

Linux Basis Systeembeheer

D. Leeuw

20 december 2023

v.0.4.0



© 2021-2023 Dennis Leeuw

Dit werk is uitgegeven onder de Creative Commons BY-NC-SA Licentie en laat anderen toe het werk te kopiëren, distribueren, vertonen, op te voeren, en om afgeleid materiaal te maken, zolang de auteurs en uitgever worden vermeld als maker van het werk, het werk niet commercieel gebruikt wordt en afgeleide werken onder identieke voorwaarden worden verspreid.

Over dit Document

Inhoudsopgave

Over dit Document	i
1 Booting Linux en /boot/	1
1.1 GRUB Configuratie	1
1.2 Kernel boot bestanden	2
2 Werken met modules	3
3 Werken met daemons	5
3.1 init en systemd	5
3.2 Unit files	6
3.3 systemctl	7
4 Logging	9
4.1 Log rotation	9
4.2 syslog	10
4.3 journalctl	11
5 Monitoring services	13
5.1 Proces monitoring	13
5.2 Daemons en netwerken	14
6 Troubleshooting	17
Index	19

Hoofdstuk 1

Booting Linux en `/boot/`

Nadat het BIOS klaar is met zijn taken wil deze een boot-loader laden. Vroeger was dit LiLo, de Linux Loader, maar tegenwoordig wordt GRUB, GRand Unified Bootloader, gebruikt. GRUB kan verschillende operating systems laden en kan dus gebruikt worden als universele bootloader om zowel Linux als Windows te starten. Zo kan je op één machine Windows en Linux hebben staan en kan je door te herstarten kiezen welk OS je wil gebruiken.

De bootloader laadt de kernel, Linux, welke ervoor zorgt dat er tegen de hardware gesproken kan worden. De taken van de kernel zijn:

- het beheren van het fysieke geheugen en daarbij behorend het opdelen van het virtuele geheugen voor proces isolatie
- zorgen dat zowel processen als hardware de resources krijgen waarom ze vragen (interrupts), als dat mag
- het serialiseren van de taken die uitgevoerd moeten worden, ofwel proces management (schedulig)

Als de kernel zijn taken verder op de achtergrond kan uitvoeren kan proces nummer 1 gestart worden om het verdere bootproces af te maken, meestal eindigend in een grafische of een command line interface, waarna de gebruiker(s) aanzet zijn.

Als je een `ls` doet van de `/boot/` directory dan vinden we daarin de belangrijkste bestanden die nodig zijn om een Linux systeem te op te kunnen starten.

1.1 GRUB Configuratie

De configuratie van GRUB staat in `/etc/` en in `/boot/grub/` (soms in `/boot/grub2/` of een ander versienummer). De GRUB bestanden in `/boot/`

zijn gegenereerde bestanden en daar zou je nooit zelf wijzigingen in moeten aanbrengen. De wijzigingen worden aangebracht in `/etc/default/grub` en de snippets in `/etc/grub.d`. Een script `grub-mkconfig` maakt van dit geheel de configuratie in `/boot/`.

1.2 Kernel boot bestanden

De kernel, Linux, bestaat uit een modulaire kernel. Er is een stuk basis software die geladen wordt tijdens het opstarten en deze software kan uitgebreid worden met modules. Modules voegen extra functionaliteit toe, zoals bij voorbeeld netwerk-drivers. De reden dat er gekozen voor een modulaair design is dat de basis kernel daardoor klein kan zijn en minder geheugen inneemt. Alleen de drivers of andere functionaliteit die echt nodig is wordt geladen.

De basis kernel ook bekend als `vmlinuz` kan gevonden worden in de `/boot/` directory, in het verleden werd de kernel in de `/` directory gezet en kan daar op sommige systemen nog gevonden worden. In de `/boot/` directory kan je ook bestanden vinden die beginnen met `vmlinuz` en waarna er een serienummer volgt, zo kunnen er meerdere versies van de kernel staan, waaruit eventueel via de bootloader gekozen kan worden.

Naast het bestand `vmlinuz` staan er nog een paar bestanden die bij de kernel horen:

config is de configuratie van de kernel zoals deze gebruikt is tijdens het compilen van de kernel. Het beschrijft o.a. welke stukken van de kernel als module gebouwd moeten worden. Dit document is niet belangrijk voor het opstarten van het systeem.

initrd.img Ook de kernel heeft tijdens het opstarten modules nodig, denk daarbij aan de netwerk-drivers. De `initrd.img` bevat deze drivers en deze image wordt in RAM geladen zodat de modules snel beschikbaar zijn (ipv. dat ze van een trage harddisk moeten komen)

System.map bevat een overzicht van welke functies of variabelen zich op welke lokatie bevinden in de kernel. Dit bestand wordt gebruikt om lookups te doen als er een kernel oops of kernel panic zich voordoet.

Hoofdstuk 2

Werken met modules

De modules die bij de kernel horen kunnen gevonden worden in de directories onder `/lib/modules/`. De modules die behoren bij de draaiende kernel kunnen gevonden worden via:

```
$ ls /lib/modules/$(uname -r)/kernel/
```

Modules kunnen on-the-fly geladen worden. Een overzicht van de reeds geladen modules kan verkregen worden met :

```
$ lsmod
```

Om te zorgen dat onze kernel kan omgaan met het FAT bestandssysteem gaan we de fat-modules toevoegen aan de draaiende kernel. Dit mag natuurlijk alleen root doen.

```
$ sudo modprobe fat
```

Bij geen error-melding is alles goed gegaan. Dit kunnen we controleren

```
$ lsmod | grep fat
```

Na deze handeling zouden we bijvoorbeeld een USB-stick met een FAT bestandssysteem kunnen lezen.

Het verwijderen van een module is zo simpel als aan `modprobe` vertellen met de `-r` optie dat het een module moet verwijderen

```
$ sudo modprobe -r fat
```


Hoofdstuk 3

Werken met daemons

In de Unix wereld zeggen we altijd 'Everything is a file and if it is not a file it is a process'.

Processen die op de achtergrond draaien heten daemons in de Unix-wereld. Ze zijn ook bekend als services, zeker in de Windows wereld. Daemons zijn dus processen waar je weinig van merkt en die rustig hun werk staan te doen. Deze processen draaien zonder dat ze input van de gebruiker nodig hebben.

Bekende daemons zijn bijvoorbeeld web- of mailservers, maar het kan ook een printserver zijn. Op een Linux-machine, ongeacht of het een desktop of een server is, draaien verschillende daemons in de achtergrond.

3.1 init en systemd

Nadat de kernel in het geheugen geladen is zal deze gestart worden. De Linux kernel zorgt ervoor dat de beschikbare hardware klaar is voor gebruik en dat er processen gestart kunnen worden. Het eerste proces dat de kernel start is **systemd**. Vroeger was dit **init** en dat kan je op vele ander Unix-achtige besturingssystemen nog tegen komen, maar de meeste Linux distributies zijn over naar **systemd**.

systemd is het proces dat ervoor zorgt dat alle ander processen gestart worden. De **systemd** daemon heeft proces nummer 1. Het is de eerste daemon die start bij het opstarten van het systeem en de laatste die afgesloten wordt bij het afsluiten van het systeem.

Het proces met nummer 1 wordt ook wel "The mother of all processes" genoemd. Nadat de kernel klaar is met opstarten tijdens het boot-proces start het dit eerste proces op. Proces nummer 1 is daarna verantwoordelijk voor het verdere opstart proces van het systeem. Het is dan ook proces 1

die bepaalt of er bijvoorbeeld een grafische interface wordt gestart, maar ook welke functies een machine kan aanbieden nadat het opgestart is. Is de machine een mail-server of een web-server, of misschien wel helemaal geen server.

3.2 Unit files

Om een service te starten of te stoppen gebruikt **systemd** bestanden waarin beschreven wordt hoe een service heet en welke commando's er nodig zijn om te starten en stoppen. Deze beschrijving van een service wordt binnen **systemd** een **unit** genoemd. Units kan je onder andere terug vinden in de `/etc/systemd/system` directory. Een voorbeeld van een unit-file in Debian 11 is de `syslog.service`. Deze ziet er zo uit:

```
[Unit]
Description=System Logging Service
Requires=syslog.socket
Documentation=man:rsyslogd(8)
Documentation=man:rsyslog.conf(5)
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
```

Onder het kopje **Unit** vind je een beschrijving van de service en wat deze nodig heeft om te kunnen werken. Bij *Requires* staat dat **syslog** een **syslog.socket** nodig heeft. Voor nu gaan we ervan uit dat het systeem dit oplost. Wij gaan verder naar de **Service** kop.

Bij *ExecStart* staat het commando dat gebruikt moet worden op de service op te starten. Zoals je kan zien is er geen commando gegeven om de service te stoppen. Een Unix-achtig systeem heeft de **kill** functie om om te gaan met processen (zie Linux CLI documentatie). Deze kan gebruikt worden door **systemd** om processen te stoppen. De *Type* met als waarde **Notify** zegt dat

de daemon aan systemd laat weten wanneer het klaar is met opstarten. Dit heeft als voordeel dat als **systemd** dit signaal krijgt dat het door kan met andere zaken te doen en niet langer hoeft te wachten. Tot slot is er in deze sectie nog de *Restart* optie die we willen behandelen. Het geeft niet aan wat we moeten doen bij een restart, maar wanneer we moeten restarten. In dit geval als **rsyslogd** crashed of elke andere willekeurig melding van een error geeft dan wordt het proces geherstart.

Alle Unit-files zijn platte tekst bestanden en kunnen dus met een willekeurige editor aangepast worden. Als je een bestand (service) hebt gewijzigd of toegevoegd moet je **systemctl** vertellen dat er iets nieuws is. Dat doe je met het volgende commando:

```
$ sudo systemctl daemon-reload
```

Hierna kan je de nieuwe of aangepaste service stoppen, starten of herstarten met de gewijzigde settings.

3.3 systemctl

Na het opstarten van een systeem kan het soms nodig zijn met bestaande processen te stoppen of herstarten omdat er bijvoorbeeld een wijziging in de configuratie is gebeurd, of omdat bepaalde functionaliteit niet langer via deze machine aangeboden wordt. Het kan ook voorkomen dat nieuwe functionaliteit is toegevoegd en dat er een nieuw proces gestart moet worden. Het is dus van belang dat we **systemd** kunnen vertellen om deze acties uit te voeren. Het commando dat doorvoor beschikbaar is heet **systemctl**. **systemctl** kent een aantal standaard commando's die we kunnen meegeven. Voor een uitleg van de syntax van het commando verwijzen we naar de man-page. Wij behandelen hier alleen de belangrijkste commando's:

start Start een service

stop Stop een service

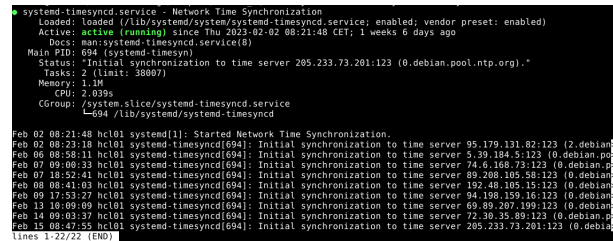
restart Stop een service en als dat klaar is start het weer

reload Vraag een service om zijn configuratie opnieuw te laden, kan niet met alle daemons. Dit is niet de Unit-configuratie, maar de configuratie van de daemon zelf.

Om wat ervaring op te doen met **systemctl** gaan we wat oefenen met een daemon die standaard met de installatie is meegekomen en waar we wat mee kunnen spelen zonder dat het meteen destructief is voor het systeem. De

standaard met de installatie meegekomen daemon is de `systemd-timesyncd` daemon. We gaan deze daemon gebruiken om een beetje vertrouwd te raken met `systemctl`.

```
sudo systemctl status systemd-timesyncd
```



```
systemd-timesyncd.service - Network Time Synchronization
Loaded: loaded (/lib/systemd/system/systemd-timesyncd.service; enabled; vendor preset: enabled)
Active: active (running) since Thu 2023-02-02 08:21:48 CET; 1 weeks 6 days ago
Docs: man:systemd-timesyncd.service(8)
Main PID: 694 (systemd-timesyn)
Status: Initial synchronization to time server 205.233.73.201:123 (0.debian.pool.ntp.org).
Tasks: 2 (limit: 38807)
Memory: 1.1M
CPU: 2.039s
CGroup: /system.slice/systemd-timesyncd.service
        └─694 /lib/systemd/systemd-timesyncd

Feb 02 08:21:48 hcl01 systemd[1]: Started Network Time Synchronization.
Feb 02 08:23:18 hcl01 systemd-timesyncd[694]: Initial synchronization to time server 95.179.131.82:123 (2.debian.pool.ntp.org).
Feb 06 08:38:11 hcl01 systemd-timesyncd[694]: Initial synchronization to time server 5.39.184.5:123 (0.debian.pool.ntp.org).
Feb 07 09:08:33 hcl01 systemd-timesyncd[694]: Initial synchronization to time server 74.6.168.73:123 (0.debian.pool.ntp.org).
Feb 07 18:52:41 hcl01 systemd-timesyncd[694]: Initial synchronization to time server 89.208.105.58:123 (0.debian.pool.ntp.org).
Feb 08 08:41:03 hcl01 systemd-timesyncd[694]: Initial synchronization to time server 192.48.105.15:123 (0.debian.pool.ntp.org).
Feb 09 17:53:27 hcl01 systemd-timesyncd[694]: Initial synchronization to time server 94.198.159.16:123 (0.debian.pool.ntp.org).
Feb 13 18:09:09 hcl01 systemd-timesyncd[694]: Initial synchronization to time server 69.89.207.109:123 (0.debian.pool.ntp.org).
Feb 14 09:03:37 hcl01 systemd-timesyncd[694]: Initial synchronization to time server 72.30.35.89:123 (0.debian.pool.ntp.org).
Feb 15 08:47:53 hcl01 systemd-timesyncd[694]: Initial synchronization to time server 205.233.73.201:123 (0.debian.pool.ntp.org).
times 1:22:22 (430)
```

Figuur 3.1: Status output van `systemctl`

Als je net als in het voorbeeld (Figuur 3.1) een lijn hebt met `END` dan kan je gebruik maken van `'q'` om weer op de command-prompt terecht te komen.

Door aan `systemctl` de commando's `stop` of `start` mee te geven kunnen we daemons op ons systeem stoppen en starten. De tijd synchronisatie daemon is op dit moment gestart, dus het eerste wat we kunnen doen is hem stoppen:

```
sudo systemctl stop systemd-timesyncd
```

Met `systemctl status` kunnen nu de status van de daemon zien en dan zien we dat deze gestopt is. Het opnieuw opstarten doen we met `start`:

```
sudo systemctl start systemd-timesyncd
```

Probeer nu zelf de herstart en reload functies uit. Wat neem je waar?

Hoofdstuk 4

Logging

Omdat daemons op de achtergrond draaien en geen direct contact met een gebruiker of met root hebben zullen ze op een andere manier moeten laten weten of het goed met ze gaat, wat ze aan het doen zijn en of de beheerder misschien moet ingrijpen omdat er iets fout gaat. Daemons laten ons weten wat er aan de hand is door hun berichten te loggen. Loggen is het wegschrijven van de berichten naar een logbestand.

Linux systemen verzamelen alle log bestanden in een directory `/var/log`. Een daemon kan zelfstandig zijn eigen log wegschrijven naar een bestand, of de daemon kan gebruik maken van een log-server. Het voordeel van een log-server is dat de daemon alleen de API van de log-server hoeft te kennen. Alle andere logica rond het loggen wordt afgehandeld door de log-server. Voor meer informatie over de log-server zie de sectie over syslog (4.2).

Om direct te kunnen volgen wat er op een systeem gebeurt wordt veel het `tail` commando gebruikt. `tail` heeft een optie `-f` die follow betekent. Het blijft volgen welke nieuwe regels er aan een bestand worden toegevoegd. Een manier om bijvoorbeeld de `messages` log te blijven volgen is:

```
$ sudo tail -f /var/log/messages
```

Omdat alle log-bestanden tekst bestanden zijn kan je ook bijvoorbeeld `grep` erop los laten en zo zoeken naar bepaalde patronen in de log-bestanden.

4.1 Log rotation

Als applicaties berichten loggen naar bestanden dan worden de log bestanden steeds langer en daarmee het systeem steeds trager. Dat is niet handig. Eens in de zoveel tijd moet een log-bestand dan ook geleegd worden, maar we willen de oude logs niet kwijt raken. Het proces dat ervoor zorgt dat logs op

de juiste manier behandeld worden zodat ze niet te groot worden heet log rotate.

Er zijn verschillende manieren om het roteren van logs voor elkaar te krijgen. Natuurlijk kunnen we het handmatig doen. We moeten dan:

1. de daemon stoppen,
2. het log-bestand verplaatsen naar een andere naam (bijvoorbeeld, log-bestand.backup),
3. een nieuw leeg log-bestand aanmaken met de juiste rechten,
4. de daemon opstarten.

Dit is een hoop werk en tijdens het werk staat de daemon uit. Tevens is het elke keer dezelfde handeling, dus een prima klus om te automatiseren.

We kunnen een script schrijven dat dit doet en dat via cron laten draaien. Een andere optie is dat we een tool gebruiken die het voor ons regelt zoals bijvoorbeeld `logrotate`.

4.2 syslog

Het proces dat op de achtergrond draait en dat verantwoordelijk is voor het schrijven van de verschillende logbestanden heet syslog, een afkorting van system logging. Het gebruik van syslog heeft als groot voordeel dat niet elke software ontwikkelaar alle code hoeft te schrijven om logbestanden te schrijven, hij hoeft alleen maar te weten hoe hij data moet aanlever bij de syslog server. Voor de systeembeheerder is het makkelijk omdat hij niet elk programma hoeft te vertellen waar deze zijn meldingen weg moet schrijven, die hoeft alleen maar de syslog-server te beheren en het laatste voordeel is dat de output van syslog altijd dezelfde is, dus de logbestanden worden nog makkelijker leesbaar ook.

Vele Linux distributies gebruiken rsyslog als hun syslog server. De belangrijkste logbestanden in de `/var/log` die beheerd worden door de syslog server zijn:

`messages` Alle logs

`mail` (**SuSE**) `maillog` (**RedHat**) of `mail.log` (**Debian**) Mail logging

`mail.info` Mail info messages

`mail.warn` Mail warning messages

`mail.err` Mail error messages

`daemon.log` (**Debian**) Berichten van Daemons

`auth.log` (**Debian**) of `secure` (**RedHat**) Authenticatie gerelateerde berichten

4.3 journalctl

`systemd` is het eerste proces (daemon) dat opgestart wordt op een Linux systeem. Het is daarmee de moeder van alle processen. Als er een daemon opgestart moet worden dan wordt dat gedaan via `systemd`. `systemd` is dus de plek waar ook in de gaten gehouden kan worden wat er opgestart is en wat niet en daarmee kan `systemd` een bijzondere inkijk geven in de status van een systeem. Om meer inzicht te krijgen in wat `systemd` allemaal voorbij heeft zien komen is er een tool met de naam `journalctl`.

Het opstarten van `journalctl` zonder enige opties of argumenten laat zien wat `systemd` heeft verzameld sinds het opstarten van het systeem, of sinds de laatste logrotate.

Om alleen de berichten te zien van een specifiek proces kunnen we het volgende commando gebruiken:

```
$ sudo journalctl -f -u systemd-journald
```

De `-f` optie doet hetzelfde als bij `tail`, het zegt dat de log gevolgd (follow) moet worden. Met `-u` geef je op welke 'unit' er gemonitord moet worden.

Hoofdstuk 5

Monitoring services

In de sectie over `journalctl` hebben we gezien hoe we informatie over een proces kunnen opvragen. We kunnen zien of een daemon gestart of gestopt is. Daemons zijn processen die op de achtergrond draaien en die allerlei taken kunnen uitvoeren. Zo is `cron` een daemon, maar een daemon kan ook een mail-server of web-server zijn. Deze laatste twee luisteren op het netwerk naar inkomende verbindingen. De mail-server doet dat voor inkomende mail op poort 25 en de web-server op poort 80. Het is handig als we kunnen zien welke processen er draaien en wat ze aan het doen zijn. Dit hoofdstuk beschrijft enkele commando's die we daarvoor kunnen gebruiken.

5.1 Proces monitoring

In het document 'Linux CLI' hebben we `ps` behandeld. Met `ps` kan je zien welke processen er draaien op een systeem. In deze sectie gaan we `ps` met andere opties gebruiken dan we eerder hebben gedaan. De Linux variant van `ps` kent de mogelijkheid om ook opties te gebruiken zonder gebruik van het min teken voor de optie, dit is de zogenaamde BSD-style:

```
$ ps aux
```

doet bijna hetzelfde als `ps -ef`. De output ziet er ook bijna hetzelfde uit:

```
$ ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME
COMMAND
root         1  0.0  0.0 167316 11924 ?        Ss   Feb02   0:48 /sbin/init
root         2  0.0  0.0   0     0 ?        S    Feb02   0:00 [kthreadd]
root         3  0.0  0.0   0     0 ?        I<   Feb02   0:00 [rcu_gp]
root         4  0.0  0.0   0     0 ?        I<   Feb02   0:00 [rcu_par_gp]
root         9  0.0  0.0   0     0 ?        I<   Feb02   0:00 [mm_percpu_wq]
root        10  0.0  0.0   0     0 ?        S    Feb02   0:00 [rcu_tasks_rude_]
```

root	11	0.0	0.0	0	0 ?	S	Feb02	0:00	[rcu_tasks_trace]
root	12	0.0	0.0	0	0 ?	S	Feb02	1:25	[ksoftirqd/0]
root	13	0.0	0.0	0	0 ?	I	Feb02	6:07	[rcu_sched]
root	14	0.0	0.0	0	0 ?	S	Feb02	0:03	[migration/0]
root	16	0.0	0.0	0	0 ?	S	Feb02	0:00	[cpuhp/0]
root	17	0.0	0.0	0	0 ?	S	Feb02	0:00	[cpuhp/1]
root	18	0.0	0.0	0	0 ?	S	Feb02	0:03	[migration/1]
root	19	0.0	0.0	0	0 ?	S	Feb02	0:10	[ksoftirqd/1]

De extra informatie is voornamelijk de extra data over het gebruik van de CPU en van het geheugen (MEM). Deze informatie kennen we ook uit de **top** applicatie. Ook dit commando is behandeld in de 'Linux CLI' handleiding.

5.2 Daemons en netwerken

Veel daemons worden gebruikt om diensten aan het netwerk aan te bieden. Om een dienst aan te kunnen bieden moet een daemon luisteren op een port en zodra er een verbinding wordt gemaakt op de port moet de daemon een nieuw proces laten luisteren. Veel daemons kan je vertellen op welke interface of welk IP-adres ze moeten luisteren, zo kan je er bijvoorbeeld voor zorgen dat een MySQL server alleen luistert op de interface van localhost zodat hij niet van buitenaf beschikbaar is.

Als we een daemon opstarten willen we kunnen controleren of de daemon daadwerkelijk luistert op een interface. Van oudsher werd hiervoor het **netstat** commando gebruikt, nieuwere systemen hebben meestal het **ss** commando. Beide commando's hebben de dezelfde functionaliteit. Je kan ze gebruiken om te zien welk proces er luistert op welke interface en je kan zien welke verbindingen er op dit moment gaande zijn. We zullen voor de compleetheid beide commando's laten zien.

Als een daemon luistert op een poort en we willen kunnen zien of dat daadwerkelijk gebeurt dan moeten we de commando's vragen om ons de LISTENING status te laten zien:

```
$ netstat -l
```

```
$ ss -l
```

Als je beide commando's geïnstalleerd hebt staan dan zal je zien de de output iets verschillend is.

Er luisteren over het algemeen veel processen naar allerlei verschillende zaken. Vaak willen we alleen weten of ze op TCP of UDP luisteren. Met de **-u** optie kunnen we kiezen voor UDP en met **-t** voor TCP. We kunnen ze ook tesamen gebruiken om zowel de TCP als de UDP porten te laten zien:

```
$ netstat -lut
```

```
$ ss -lut
```

Je ziet dat voor beide commando's de opties identiek zijn.

Tot slot van het bekijken van de luisterende poorten willen we ook kunnen zien welk proces (welke daemon) de poort gebruikt. Dit kan alleen root zien, dus moeten we sudo gebruiken:

```
$ sudo netstat -tulp
```

```
$ sudo ss -tulp
```

Beide commando's kunnen ook gebruikt worden om te zien welke verbindingen er nu open staan met je computer:

```
$ sudo netstat -tuanp | grep -vE 'LISTEN|:\*'
```

```
$ sudo ss -tu -o state connected
```

Gebruik de man-page van de verschillende commando's om te zien wat de opties betekenen.

Hoofdstuk 6

Troubleshooting

De basis van problemen zoeken op een Linux systeem zijn in de voorgaande hoofdstukken behandeld. Vooral de log bestanden zijn altijd een rijke bron van informatie. Soms is de error melding die je krijgt niet helemaal duidelijk, het is dan handig om de melding letterlijk te knippen en plakken in een Internet zoekmachine. Linux en bijna alle programma's die erop draaien zijn open source en het beheer en troubleshooten van software uit de open source wereld wordt op Internet gedaan, daarom zijn veel van de problemen dan ook op het Internet te vinden, vaak met de bij behorende oplossing. Er zijn vele sites met vele forums en soms worden vragen niet in het juiste forum gesteld. Het kan dus zijn dat de eerste hit van de zoekmachine niet de juiste is en dat het antwoord niet tot een juiste oplossing leidt. Zoek dan nog even verder. Het kan ook gebeuren dat een bepaalde oplossing voor een bepaalde versie werkt, maar niet voor een nieuwere versie. Let dus ook altijd goed op de datum dat bij een antwoord staat, nieuwer is vaak beter.

Tikfouten in configuratie bestanden zijn waarschijnlijk de meeste voorkomende beginnersfouten op Linux. Het gebruik van enkele-quotes of dubbele-quotes, het vergeten van een ; of het perongeluk vervangen door een : zijn ook bekende voorbeelden van fouten. Zeker het blind knippen en plakken van code of commando's kan leiden tot bijzondere effecten, zeker als er niet op het font gelet wordt.

Maar misschien wel de belangrijkste vaardigheid bij het zoeken van problemen is het goed lezen en de tijd nemen om te begrijpen wat er in een foutmelding staat. Linux en open source in zijn algemeenheid wordt geschreven door mensen. Het zijn mensen die wereldwijd samenwerken ieder met zijn eigen culturele en eigen taal achtergrond. Dat vertaalt zich ook in de manier waarop fouten gemeld worden aan de gebruiker. Een error-melding is vervelend en soms moeilijk te lezen, maar erger zou het zijn als er geen melding werd gegeven.

Index

/boot/, 2
/lib/modules/, 3
init, 5
journalctl, 11
kernel, 2
log rotate, 10
log-server, 9
loggen, 9
logging, 9
logrotate, 10
lsmod, 3
modprobe, 3
modules, 2
netstat, 14
probleem zoeken, 17
ps, 13
rsyslog, 10
ss, 14
syslog, 10
systemctl, 7
systemd, 5, 11
 timesyncd, 8
timesyncd, 8
troubleshooting, 17
vmlinuz, 2