechtigung hat. Dazu muss der Parameter „RuntimeDirectory“ im Unit-File hinzugefügt werden.

Beispiel für Cerebro (eine Monitoring-GUI für Elasticsearch):

```
[Unit]
Description=Cerebro

[Service]
Type=simple
User=elasticsearch
Group=elasticsearch
RuntimeDirectory=cerebro
ExecStart=/srv/cerebro/current/bin/cerebro "-Dpidfile.path=/var/run/cerebro/cerebro.pid"
Restart=always
WorkingDirectory=/srv/cerebro

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

Das Verzeichnis in /run gehört dem User und der Gruppe, die im Unit-File angegeben sind. Die Modes lassen sich über den Parameter RuntimeDirectoryMode ändern.

Kernel-Setting Transparent Hugepage konfigurieren

Dieses Setting ist wichtig z.B. für MongoDB oder Redis (auch für andere Datenbanken!). Beide Services wollen das gerne auf 'never' haben.

Mit systemd ist das recht einfach. Folgendes in /etc/tmpfiles.d/disable-thp.conf einfügen:

```
#Type Path                                   Mode UID GID  Age Argument
w     /sys/kernel/mm/transparent_hugepage/enabled       -    -   -    -   never
w     /sys/kernel/mm/transparent_hugepage/defrag        -    -   -    -   never
```

und folgendes Kommando ausführen:

```
systemd-tmpfiles --create --prefix=/sys/kernel/mm/transparent_hugepage/ /etc/tmpfiles.d/disable-thp.conf
```

und überprüfen:

```
grep -E . /sys/kernel/mm/transparent_hugepage/defrag /sys/kernel/mm/transparent_hugepage/enabled
```

und hier für schnelles Copy&Paste zusammengefasst

```
cat >/etc/tmpfiles.d/disable-thp.conf <<EOF
#Type Path                                   Mode UID GID  Age Argument
w     /sys/kernel/mm/transparent_hugepage/enabled       -    -   -    -   never
w     /sys/kernel/mm/transparent_hugepage/defrag        -    -   -    -   never
EOF
systemd-tmpfiles --create --prefix=/sys/kernel/mm/transparent_hugepage/ /etc/tmpfiles.d/disable-thp.conf
grep -E . /sys/kernel/mm/transparent_hugepage/defrag /sys/kernel/mm/transparent_hugepage/enabled
```

Limits setzen/überschreiben mit systemd

Wie kann ich Limits für Services setzen, die via systemd gestartet werden? Meine Einstellungen in `/etc/security/limits.conf` oder `/etc/security/limits.d/*.conf` werden ignoriert, da diese nur von pam_limits.so verwendet werden, was systemd nicht nutzt.

Um die Limit anzupassen muss das systemd unit angepasst werden, z.B. für MySQL:

```
$ systemctl edit mysql.service
# einfügen und speichern:
[Service]
LimitNOFILE=500000

# Service neu starten
$ systemctl restart mysql.service
```

Im systemctl status sieht man jetzt den Override

```
$ systemctl status mysql.service
● mysql.service - Percona Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
  Drop-In: /etc/systemd/system/mysql.service.d
           └─override.conf
```

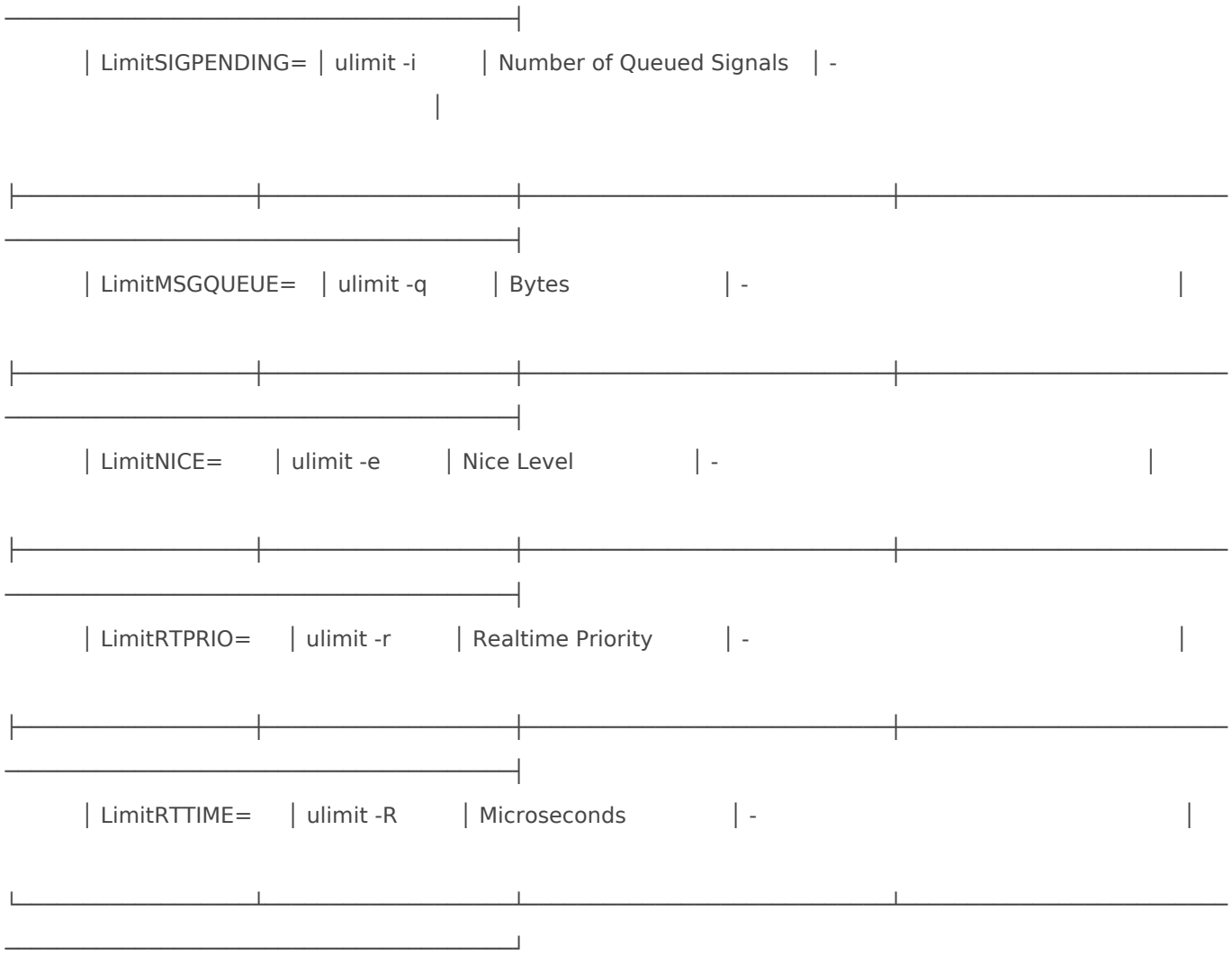
Limits für alle Prozesse überschreiben

```
mkdir -p /etc/systemd/system.conf.d/
cat >/etc/systemd/system.conf.d/10-filelimit.conf <<EOF
[Manager]
DefaultLimitNOFILE=500000
EOF
systemctl daemon-reload
## ggf. Reboot!
```

Folgende Limits können überschrieben werden:

Directive	ulimit equivalent	Unit	Notes
LimitCPU=	ulimit -t	Seconds	-
LimitFSIZE=	ulimit -f	Bytes	-
LimitDATA=	ulimit -d	Bytes	Don't use. This limits the allowed address range, not memory use! Defaults to unlimited and should not be lowered. To limit memory use, see MemoryMax= in systemd.resource-control(5).
LimitSTACK=	ulimit -s	Bytes	-
LimitCORE=	ulimit -c	Bytes	-
LimitRSS=	ulimit -m	Bytes	Don't use. No effect on Linux.
LimitNOFILE=	ulimit -n	Number of File Descriptors	Don't use. Be careful when raising the soft limit above 1024, since select(2) cannot function with file descriptors above 1023 on Linux. Nowadays, the hard limit defaults to 524288, a

			very high value compared to historical defaults. Typically
			applications should increase their soft limit to the hard limit on their own, if they are OK with working with file descriptors above 1023, i.e. do not use select(2). Note that file descriptors are nowadays accounted like any other form of memory, thus there should not be any need to lower the hard limit. Use MemoryMax= to control overall service memory use, including file descriptor memory.
LimitAS=	ulimit -v	Bytes	Don't use. This limits the allowed address range, not memory use! Defaults to unlimited and should not be lowered. To limit memory use, see MemoryMax= in systemd.resource-control(5).
LimitNPROC=	ulimit -u	Number of Processes	This limit is enforced based on the number of processes belonging to the user. Typically it's better to track processes per service, i.e. use TasksMax=, see systemd.resource-control(5).
LimitMEMLOCK=	ulimit -l	Bytes	-
LimitLOCKS=	ulimit -x	Number of Locks	-



Weitere Infos in den manpages

```
man 5 systemd.exec
man 5 systemd.resource-control
```