Linux kerneld mini-HOWTO

Henrik Storner

kerneld-howto@linuxdoc.org

Vertaald door: Ellen Bokhorst

bokkie@nl.linux.org

Table of Contents

1. Over de kerneld mini-HOWTO

In dit document wordt uitgelegd hoe de automatische kernel module loader "kerneld" te installeren en gebruiken. De laatst uitgegeven versie van dit document is te vinden bij het Linux Documentatie Project (http://www.linuxdoc.org)

1.1. Krediet

Dit document is gebaseerd op een oorspronkelijke HTML versie 1.7 gedateerd 19 juli 1997 door Henrik Storner <`storner@osiris.ping.dk`> en het werd gereviseerd en omgezet volgens de DocBook DTD door Gary Lawrence Murphy <`garym@teledyn.com`> 20 mei 2000.

De volgende mensen hebben op een of andere wijze een bijdrage aan deze mini-HOWTO geleverd:

- Bjorn Ekwall bj0rn@blox.se
- Ben Galliart bgallia@luc.edu
- Cedric Tefft cedric@earthling.net
- Brian Miller bmiller@netspace.net.au
- James C. Tsiao jtsiao@madoka.jpl.nasa.gov

Stuur alsjeblieft een e-mail naar <`kerneld-howto@linuxdoc.org`> als je fouten aantreft in dit document. Je opmerkingen, aanmoedigingen, en suggesties zijn welkom en worden gewaardeerd, en zijn van hulp bij het garanderen dat deze handleiding actueel en accuraat blijft.

2. Wat is kerneld?

De kerneld feature werd geïntroduceerd tijdens de 1.3 ontwikkelaarskernels door Bjorn Ekwall. Je kunt er kernelmodules, zoals devicedrivers, netwerkdrivers en bestandssystemen automatisch mee laden wanneer ze nodig zijn, in plaats van dat je dit handmatig moet doen met modprobe of insmod.

En voor de wat amusantere aspecten, alhoewel deze (nog?) niet zijn geïntegreerd in de standaardkernel:

- Het kan zo worden ingesteld dat het een gebruikersprogramma uitvoert in plaats van de standaard schermbeveiliging, dus dat het je elk programma als screen-saver laat gebruiken.
- Vergelijkbaar met de screen-blanker ondersteuning, kun je ook de standaard console beep in iets totaal anders wijzigen.

kerneld bestaat uit twee componenten:

- Ondersteuning in de Linux kernel voor het versturen van verzoeken naar een daemon vragend om een een module voor een bepaalde taak.
- Een user-space daemon die uit kan zoeken welke modules moeten worden geladen om aan het verzoek van de kernel te voldoen.

Beide componenten moeten werken wil de kerneld ondersteuning functioneren; het is niet genoeg dat slechts de een of de ander is ingesteld.

2.1. Waarom zou ik het willen gebruiken?

Er zijn een aantal goede redenen waarom je kerneld zou willen gebruiken. De redenen die ik hier benoem, zijn mijn redenen, anderen hebben weer andere redenen.

- Als je kernels hebt gebouwd voor verscheidene systemen met slechts kleine verschillen, zoals een ander soort netwerkkaart, bijvoorbeeld, dan kun je een enkele kernel bouwen met een aantal modules in plaats van individuele kernels voor elk systeem.
- Modules zijn voor ontwikkelaars eenvoudiger te testen. Je hoeft het systeem niet opnieuw op te starten om de driver te laden of uit het geheugen te laten verwijderen; dit geldt voor alle modules, niet alleen voor via kerneld geladen modules.
- Het scheelt je in geheugen voor de kernel, waardoor er meer geheugen overblijft voor applicaties. Geheugen dat door de kernel wordt gebruikt, wordt nooit naar disk geswapt, dus als je voor 100Kb aan ongebruikte drivers in je kernel hebt gecompileerd, dan is dat simpelweg een verspilling van RAM.
- Een aantal dingen die ik gebruik zoals bijvoorbeeld de ftape floppy-tape driver of iBCS, zijn alleen beschikbaar als modules, maar ik wil me niet druk hoeven maken om het laden of weer uit het geheugen verwijderen van deze modules wanneer ik ze nodig heb.
- Mensen die Linux distributies samenstellen hoeven geen 284 verschillende bootimages te bouwen: Elke gebruiker laadt de modules die hij nodig heeft voor zijn hardware. De meeste moderne Linux distributies zullen je hardware detecteren en zullen alleen die modules laden die werkelijk nodig zijn.

Natuurlijk zijn er ook redenen waarom je ze niet wilt gebruiken. Als je het gebruik van slechts één kernel image bestand waar alle drivers zijn ingebouwd prefereert, dan lees je het verkeerde document.

2.2. Waar haal ik de benodigde onderdelen vandaan? ?

De ondersteuning in de Linux kernel werd bij Linux 1.3.57 geïntroduceerd. Als je een eerdere kernelversie hebt, dan zul je moeten upgraden als je de kerneld ondersteuning wilt. De huidige Linux kernelsources zijn te vinden op de meeste Linux FTP archiefsites waaronder:

- Kernel.Org archief (ftp://ftp.kernel.org/pub/linux/kernel/)
- Metalab Linux archief (ftp://metalab.unc.edu/pub/Linux/kernel/)
- TSX-11 op MIT (ftp://tsx-11.mit.edu/pub/linux/sources/system/)

De user-space daemon is opgenomen in het modules package. Deze zijn normaal gesproken vanaf dezelfde plaats beschikbaar als de kernelsources.

Als je het module laden wilt proberen met de laatste *ontwikkelaars* kernels, dan moet je het nieuwere modutils package gebruiken en niet de modules. Bekijk altijd het `Documentation/Changes` bestand in de kernel sources voor het minimum vereiste versienummer voor je kernelimage. Zie ook over de problemen die zich voordeden met modules en 2.1 kernels.

3. Hoe stel ik het in?

Zorg eerst dat je aan de benodigde onderdelen komt: een geschikte kernel en het laatste modules package. Dan moet je de module utility's volgens de instructies in het package installeren. Tamelijk simpel: Pak gewoon de broncode uit en start make install. Hiermee worden de volgende programma's in `/sbin` geïnstalleerd: genksysm, insmod, lsmod, modprobe, depmod en kerneld. Ik raad je aan wat regels toe te voegen aan je opstartscripts om de benodigde setup uit te voeren op het moment dat je Linux boot. Voeg de volgende regels toe aan het bestand `/etc/rc.d/rc.S` (als je Slackware draait) of aan `/etc/rc.d/rc.sysinit` als je SysVinit, d.w.z. Debian, Corel, RedHat, Mandrake of Caldera draait:

```
        # Start kerneld - dit moet zeer vroeg gebeuren in het bootproces,
 # zeker VOORDAT je fsck op bestandssystemen uitvoert
        # waarvoor wellicht automatisch diskdrivers moeten worden geladen
        if [ -x /sbin/kerneld ]
        then
                /sbin/kerneld
        fi

        # Hier komen de standaard fsck opdrachten
        # En de mount opdracht om het root fs read-write te mounten

        # Update kernel-module dependencies file
        # Je root-fs MOET nu read-write zijn gemount
        if [ -x /sbin/depmod ]
        then
                /sbin/depmod -a
        fi
```

Deze opdrachten kunnen reeds voorkomen in het SysV init script. Het eerste deel start kerneld zelf. Het tweede deel roept depmod -a tijdens de systeemstart aan om een lijst met alle beschikbare modules samen te stellen en het analyseert de onderlinge afhankelijkheden. De depmod map vertelt kerneld dan of voor een module een andere module moet zijn geladen voordat het zichzelf laadt.

Recente versies van kerneld hebben een optie om te worden gelinkt met de GNU gdbm library, libgdbm. Als je dit activeert bij het bouwen van de module utility's, dan *zal kerneld niet starten als libgdm niet beschikbaar is* wat het geval kan zijn als `/usr` zich op een aparte partitie bevindt en het starten van kerneld plaatsvindt voordat `/usr` is gemount. De aanbevolen oplossing is om `/usr/lib/libgdbm` te verplaatsen naar `/lib`, of om kerneld statisch te linken.

Pak vervolgens de kernelsources uit, configureer en bouw naar wens een kernel. Als je dit nog nooit eerder hebt gedaan, dan moet je beslist het README bestand lezen in de hoofddirectory van de Linux sources. Wanneer je make xconfig uitvoert om de kernel te configureren, let dan op de volgende vragen:

```
Enable loadable module support (CONFIG_MODULES) [Y/n/?] Y
```

Je moet de laadbare module ondersteuning selecteren, anders zullen er geen te laden modules zijn voor kerneld! Geef gewoon Y als antwoord.

```
Kernel daemon support (CONFIG_KERNELD) [Y/n/?] Y
```

Dit is natuurlijk ook nodig. Dan is er heel veel wat als module kan worden gebouwd, je zult vragen te zien krijgen als:

```
Normal floppy disk support (CONFIG_BLK_DEV_FD) [M/n/y/?]
```

waar je kunt antwoorden met een M voor "Module". Gewoonlijk hoeven alleen de drivers die nodig zijn op je systeem om op te starten in de kernel te worden gebouwd; de rest kan worden gebouwd als modules.

Essentiële drivers

Essentiële drivers die noodzakelijk zijn om je systeem te booten moeten in de core kernel worden gecompileerd en kunnen niet als modules worden geladen. Kenmerkend zijn hier de harddisk-driver en de driver voor het root filesysteem. Als je een dual-boot machine hebt en het rekent op bestanden die te vinden zijn op de andere partitie, dan moet je in de core kernel ook ondersteuning compileren voor dat bestandssysteem.

Wanneer je make config hebt doorlopen, compileer en installeer je de nieuwe kernel en de modules met make dep clean bzlilo modules modules_install.

Oef.

Compileren van een kernel-image

De opdracht **make zImage** zal stoppen zonder een kernel te installeren en de nieuwe kernel-image achterlaten in het bestand `arch/i386/boot/zImage`. Om dit image te gaan gebruiken, zul je het naar waar je je boot-image hebt, moeten kopiëren en het handmatig met LILO moeten installeren.

Zie voor meer informatie over het configureren, bouwen en installeren van je eigen kernel de Kernel-HOWTO, welke regelmatig wordt gepost naar `comp.os.linux.answers`, en beschikbaar is vanaf het Linux Documentatie Project (http://www.linuxdoc.org) en mirrors.

3.1. Kerneld uitproberen

Start je systeem nu opnieuw op met de nieuwe kernel. Wanneer het systeem weer opkomt, kun je ps ax uitvoeren, en als het goed is, zie je dan een regel voor kerneld:

```
  PID TTY STAT  TIME COMMAND
   59  ?  S     0:01 /sbin/kerneld
```

Een van de fraaie dingen met kerneld is dat zodra je de kernel en de daemon hebt geïnstalleerd, er weinig instellingen nodig zijn. Probeer om te beginnen één van de drivers die je als een module compileerde; het is eerder waarschijnlijk dan niet dat het zonder verdere configuratie zal werken. Als ik de floppy driver als een module compileerde, dan zou ik een DOS diskette in het diskettestation kunnen doen en typen:

```
osiris:~ $ mdir a:
 Volume in drive A has no label
 Volume Serial Number is 2E2B-1102
 Directory for A:/

binuti~1 gz       1942 02-14-1996  11:35a binutils-2.6.0.6-2.6.0.7.diff.gz
libc-5~1 gz      24747 02-14-1996  11:35a libc-5.3.4-5.3.5.diff.gz
        2 file(s)        26689 bytes
```

De floppy driver werkt! Het wordt automatisch door kerneld geladen wanneer ik de floppy disk probeer te gebruiken.

Om te bekijken of de floppy module inderdaad is geladen, kun je /sbin/lsmod uitvoeren om een opsomming van alle thans geladen modules te zien te krijgen:

```
osiris:~ $ /sbin/lsmod
Module:        #pages:  Used by:
floppy            11    0 (autoclean)
```

De "(autoclean)" betekent dat de module automatisch door kerneld zal worden verwijderd als het meer dan een minuut niet wordt gebruikt. Dus de 11 pagina's geheugen (= 44kB, één pagina is 4 kB) zullen alleen worden gebruikt wanneer ik het diskettestation benader. Gebruik ik de diskette meer dan een minuut niet, dan wordt het geheugen vrijgegeven. Heel fraai, als je weinig geheugen hebt voor je applicaties!

4. Hoe weet kerneld welke module te laden?

Alhoewel kerneld met ingebouwde kennis wordt geleverd over de meest gebruikelijke typen modules, zijn er situaties waar kerneld niet weet hoe een verzoek van de kernel af te handelen. Dit is het geval bij bv CD-ROM drivers of netwerkdrivers, waarbij er meer dan één mogelijke module kan worden geladen.

Het verzoek dat de daemon kerneld krijgt van de kernel is voor één van de volgende items:

- een block-device driver
- een character-device driver
- een binair formaat
- een lijndiscipline
- een bestandssysteem
- een netwerkdevice
- een netwerkservice (b.v. rarp)
- een protocol (b.v. IPX)

Kerneld stelt vast welke module moet worden geladen door het configuratiebestand `/etc/conf.modules` [1] te scannen. In dit bestand bevinden zich twee soorten regels: Directorypaden waar de module bestanden zijn te vinden, en aan de module toegekende aliassen die moet worden geladen voor een gegeven service. Als dit bestand nog niet bestaat, dan kun je het aanmaken door het uitvoeren van:

```
/sbin/modprobe -c | grep -v '^path' /etc/conf.modules
```

Als je nog een andere path directive toe wilt voegen aan de standaard directorypaden, dan moet je tevens alle standaard directorypaden opnemen, aangezien een path directive in `/etc/conf.modules` alle paden die modprobe standaard kent zullen worden vervangen door het toegevoegde pad.

Normaal gesproken zul je zelf geen directorypaden willen toevoegen, aangezien de ingebouwde set zou moeten voorzien in alle normale setups.

Als je aan de andere kant een alias of option directive toe wilt voegen, dan zullen je nieuwe regels in `/etc/conf.modules` worden toegevoegd aan die modprobe reeds kent. Als je een alias of optie herdefinieert dan zullen je nieuwe regels in `/etc/conf.modules` de ingebouwde regels overschrijven.

4.1. Block devices

Als je /sbin/modprobe -c uitvoert, dan zul je een overzicht krijgen met de modules die kerneld kent, en met welke verzoeken zij corresponderen. Het verzoek bijvoorbeeld dat er op neerkomt dat de floppy driver wordt geladen is voor het block-device met major nummer 2:

```
osiris:~ $ /sbin/modprobe -c | grep floppy
alias block-major-2 floppy
```

Waarom `block-major-2`? Omdat de floppy devices `/dev/fd*` gebruik maken van major device 2 en het block devices zijn:

```
osiris:~ $ ls -l /dev/fd0 /dev/fd1
brw-rw-rw-   1 root     root       2,   0 Mar  3  1995 /dev/fd0
brw-r--r--   1 root     root       2,   1 Mar  3  1995 /dev/fd1
```

4.2. Character devices

Met Character devices wordt op vergelijkbare wijze omgegaan. B.v. de ftape floppy tape driver neemt de plaats in van major-device 27:

```
osiris:~ $ ls -lL /dev/ftape
crw-rw----   1 root     disk      27,   0 Jul 18  1994 /dev/ftape
```

Kerneld is standaard echter niet op de hoogte van de ftape driver, het wordt niet weergegeven in de uitvoer van /sbin/modprobe -c. Voor het instellen van kerneld dat het de ftape driver laadt, moet ik een regel toevoegen aan het configuratiebestand `/etc/conf.modules`:

```
alias char-major-27 ftape
```

4.3. Netwerkdevices

Je kunt ook de naam van het device gebruiken in plaats van de `char-major-xxx` of `block-major-yyy` setup. Dit is vooral handig bij netwerkdrivers. Een driver voor een ne2000 netwerkkaart bijvoorbeeld, fungerend als `eth0` zou worden geladen met

```
alias eth0 ne
```

Als je een aantal opties aan de driver door moet geven, bijvoorbeeld om de module te laten weten welk IRQ de netwerkkaart gebruikt, dan moet je een “options” regel toevoegen:

```
options ne irq=5
```

Dit zorgt ervoor dat kerneld de NE2000 driver laadt met de opdracht

```
/sbin/modprobe ne irq=5
```

Natuurlijk zijn de werkelijke beschikbare opties specifiek voor de module die je laadt.

4.4. Binaire formaten

Binaire formaten worden op vergelijkbare wijze afgehandeld. Wanneer je een programma probeert uit te voeren waarvan de kernel niet weet hoe het te laden, dan krijgt kerneld een verzoek voor `binfmt-xxx`, waar `xxx` een nummer is dat is bepaald uit de eerste paar bytes van het uitvoerbare bestand. Dus de configuratie van kerneld om de binfmt_aout module voor ZMAGIC (a.out) uitvoerbare bestanden te laden is:

```
alias binfmt-267 binfmt_aout
```

Het magic nummer voor ZMAGIC bestanden is 267 en als je in `/etc/magic` kijkt, dan zie je het nummer 0413; houdt in gedachten dat `/etc/magic` gebruik maakt van octale getallen terwijl kerneld gebruik maakt van decimale getallen en 413 in het octale stelsel gelijk is aan 267 in het decimale stelsel. Er zijn feitelijk drie iets andere varianten op a.out uitvoerbare bestanden (NMAGIC, QMAGIC en ZMAGIC), dus voor volledige support van de binfmt_aout module hebben we nodig

```
alias binfmt-264 binfmt_aout  # pure executable (NMAGIC)
alias binfmt-267 binfmt_aout  # demand-paged executable (ZMAGIC)
alias binfmt-204 binfmt_aout  # demand-paged executable (QMAGIC)
```

a.out, Java en iBCS binaire formaten worden automatisch zonder enige configuratie door kerneld herkend.

4.5. Lijndisciplines (slip, cslip en ppp)

Om lijndisciplines wordt verzocht met `tty-ldisc-x`, waarbij `x` gewoonlijk 1 is (voor SLIP) of 3 (voor PPP). Beiden worden automatisch herkend door kerneld.

Nu we het toch over ppp hebben, als je wilt dat kerneld de bsd_comp data compressie module voor ppp laadt, dan moet je de volgende twee regels aan `/etc/conf.modules` toevoegen:

```
alias tty-ldisc-3 bsd_comp
alias ppp0 bsd_comp
```

4.6. Netwerk protocol family's (IPX, AppleTalk, AX.25)

Tevens kunnen een aantal netwerkprotocollen worden geladen. De kernel vraagt kerneld om een protocol familie (b.v. IPX) met een verzoek om `net-pf-X` waarbij `X` een nummer is die de gewenste familie aangeeft. B.v. `net-pf-3` is AX.25, `net-pf-4` is IPX en `net-pf-5` is AppleTalk; Deze nummers worden vastgesteld door de AF_AX25, AF_IPX enz. definities in het linux bronbestand `include/linux/socket.h`. Dus voor het automatisch laden van de IPX module, heb je in `/etc/conf.modules` een regel nodig als:

```
alias net-pf-4 ipx
```

Zie Common Problems voor informatie over hoe je een aantal ergerlijke meldingen tijdens het booten kunt voorkomen die zijn gerelateerd aan ongedefiniëerde protocol family's.

4.7. Bestandssystemen

kerneld verzoeken om bestandssystemen bestaan simpelweg uit de naam van het type bestandssysteem. Een veelvoorkomend gebruik hiervan is het laden van de module isofs voor CD-ROM bestandssystemen, d.w.z. bestandssystemen van het type iso9660:

```
alias iso9660 isofs
```

5. Devices waarvoor speciale configuratie nodig is

Voor een aantal devices is extra configuratie nodig naast de normale aliasing van een device naar een module.

- Character devices op major nummer 10: De diverse devices
- SCSI devices
- Devices waarvoor speciale initialisatie nodig is

5.1. char-major-10 : Muizen, watchdogs en randomness

Hardware devices worden gewoonlijk geïdentificeerd via hun major device nummers, b.v. ftape is `char-major-27`. Als je echter de bestanden in `/dev` bekijkt met char major 10, dan zul je zien dat dit een boel verschillende devices zijn, waaronder

- Diverse soorten muizen (busmuizen, PS/2 muizen)
- Watchdog devices

- De kernel `random` device
- APM (Advanced Power Management) interface

Deze devices worden door verschillende modules bestuurd, niet slechts door één, en daarom wordt voor deze misc. devices in de kerneld configuratie gebruik gemaakt van het major nummer en het minor nummer:

```
alias char-major-10-1 psaux     # Voor PS/2 mouse
alias char-major-10-130 wdt     # Voor WDT watchdog
```

Je hebt kernelversie 1.3.82 of nieuwer nodig om hiervan gebruik te kunnen maken; eerdere versies geven het minor nummer niet door aan kerneld, waardoor het voor kerneld onmogelijk is om uit te zoeken welk misc. device het moet laden.

5.2. Het laden van SCSI drivers: De `scsi_hostadapter` entry

Drivers voor SCSI devices bestaan uit een driver voor de SCSI host adapter (b.v. een Adaptec 1542), en een driver voor het type SCSI device dat je gebruikt, b.v. een harddisk, een CD-ROM of een tape-drive. Deze kunnen allen worden geladen als modules. Als je echter bijvoorbeeld de CD-ROM drive wilt benaderen die is gekoppeld aan de Adaptec kaart, dan weten de kernel en kerneld alleen dat ze de `sr_mod` module moeten laden om SCSI CD-ROM's te ondersteunen; niet bekend is op welke SCSI-controller de CD-ROM is aangesloten en daarom niet welke module moet worden geladen voor ondersteuning van de SCSI-controller.

Om hier iets aan te doen, kun je een regel toevoegen aan `/etc/conf.modules` voor de SCSI drivermodule waarbij aan kerneld wordt opgegeven welke van de vele mogelijke SCSI controllermodules het moet laden:

```
alias scd0 sr_mod               # sr_mod voor SCSI CD-ROM's ...
alias scsi_hostadapter aha1542  # ... Adaptec driver nodig
```

Dit werkt alleen bij kernelversie 1.3.82 of later.

Dit werkt als je slechts één SCSI-controller hebt. Als je er meer dan één hebt, dan wordt het iets moeilijker.

Over het algemeen kun je kerneld geen driver laten laden voor een SCSI-hostadapter, als er reeds een driver voor een andere hostadapter is geïnstalleerd. Je moet of beide drivers in je kernel inbouwen (niet als modules) of de modules handmatig laden.

Er *is* een manier om kerneld meerdere SCSI-drivers te laten laden. James Tsiao kwam met het volgende idee:

> Je kunt kerneld makkelijk de tweede scsi driver laten laden door met de hand de afhankelijkheid in te stellen in modules.dep. Hiervoor is de volgende regel nodig:
>
> ```
> /lib/modules/2.0.30/scsi/st.o: /lib/modules/2.0.30/scsi/aha1542.o
> ```
>
> Om kerneld `aha1542.o` te laten laden voordat het `st.o` laadt. Op mijn machine thuis is het vrijwel exact zo ingesteld als de setup hierboven, en het werkt prima bij al mijn tweede scsi devices, waaronder tape, cd-rom, en generic scsi devices. De keerzijde is dat **depmod -a** deze afhankelijkheden niet automatisch kan detecteren, dus de gebruiker moet ze met de hand toevoegen en geen **depmod -a** tijdens de systeemstart uitvoeren. Maar als het eenmaal is ingesteld, zal kerneld `aha1542.o` prima automatisch laden.

Wees je ervan bewust dat deze techniek alleen werkt als je verschillende soorten SCSI-devices op de twee controllers hebt, zoals bijvoorbeeld harddisks op de ene controller en cd-rom drives, tapes of generic SCSI-devices op een ander.

5.3. Wanneer het laden van een module niet genoeg is: De `post-install` entry

Soms is het slechts laden van de module niet genoeg om iets werkend te krijgen. Als je bijvoorbeeld je geluidskaart als een module hebt gecompileerd, is het vaak handig om een bepaald volumeniveau in te stellen. Het probleem is echter dat de instelling de volgende keer dat de module wordt geladen verdwijnt. Hier is een aardige truuk van Ben Galliart (<`bgallia@luc.edu`>):

> De uiteindelijke oplossing vereiste de installatie van het setmix package (ftp://sunsite.unc.edu/pub/Linux/apps/sound/mixers/) om daarna de volgende regel toe te voegen aan `/etc/conf.modules`:
>
> ```
> post-install sound /usr/local/bin/setmix -f /etc/volume.conf
> ```

Nadat de sound module is geladen, voert kerneld de opdracht uit die wordt aangegeven in de `post-install sound` entry. Dus de sound module wordt geconfigureerd met de opdracht /usr/local/bin/setmix -f /etc/volume.conf.

Dit kan ook voor andere modules handig zijn, de `lp` module bijvoorbeeld kan worden geconfigureerd met het programma `tunelp` door de toevoeging:

```
post-install lp tunelp options
```

Je hebt kerneld versie 1.3.69f of later nodig wil kerneld deze opties herkennen.

In een eerdere versie van deze mini-HOWTO werd een pre-remove optie genoemd dat kon worden gebruikt om een opdracht op te starten voordat kerneld een module zou verwijderen. Dit heeft echter nooit gewerkt en het gebruik ervan wordt daarom ontmoedigd. Naar alle waarschijnlijkheid zal deze optie in een toekomstige kerneld release verdwijnen. Het gehele onderwerp over module instellingen ondergaat op het moment een aantal wijzigingen en wellicht dat het er anders uitziet op je systeem wanneer je dit leest.

6. Kerneld bespioneren

Mocht je alles al hebben geprobeerd en kun je er niet achter komen wat de kernel vraagt kerneld te doen, dan is er een manier om de verzoeken die kerneld ontvangt te bekijken, om er zo achter te komen wat in `/etc/conf.modules` moet komen te staan: Het kdstat utility.

Dit handige kleine programmaatje wordt geleverd met het modules-package, maar is standaard niet gecompileerd of geïnstalleerd. Om het samen te stellen ga je naar de directory met kerneld bronnen en typ je make kdstat. Om dan kerneld informatie te laten weergeven wat het aan het doen is, voer je kdstat debug uit. Kerneld zal beginnen met het spuwen van wat meldingen op de console over wat het aan het doen is. Als je dan de opdracht probeert en uitvoert die je wilt gebruiken, dan zul je de kerneld verzoeken te zien krijgen; deze kunnen worden geplaatst in `/etc/conf.modules` en aliased naar de module die nodig is om de job te klaren.

Start /sbin/kdstat om het debuggen uit te zetten.

7. Speciale kerneld gebruiken

Ik wist dat je zou vragen naar het hoe voor het instellen van de screen-saver module!

In de `kerneld/GOODIES` directory in het modules package staan een paar kernelpatches voor screen-saver en console-beep ondersteuning in kerneld; deze maken nog geen onderdeel uit van de officiële kernel, dus je zult de kernel-patches moeten installeren en de kernel opnieuw moeten compileren.

Voor het installeren van een patch, gebruik je de opdracht patch:

```
cd /usr/src/linux
patch -s -p1 /usr/src/modules-*/kerneld/GOODIES/blanker_patch
```

Stel de kernel dan opnieuw samen en installeer deze.

Wanneer de schermbeveiliging wordt geactiveerd, zal kerneld de opdracht `/sbin/screenblanker` uitvoeren; dit bestand kan alles zijn wat je maar wilt, zoals bijvoorbeeld een shellscript die je favoriete schermbeveiliging uitvoert.

Wanneer de kernel het scherm weer in normale toestand terug wil brengen, stuurt het een SIGQUIT signaal naar het proces dat `/sbin/screenblanker` draait. Je shellscript of schermbeveiliger zou dit af moeten vangen. Denk eraan het scherm in de oorspronkelijke tekstmodus terug te brengen!

8. Algemene problemen en dingen die je je wellicht afvraagt

1. Waarom krijg ik Cannot locate module for net-pf-`X` meldingen wanneer ik /sbin/ifconfig uitvoer?

Zo om en nabij kernelversie 1.3.80 werd de netwerkcode gewijzigd dat het 't laden van protocol families als modules mogelijk maakte (b.v. IPX, AX.25 en AppleTalk). Dit zorgde voor de aanvulling van een nieuw kerneld verzoek: `net-pf-X`, waarbij `X` een nummer is die het protocol aanduidt (zie `/usr/src/linux/include/linux/socket.h` voor de betekenis van de diverse nummers). Helaas vangt ifconfig deze meldingen per ongeluk af, zodat bij een heleboel mensen een paar meldingen worden gelogd wanneer het systeem boot en het draait ifconfig om het loopback device in te stellen. De meldingen zijn onschuldig, en je kunt ze deactiveren door het toevoegen van de regels:

```
alias net-pf-3 off      # Vergeet AX.25
alias net-pf-4 off      # Vergeet IPX
alias net-pf-5 off      # Vergeet AppleTalk
```

aan `/etc/conf.modules`. Je voegt natuurlijk geen regel toe om IPX te deactiveren als je IPX wel als een module gebruikt.

2. Na het starten van kerneld, wordt mijn systeem zo langzaam als een slak wanneer ik mijn ppp-verbinding activeer

Er zijn hier diverse meldingen van. Het lijkt op een ongelukkige interactie tussen kerneld en het tkPPP script dat op een aantal systemen wordt gebruikt om de PPP verbinding op te zetten en te monitoren. Het script voert blijkbaar loops uit terwijl het ifconfig draait. Dit activeert kerneld, te zoeken naar de `net-pf-X` modules (zie hiervoor), de systeembelasting hoog houdend en mogelijk veel Cannot locat module for `net-pf-X` meldingen in de systeemlog veroorzakend. Er is geen bekende oplossing voor, anders dan geen gebruik te maken van tkPPP, of het zodanig te wijzigen dat het een andere manier om de verbinding te monitoren gebruikt.

3. kerneld laadt mijn SCSI-driver niet!

Voeg een regel toe voor de SCSI-hostadapter aan het bestand `/etc/conf.modules`. Zie de beschrijving van de `scsi_hostadapter` hiervoor.

4. modprobe klaagt over een ongedefinieerde gcc2_compiled

Dit is een bug in de module utility's, die alleen in binutils 2.6.0.9 en later aan het licht komt en het is ook gedocumenteerd in de release note van de binutils. Dus lees dat of haal een upgrade op voor de module-utilities die deze bug corrigeert.

5. Mijn geluidsdriver blijft zijn instellingen voor het volume enz vergeten

De instellingen voor een module worden opgeslagen in de module zelf wanneer het wordt geladen. Dus wanneer kerneld automatisch een module uit het geheugen verwijdert, zijn alle instellingen die je hebt ingesteld vergeten en de volgende keer dat de module weer wordt geladen keert het terug naar de standaardinstellingen.

Je kunt kerneld aangeven een module te configureren door een programma uit te voeren nadat de module automatisch is geladen. Zie Pre/Post Install on the `post-install` entry.

6. DOSEMU heeft een aantal modules nodig; hoe kan ik kerneld zover krijgen dat het deze laadt?

Dat kan niet. Geen van de dosemu versies, officiële of ontwikkelaarsversies ondersteunt het laden van de dosemu modules via kerneld. Met kernel 2.0.26 echter, heb je de speciale dosemumodules niet meer nodig; upgrade gewoon naar dosemu 0.66.1 of een nieuwere versie.

7. Waarom krijg ik Ouch, kerneld timed out, message failed meldingen ?

Wanneer de kernel een verzoek aan kerneld verstuurt, verwacht het hiervan binnen een seconde een bevestiging terug te ontvangen. Als kerneld deze bevestiging niet zendt, wordt deze melding gelogt. Het verzoek wordt opnieuw gestuurd en zou tenslotte door moeten komen.

Dit gebeurt gewoonlijk op systemen met een zeer hoge belasting. Aangezien kerneld een gebruikersproces is, wordt het net als elk ander proces op het systeem gepland. Ten tijden van hoge belasting, wordt het wellicht niet op tijd uitgevoerd om de bevestiging terug te sturen voordat de kernel onderbreekt.

Probeer het opnieuw opstarten van kerneld als dit bij een lage belasting gebeurt. Kill het kerneld proces, en start het weer op met de opdracht /usr/sbin/kerneld. Als het probleem blijft aanhouden dan zou je een bug report moeten mailen naar <`linux-kernel@vger.rutgers.edu`>, maar verzeker je er alsjeblieft van dat de door je in gebruik zijnde versies van de kernel, kerneld en module utility's up-to-date zijn voor je het als probleem post. Controleer de benodigdheden in `linux/Documentation/Changes`

8. Mount wacht niet op kerneld totdat het de module voor het bestandssysteem heeft geladen

Er zijn een aantal meldingen waardoor de opdracht mount(8) niet wacht op kerneld totdat het de module voor het bestandssysteem heeft geladen. lsmod toont wel dat kerneld de module laadt, en als je de opdracht mount onmiddellijk herhaalt, lukt het wel. Dit schijnt een bug te zijn in de module-utilities versie 1.3.69f waardoor een aantal Debian gebruikers wordt getroffen. Het kan worden gecorrigeerd door een latere versie van de module utility's op te halen.

9. kerneld kan de `ncpfs` module niet laden

Je moet de ncpfs utility's compileren met -DHAVE_KERNELD. Zie de ncpfs `Makefile`.

10. kerneld kan de `smbfs` module niet laden

Je hebt een oudere versie van de smbmount utility's in gebruik. Haal de laatste versie op (0.10 of later) vanaf het SMBFS archief op TSX-11 (ftp://tsx-11.mit.edu/pub/linux/filesystems/smbfs/)

11. Ik heb alles als modules gecompileerd, en nu boot mijn system niet of kerneld kan de module voor het root bestandssysteem niet laden!

Je kunt niet alles modularizeren: In de kernel moeten zich voldoende drivers bevinden om je root bestandssysteem te kunnen mounten, en de nodige programma's om kerneld te starten.[2]. Wat je niet kunt modularizeren:

- de driver voor de harddisk waar je root bestandssysteem op voorkomt
- de driver voor het root bestandssysteem zelf

- de binary format loader voor init, kerneld en andere programma's

12. kerneld laadt niet tijdens het booten; het geeft foutmeldingen over libgdbm

Nieuwere versies van kerneld hebben de GNU dmb library, `libgdbm.so` nodig om te kunnen draaien. De meeste installaties hebben dit bestand in `/usr/lib`, maar waarschijnlijk start je kerneld voordat het `/usr` bestandssysteem is gemount. Een symptoom hiervan is dat kerneld tijdens het starten van het systeem niet (vanuit je rc-scripts) zal starten, maar prima draait als je het met de hand start nadat het systeem is geboot. De oplossing is de kerneld opstart te verplaatsen nadat `/usr` is gemount of door de gdbm library naar je root bestandssysteem te verplaatsen, b.v. naar `/lib`.

13. Ik krijg Cannot load module `xxx` maar ik heb mijn kernel net opnieuw geconfigureerd zonder `xxx` ondersteuning!

De Slackware installatie (mogelijk anderen) bouwt een standaard `/etc/rc.d/rc.modules` die een expliciete modprobe uitvoert op een variëteit aan modules. Exact welke modules worden geladen met modprobe is afhankelijk van de oorspronkelijke configuratie van de kernel. Je hebt je kernel waarschijnlijk opnieuw geconfigureerd om er één of meer modules uit te laten die in rc.modules met modprobe worden getracht te laden, vandaar de foutmelding(en). Werk het rc.modules bestand bij door voor modules die je niet langer gebruikt een commentaarteken te plaatsen of verwijder `rc.modules` in zijn geheel en laat kerneld de modules laden wanneer ze nodig zijn.

14. Ik heb mijn kernel en modules opnieuw gecompileerd en krijg bij het booten steeds meldingen over unresolved symbols

Je hebt waarschijnlijk je kernel opnieuw geconfigureerd en gecompileerd en daar wat modules uitgelaten. Je hebt een aantal modules die je niet langer gebruikt in de `/lib/modules` directory. De eenvoudigste manier om dit te corrigeren is het verwijderen van de `/lib/modules/x.y.z` directory en de opdracht make modules_install vanuit de kernel broncode directory nogmaals op te starten. Dit probleem komt alleen voor wanneer je de kernel opnieuw configureert zonder van versie te wijzigen. Wanneer je deze foutmelding te zien krijgt wanneer je een nieuwe kernelversie hebt geïnstalleerd, dan gaat het om een ander probleem.

15. Ik installeerde Linux 2.1/2.3 en nu kan ik geen enkele module meer laden!

Oneven genummerde Linux versies zijn ontwikkelaarskernels. Als zodanig kan ervan worden verwacht dat er van tijd tot tijd iets breekt. Een van de dingen die veelbetekenend is gewijzigd is de wijze waarop modules worden afgehandeld, en waar de kernel en modules in het geheugen worden geladen.

Samengevat: als je modules wilt gebruiken in combinatie met een ontwikkelaarskernel, dan moet je

- het bestand `Documentation/Changes` lezen en hierin opzoeken welke packages moeten worden bijgewerkt op je systeem
- het laatste modutils package gebruiken, beschikbaar vanaf AlphaBits op Red Hat (ftp://ftp.redhat.com/pub/alphabits/) of de mirror site op TSX-11 (ftp://tsx-11.mit.edu/pub/linux/packages/alphabits/)

Ik raad op z'n minst het gebruik van kernel 2.1.29 aan, als je modules wilt gebruiken met een 2.1. kernel.

16. Hoe zit het met dial-on-demand netwerkverbinding?

kerneld bood oorspronkelijk enige ondersteuning voor het tot stand brengen van dialup netwerkverbindingen op verzoek; pakketjes naar een netwerk proberen te versturen zonder dat er een verbinding was zorgde ervoor dat kerneld het `/sbin/request_route` script opstartte om een PPP of SLIP-verbinding op te zetten.

Achteraf leek dit niet zo'n goed idee. Alan Cox van Linux networking fame schreef naar de linux-kernel mailinglist

> De request-route materie is verouderd, gebrekkig en onnodig [...] Het is bovendien verwijderd uit de 2.1.x structuren.

In plaats van het gebruik van het request-route script in combinatie met kerneld, kan ik je van harte het diald package (http://www.dna.lth.se/~erics/diald.html) van Eric Schenk aanbevelen om je demand dialing beheren.

Notes

1. Een aantal distributies noemen dit bestand `modules.conf`
2. In werkelijkheid is dit niet waar. Late 1.3.x en alle 2.x kernels ondersteunen het gebruik van een initiële ramdisk die door LILO of LOADLIN wordt geladen; het is mogelijk in een zeer vroeg stadium van het bootproces modules vanaf deze disk te laden. Hoe je dit doet is beschreven in het bestand `linux/Documentation/initrd.txt` dat wordt meegeleverd met de kernel bronbestanden.