Kerneld mini-HOWTO

Henrik Storner (storner@osiris.ping.dk) <mailto:storner@osiris.ping.dk> Diterjemahkan oleh: *Jimmy Yusandra* <mailto:jimmy@padang.wasantara.net.id> Version 1.7, last updated: July 19, 1997 (Terj. 16 Desember 1998)

Contents

1 Abstrak

Dokumen ini menjelaskan bagaimana cara untuk menggunakan fungsi kerneld di kernel Linux. Ini menerangkan :

- Apa itu kerneld
- Mengapa penulis ingin menggunakannya
- Dimana mendapatkan bagian-bagian yang dibutuhkan
- Bagaimana men-setup-nya
- Bagaimana memberitahu kerneld tentang *modules* yang tidak dikenalnya

- Bagaimana mengintip kerneld (bermanfaat dalam melakukan setup).
- Penggunaan khusus kerneld
- Masalah umum dan kejanggalan

Rilis terbaru dari dokumen ini dapat diperoleh di *kerneld-mini-HOWTO.html* `<http://eolicom.olicom.dk/~storner/kerneld-mini-HOWTO.html>`. Diantara rilis dari mini-HOWTO ini dapat diperoleh updates di daftar perubahan yang belum terstruktur di *kern.html* `<http://eolicom.olicom.dk/~storner/kern.html>`

Versi yang telah ditranslate dari dokumen ini dapat diperoleh di *ID-Kerneld-Mini-HOWTO* `<http://ldp.linux.or.id/HOWTO/mini/other-formats/html/ID-Kerneld-Mini-HOWTO.html>`.

2 Kredit

Jika Anda menemukan hal-hal yang salah didalam dokumen ini, silahkan kirimkan catatan tentang hal itu. Orang-orang berikut ini telah memberikan kontribusi atas mini-HOWTO ini dalam beberapa hal:

- 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>

Penghargaan yang sebesarnya atas perhatian dan anjuran yang telah dikirimkan oleh pembaca mini-HOWTO ini.

3 Apakah Kerneld itu?

kerneld adalah kelebihan yang diperkenalkan selama pengembangan kernel 1.3 oleh *Bjorn Ekwall* `<mailto:bj0rn@blox.se>`. Kerneld dimasukan ke dalam seluruh kernel 2.0 dan 2. Ini membuat "*modules*" - contohnya *device drivers, network drivers, filesystems* - untuk dapat di*load* / secara otomatis ketika dibutuhkan, daripada melakukannya secara manual dengan `modprobe` atau `insmod`.

Untuk aspek yang lebih mengagumkan, meskipun (belum ?) terintegrasi dengan kernel standar:

- Dapat di*setup* untuk menjalankan program pemakai daripada *screen blanker default*, sehingga memudahkan Anda menggunakan program apapun sebagai *screen-saver*.
- Seperti juga dukungan *screen-blanker*, Anda juga dapat mengubah "*beep*" konsol standar dengan sesuatu yang benar-benar berbeda . .

Kerneld terdiri dari dua bagian yang terpisah:

- dukungan dalam kernel linux untuk pengiriman permintaan kepada *daemon* yang dibutuhkan oleh *module* untuk tugas tertentu.
- *User-level* daemon yang dapat menentukan *modules* apa yang harus di*load* / untuk memenuhi permintaan dari kernel.

kedua bagian harus jalan agar kerneld dapat berfungsi - tidak cukup hanya satu atau yang lainnya yang di setup.

4 Untuk apa menggunakan kerneld?

Ada beberapa alasan yang bagus untuk menggunakan kerneld. Berikut ini yang akan penulis sebutkan adalah alasan penulis sendiri - yang lain mungkin ingin menggunakan alasan lainnya.

- Jika Anda harus membuat kernel untuk berbagai sistem yang hanya sedikit berbeda - jenis *network card* yang berbeda, sebagai contohnya - kemudian Anda bisa membangun kernel tunggal dan beberapa *modules*, daripada harus membangun satu kernel untuk setiap sistem.
- *modules* lebih mudah untuk di test oleh *developer* - tidak perlu *reboot* sistem untuk *load* / dan *unload driver*. (Ini berlaku untuk seluruh *modules*, tidak hanya untuk yang di*load* / kerneld.
- *Modules* mengurangi penggunaan memori oleh kernel, artinya Anda punya lebih banyak memori yang tersedia untuk aplikasi. Penggunaan memori oleh kernel *TIDAK PERNAH* di*swap*, sehingga jika digunakan memori 100Kb dari driver-driver yang tidak digunakan yang di*compile* kedalam kernel, ini merupakan pemborosan RAM.
- Beberapa peralatan yang penulis gunakan – misalnya *ftape floppy-tape driver*, atau iBCS – hanya tersedia sebagai *modules*, tapi penulis tidak ingin berurusan dengan proses *load* / dan *unload* ketika tidak membutuhkannya.
- Orang-orang membuat distribusi linux tidak perlu membangun 284 boot image yang berbeda - setiap pemakai me*load* / driver yang ia butuhkan hanya untuk hardwarenya. Ini digunakan, contohnya oleh *RedHat 4.0* didalam installasinya.

Tentu saja, juga ada alasan mengapa tidak ingin untuk menggunakannya - Anda mungkin lebih suka untuk hanya memiliki satu file *image* kernel dengan seluruh *driver* di dalamnya. Dalam kasus ini, Anda membaca dokumen yang salah.

5 Bagaimana cara memperoleh kerneld ?

Dukungan dalam linux kernel telah diperkenalkan dengan linux 1.3.57. Jika Anda memiliki kernel yang lebih lama, silahkan untuk meng*upgrade*nya jika ingin memperoleh dukungan atas kerneld. Seluruh ftp site linux utama memiliki *kernel sources* - penulis merekomendasikan agar meng*upgrade* ke rilis kernel yang stabil terbaru, 2.0 , sekarang berada pada patch level 29:

- *ftp://sunsite.unc.edu/pub/Linux/kernel/v2.0* `<ftp://sunsite.unc.edu/pub/Linux/kernel/v2.0>`

- *ftp://tsx-11.mit.edu/pub/linux/sources/system/v2.0* `<ftp://tsx-11.mit.edu/pub/linux/sources/system/v2.0>`
- *ftp://ftp.funet.fi/pub/Linux/PEOPLE/Linus/v2.0* `<ftp://ftp.funet.fi/pub/OS/Linux/PEOPLE/Linus/v2.0>`

User-level daemon dimasukkan dalam paket modules-1.2.8 dan dengan paket modules-2.0 yang lebih baru. Ini normalnya tersedia dari tempat yang sama dengan *kernel sources*, tapi lokasi resminya:

- *ftp://sunsite.unc.edu/pub/Linux/kernel/v2.0/modules-2.0.0.tar.gz* `<ftp://sunsite.unc.edu/pub/Linux/kernel/modules-2.0.0.tar.gz>`
- *ftp://tsx-11.mit.edu/pub/linux/sources/sbin/modules-2.0.0.tar.gz* `<ftp://tsx-11.mit.edu/pub/linux/sources/sbin/>`
- *ftp://ftp.funet.fi/pub/Linux/tools/modules-2.0.0.tar.gz* `<ftp://ftp.funet.fi/pub/OS/Linux/tools/>`

CATATAN: Jika ingin untuk mencoba memuat *module* dengan kernel **development** 2.1 Anda seharusnya menggunakan paket modutils- (BUKAN modules-) terbaru. Tapi 12.16 (lihat berikut ini) tentang masalah dengan *modules* dan kernel 2.1.

6 Bagaimana melakukan setup ?

Pertama sekali ambil bagian yang dibutuhkan : Kernel yang cocok dan *modules-utilities* terbaru. Kemudian install *modules-utilities*. Cukup sederhana - *unpack sources* dan jalankan `make install`. Ini meng-compile dan menginstall program berikut di `/sbin`: `genksysm`, `insmod`, `lsmod`, `modprobe`, `depmod`, `kerneld`. Penulis merekomendasikan untuk menambahkan beberapa baris ke *start-up scripts* untuk melakukan beberapa setup penting ketika mem*boot* linux. Tambahkan baris berikut ke file `/etc/rc.d/rc.S` (jika memakai slackware), atau ke `/etc/rc.d/rc.sysinit` (Kalau menjalankan SysVinit, misal: Debian, RedHat, Caldera) :

```
# Start kerneld - Seharusnya ini terjadi di bagian paling awal

# dari proses boot, tentu saja SEBELUM Anda menjalankan fsck di filesystems
# yang mungkin perlu untuk me-<em/load/  disk drivers secara otomatis
if [ -x /sbin/kerneld ]
then
        /sbin/kerneld
fi

# perintah fsck standar ada disini
# Jalankan perintah mount untuk me-mount root fs read-write

# Update kernel-module dependencies file
# root-fs HARUS di mount read write mulai dari sekarang
if [ -x /sbin/depmod ]
then
```

```
            /sbin/depmod -a
    fi
```

bagian pertama menjalankan kerneld itu sendiri.

Bagian kedua memanggil 'depmod -a' saat startup. Program depmod membuat daftar dari seluruh *modules* yang tersedia dan menganalisa *inter-dependencies*nya, sehingga tahu jika suatu *modules* memerlukan *module* lain untuk di*load* / sebelum ia kemudian me*load* / dirinya sendiri.

NOTE: Versi kerneld yang sekarang memiliki pilihan links dengan GNU dbm library, `libgdbm`. Jika Anda mengaktifkannya ketika membangun *module-utilities*, *Kerneld tidak akan dimulai jika* `libgdbm` *tidak tersedia* yang akan menjadi masalah jika Anda memiliki `/usr` di partisi yang terpisah dan memulai kerneld sebelum `/usr` di*mount*. Pemecahan yang dianjurkan adalah memindahkan `libgdbm` dari `/usr/lib` ke `/lib`, atau link kerneld secara *static*.

Selanjutnya, *unpack kernel source*, konfigurasikan dan *build* kernel ke tempat yang diinginkan. Jika Anda tidak pernah melakukan hal ini sebelumnya, Anda seharusnya membaca README file di top level dari Linux *sources*. Ketika menjalankan **make config** untuk megkonfigurasi kernel, perhatikan beberapa pertanyaan yang muncul:

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

Pilih *load* able *module* support/, atau tak akan ada *modules* yang akan di*load* oleh kerneld! Jawab saja *Yes*.

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

Yang ini, tentu saja, juga penting. Selanjutnya, banyak hal di kernel dan dapat dijadikan sebagai *modules* - Anda akan melihat pertanyaan seperti

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

Anda bisa menjawab dengan 'M' untuk 'Module'. Umumnya, hanya driver driver yang penting untuk *boot-up* sistem - seperti : driver hardisk, driver untuk root filsystem - yang seharus di masukkan kedalam kernel; sisanya dapat dijadikan sebagai modules.

Kemudian lakukan '`make config`', jalankan '`make dep`', '`make clean`', '`make zImage`' or '`make zlilo`', '`make modules`' and '`make modules_install`'.

Wuih..

Perintah '`make zImage`' meletakkan kernel image yang baru di file `arch/i386/boot/zImage`. Anda perlu meng*copy*nya ke lokasi *boot-image*, atau install di LILO.

Untuk informasi tentang mengkonfigurasi, membangun dan menginstall kernel, check *Kernel-HOWTO* `<http://sunsite.unc.edu/LDP/HOWTO/Kernel-HOWTO.html>` di kirim secara berkala ke `comp.os.linux.answers`, dan tersedia dari `sunsite.unc.edu` di direktori `/pub/Linux/docs/HOWTO`.

7 Test kerneld

Sekarang *reboot* dengan kernel yang baru. Setelah sistem hidup, jalankan '`ps -ax`', dan Anda seharusnya melihat sebuah baris untuk kerneld:

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

Salah satu hal yang menyenangkan dengan kerneld adalah sekali Anda memiliki kernel dengan *daemon* ter*install*, sedikit setup yang dibutuhkan. sebagai awalnya, coba gunakan salah satu driver yang Anda bangun sebagai modul - sepertinya lebih dari sekedar tidak bahwa ia bekerja tanpa dikonfigurasi lebih lanjut. penulis membuat driver floppy disk sebagai *module*, sehingga dapat meletakkan floppy DOS di drive dan :

```
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
```

driver floppy disk bekerja - ia di *load* secara otomatis oleh kerneld ketika penulis mencoba menggunakannya.

Untuk melihat apakah *module* floppy benar-benar di *load* , Anda jalankan /sbin/lsmod yang memperlihatkan seluruh *modules* yang baru saja di *load* :

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

Tulisan "*(autoclean)*" maksudnya adalah *module* akan secara otomatis dihapus oleh kerneld ketika tidak lagi digunakan setelah lebih dari satu menit. Sehingga 11 halaman memori (= 44 Kb, satu halaman adalah 4 Kb) hanya akan digunakan ketika ketika penulis mengakses floppy drive- jika penulis tidak menggunakannya setelah lebih dari satu menit, maka ia kan dihapus. Cukup bagus, jika Anda punya sedikit memori untuk aplikasi!

8 Bagaimana kerneld tahu *modules* apa yang akan di*load* ?

Meskipun kerneld memiliki kepandaian dalam hal mengetahui tipe- tipe yang paling umum *modules*, ada situasi dimana kerneld tidak akan tahu bagaimana menangani permintaan dari kernel. Kasus ini seperti pada driver Network atau CD-ROM, dimana ada lebih dari satu *module* yang mungkin untuk di*load* /.

Permintaan yang diterima oleh kerneld daemon dari kernel adalah untuk satu dari hal-hal berikut

- driver block-device
- driver character-device
- format binary
- tty line discipline

- filesystem
- network device
- network service (contoh: rarp)
- network protocol (contoh: IPX)

Kerneld menentukan *module* yang harus di*load* / dengan melakukan *scan* atas file konfigurasi `/etc/conf.modules`. Ada dua jenis entri dalam file ini: Paths (di mana *module-files* terletak), dan aliases (*modules* apa yang seharusnya di*load* /). Jika file ini belum ada, Anda bisa membuatnya dengan menjalankan

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

Jika ingin menambahkan "path" directive lainnya ke default paths, Anda juga *harus mengikutsertakan seluruh "default" paths*, karena path directive di /etc/conf.modules akan *mengganti* seluruh yang dikenal modprobe secara default!

Normalnya, Anda tidak ingin untuk menambahkan sendiri paths, karena *setting built-in* seharusnya melakukan semua setup "normal" (dan kemudian beberapa...), penulis janji!

Di sisi lain, jika hanya ingin untuk menambahkan alias atau sebuah option directive, entry baru di `/etc/conf.modules` akan *ditambahkan* ke yang telah dikenal oleh modprobe. Jika harus *mendefenisikan ulang* sebuah alias atau *option*, entri baru di `/etc/conf.modules` akan menggantikan yang *built-in*.

8.1 Block devices

Jika '`/sbin/modprobe -c`' dijalankan, maka akan diperoleh daftar *modules* yang dikenal oleh kerneld, dan untuk permintaan apa mereka dihubungkan. Sebagai contoh, permintaan yang berakhir dengan me*load* floppy driver adalah untuk *block-device* yang punya nomor mayor 2:

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

Mengapa *block-major-2* ? Karena floppy devices `/dev/fd*` menggunakan *major device* 2 dan merupakan *block devices*:

```
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
```

8.2 Character devices

Character devices dihubungkan dengan cara yang sama. Misal: driver ftape floppy tape yang berada di *major-device* 27:

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

walau bagaimanapun, kerneld tidak secara default mengenal driver ftape - ia tidak terdaftar di keluaran dari '`/sbin/modprobe -c`'.

Sehingga melakukan setup kerneld untuk me*load* / driver ftape, harus menambahkan satu baris ke fie konfigurasi kerneld `/etc/conf.modules`:

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

8.3 Network devices

Anda juga dapat menggunakan nama *device* di samping 'char-major-xxx' atau 'block-major-yyy'. Ini khususnya akan bermanfaat pada driver-driver network. contohnya: driver untuk *card* `ne2000` yang berfungsi sebagai eth0 akan di*load* / dengan :

```
alias eth0 ne
```

Jika Anda perlu untuk melewatkan beberapa *options* ke driver - contohnya: untuk memberitahukan *module* IRQ yang digunakan oleh *netcard*, tambahkan baris 'options':

```
options ne irq=5
```

Ini akan mengakibatkan kerneld me*load* driver NE2000 dengan perintah

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

Tentu saja, options sesunggunya yang tersedia sesuai dengan module yang di*load*.

8.4 Format Biner

Format Biner ditangani dengan cara yang serupa. Ketika mencoba untuk menjalankan sebuah program yang kerneld tidak tahu cara me*load*nya, kerneld menerima permintaan untuk `"binfmt-xxx"`, dimana xxx adalah nomor yang ditentukan dari beberapa byte awal dari *executable*. Sehingga, konfigurasi kerneld untuk mendukung proses *load* module/ `binfmt_aout` untuk ZMAGIC (a.out) *executable* adalah

```
alias binfmt-267 binfmt_aout
```

Karena *magic number* (lihat `/etc/magic`) untuk files ZMAGIC adalah 267. (Jika Anda check `/etc/magic`, Anda akan melihat nomor 0413, tapi `/etc/magic` menggunakan nomor dalam oktal sedangkan kerneld menggunakan desimal, dan octal 413 = desimal 267). Sebenarnya ada tiga perbedaan kecil antara varian-varian dari a.out *executables* (NMAGIC, QMAGIC and ZMAGIC), sehingga untuk dukungan penuh atas *module* `binfmt_aout` diperlukan

```
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)
```

Format-format biner a.out, Java and iBCS diketahui secara otomatis oleh kerneld, tanpa konfigurasi apapun.

8.5 Line disciplines (slip, cslip dan ppp)

Line disciplines diminta dengan `"tty-ldisc-x"`, dimana 'x' biasanya 1 (for SLIP) atau 3 (for PPP). Kedua hal ini dikenal oleh kerneld secara otomatis.

Bicara tentang ppp, Jika ingin kerneld untuk me*load module* kompresi data `bsd_comp` untuk ppp, maka harus ditambahkan dua baris berikut ke `/etc/conf.modules`:

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

8.6 Keluarga Network protocol (IPX, AppleTalk, AX.25)

Beberapa network protokol dapat di*load* juga sebagai modules. Kernel menanyakan kerneld untuk keluarga protokol(contoh: IPX) dengan permintaan untuk "net-pf-X" dimana X adalah sebuah nomor yang menyatakan keluarga mana yang diinginkan. Misal net-pf-3 adalah AX.25, net-pf-4 adalah IPX dan net-pf-5 adalah AppleTalk. (Nomor-nomor ini ditentukan oleh AF_AX25, AF_IPX etc. Defenisi lihat di source file include/linux/socket.h linux). Sehingga untuk me*load* secara otomatis *module* IPX, akan membutuhkan entry seperti berikut di /etc/conf.modules:

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

Lihat juga bagian berikut di 12 (masalah-masalah umum) untuk informasi tentang bagaimana menghindari pesan-pesan yang mengganggu waktu boot karena keluarga protokol yang tidak terdefinisikan.

8.7 File systems

Permintaan kerneld atas filesystems dilakukan dengan menggunakan nama dari tipe filesystem. Penggunaan umum dari hal ini adalah me*load module* isofs untuk filesystems CD-ROM, misal: tipe filesystem "iso9660":

```
alias iso9660 isofs
```

9 Device yang membutuhkan konfigurasi khusus

Beberapa devices perlu sedikit di konfigurasi diluar peng-aliasan normal device terhadap sebuah module.

- Character devices dengan major number 10: Devices lainnya
- SCSI devices
- Devices yang membutuhkan inisialisasi khusus

9.1 char-major-10 : Mouse, watchdogs dan randomness

Hardware devices biasanya di identifikasi melalui nomor-nmomor mayor device, misal : ftape adalah char-major-27. Namun, Jika Anda melihat entry-entry di /dev untuk char major 10, Anda akan melihat bahwa ini adalah kumpulan dari devices yang sangat berbeda, termasuk

- Mouse untuk berbagai jenis (mouse bus, mouse PS/2)
- Watchdog devices
- kernel 'random' device
- antarmuka APM (Advanced Power Management)

Jelaslah, devices tersebut di kontrol oleh beberapa *modules* yang berbeda, bukan yang tunggal. Sehingga, konfigurasi kerneld untuk **misc.devices** ini menggunakan major number **dan** minor number:

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

Perlu kernel versi 1.3.82 atau lebih baru untuk menggunakannya; versi yang lebih lama tidak melewatkan nomor minor ke kerneld, menyebabkan mustahil bagi kerneld untuk mengetahui misc.devices mana yang akan di *load* .

9.2 Me*load* driver SCSI : entry scsi_hostadapter

Drivers untuk SCSI devices terdiri dari sebuah driver untuk SCSI host adapter (contohnya: Adaptec 1542), dan driver untuk tipe SCSI device yang digunakan,contohnya: hard disk, CD-ROM atau a tape-drive. seluruhnya dapat di*load* sebagai modules. Namun, Ketika Anda ingin mengakses contohnya: CD-ROM drive yang terhubung ke card Adaptec, kernel dan kerneld hanya tahu bahwa perlu me*load module* sr_mod untuk mendukung SCSI CD_ROM - tapi tidak tahu di SCSI controller mana CD-ROM terhubung, dan juga tidak tahu *module* yang di*load* untuk SCSI controller.

Untuk menanggulangi ini, bisa ditambahkan sebuah entry untuk *module* SCSI driver di `/etc/conf.modules` yang memberitahu kerneld *module* mana yang harus di*load* dari sekian banyak *modules* SCSI controller:

```
alias scd0 sr_mod               # sr_mod for SCSI CD-ROM's ...
alias scsi_hostadapter aha1542  # ... need the Adaptec driver
```

Hanya bekerja dengan kernel versi 1.3.82 atau lebih baru.

Bekerja jika hanya ada satu SCSI controller. Jika Anda punya lebih dari satu, maka akan menjadi sedikit lebih sulit.

Secara umum, Anda tidak dapat membuat kerneld untuk me*load* driver untuk SCSI host adapter, jika driver untuk host adapter lainnya telah di install - Anda harus membuat kedua driver tersebut kedalam kernel (bukan sebagai *modules*), atau *load modules* secara manual.

Ini*adalah* cara yang dapat digunakan untuk membuat kerneld bisa me*load* multiple SCSI drivers. James Tsiao datang dengan ide ini :

Anda bisa dengan mudah membuat kerneld me *load* driver SCSI kedua dengan melakukan setup sendiri pada dependency di dalam modules.dep. Hanya diperlukan entry sebagai berikut:

/lib/modules/2.0.30/scsi/st.o: /lib/modules/2.0.30/scsi/aha1542.o

untuk membuat kerneld me*load* aha1542.o sebelum me*load* st.o. Mesin penulis dirumah di setup hampir sama persis dengan yang diatas, dan bekerja dengan baik untuk semua devices scsi secondary, termasuk tape,cd-rom, dan generic scsi devices. Kerugiannya adalah 'depmod -a' tidak dapat memdeteksi secara otomatis dependencies ini, sehingga user perlu untuk menambahkannya sendiri, dan tidak menjalankan 'depmod -a' saat boot up. Tapi sekali disetup, kerneld akan me*load* secara otomatis aha1542.o.

Perhatikan, bahwa teknik ini hanya bekerja jika Anda punya devices scsi yang berbeda di kedua kontroler - misal: harddisks di satu kontroler, dan drive CD-ROM, tape atau devices generic scsi di yang lainnya.

9.3 Ketika Me*load modules* tidak mencukupi: entry post install

Kadang-kadang, dengan hanya me*load module* tidak cukup untuk membuat sesuatu bekerja. Sebagai contoh, jika Anda punya soundcard yang di-compile sebagai module, maka sering lebih baik dengan mengaturnya di level volume tertentu. Hanya masalahnya adalah, setting hilang ketika *modules* di*load* untuk selanjutnya. Ini adalah trik yang indah dari Ben Galliart (bgallia@luc.edu):

```
Solusi terakhir membutuhkan paket setmix-0.1
( ftp://sunsite.unc.edu/pub/Linux/apps/sound/mixers/setmix-0.1.tar.gz )

Dan kemudian tambahkan baris berikut kedalam /etc/conf.modules :
    post-install sound /usr/local/bin/setmix -f /etc/volume.conf
```

Yang dilakukan oleh hal ini adalah setelah *module* sound di *load* , kerneld menjalankan perintah yang diberitahukan oleh entry 'post-install sound'. Sehingga *module* sound terkonfigurasi dengan perintah : '/usr/local/bin/setmix -f /etc/volume.conf'.

Mungkin ini bermanfaat juga untuk *modules* lainnya, misal: *module* lp dapat di konfigurasi memakai program `tunelp` dengan menambahkan

```
        post-install lp tunelp <options>
```

Agar kernel mengetahui pilihan-pilihan ini, dibutuhkan kerneld versi 1.3.69f atau yang lebih baru.

CATATAN: Versi awal dari mini howto ini mengingatkan option "pre-remove", mungkin ini bisa digunakan untuk menjalankan perintah sebelum kerneld menghapus module. Namun, hal ini tidak pernah bekerja dan penggunaannya kurang bermanfaat - sepertinya option ini akan menghilang di rilis kerneld yang akan datang. Isu keseluruhan dari *module* "settings" saat itu adalah sedang dalam perbaikan dan mungkin terlihat berbeda di dalam system Anda pada saat membaca ini.

10 Mengintip kerneld

Jika Anda telah mencoba segalanya, dan masih belum bisa memahami apa yang diperintahkan oleh kernel untuk dikerjakan oleh kerneld, ada cara untuk melihat permintaan yang diterima oleh kerneld, dan untuk mengetahui apa yang seharusnya pergi ke /etc/conf.modules : kdstat utility.

Program kecil ini ada didalam modules-package, tapi tidak di-compile secara default. Untuk membuatnya:

```
cd /usr/src/modules-2.0.0/kerneld
make kdstat
```

Kemudian, untuk membuat kerneld memperlihatkan informasi tentang apa yang sedang terjadi, jalankan

```
kdstat debug
```

dan kerneld akan memulai menampilkan pesan-pesan di console tentang apa yang sedang dilakukan. Jika Anda kemudian mencoba dan menjalankan perintah yang Anda ingin gunakan, maka akan terlihat permintaan-permintaan keneld; ini bisa di letakkan kedalam /etc/conf.modules dan dialiaskan ke module yang dibutuhkan untuk membuatnya bekerja.

Untuk mematikan debugging, jalankan '/sbin/kdstat nodebug' .

11 Penggunaan khusus kerneld

penulis telah tahu bahwa Anda akan menanyakan tentang bagaimana mensetup module screensaver ...

Direktori 'kerneld/GOODIES' di modules-package memiliki beberapa kernel patches untuk screensaver - dan consolebeep-support di kerneld; ini belum menjadi bagian resmi dari kernel. Sehingga perlu untuk menginstall kernel-patches dan membangun kernel.

Untuk memasang patch, gunakan perintah "patch" :

```
cd /usr/src/linux
patch -s -p1 </usr/src/modules-2.0.0/kerneld/GOODIES/blanker_patch
```

Kemudian rebuild dan install kernel baru

Ketika screensaver terpicu, kerneld akan menjalankan perintah " /sbin/screenblanker" - mungkin ini sebuah shellscript yang menjalankan screensaver favoritmu.

Ketika kernel ingin untuk melakukan unblank layar, ia mengirim sinyal SIGQUIT untuk memproses /sbin/screenblanker yag sedang jalan. Shell script atau scrensaver harus bisa menjebak hal ini, dan berhenti. Jangan lupa untuk mengembalikan layar ke modus text sebelumnya!

12 Masalah umum dan hal-hal yang membuat Anda ingin tahu

12.1 Mengapa aku mendapatkan pesan "Cannot locate *module* for net-pf-X" ketika menjalankan ifconfig

Kira-kira pada versi kenel 1.3.80, kode networking telah diubah untuk untuk mengizinkan *load* keluarga protokol) misal : IPX, AX.25 and AppleTalk) sebagai modules. Ini menyebabkan penambahan permintaan kerneld yang baru: net-pf-X, dimana X adalah sebuah angka yang mengidentifikasikan protokol (lihat /usr/src/linux/include/linux/socket.h untuk maksud dari angka tersebut. Malangnya, ifconfig secara tidak sengaja memicu pesan ini, sehingga banyak orang memperoleh pesan di log saat boot dan menjalankan ifconfig untuk melakukan setup loopback. Pesan ini tidak berbahaya, dan dapat dihilangkan dengan menambahkan baris

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

ke /etc/conf.modules. Tentu saja, Jika Anda menggunakan IPX sebagai module, Anda tidak perlu melumpuhkan IPX.

12.2 Setelah memulai kerneld, system saya menjadi makin lambat ketika mengaktifkan ppp-connection

Ada beberapa laporan tentang ini. Ini sepertinya interaksi yang kurang menguntungkan antara kerneld dan **tkPPP script** yang digunakan dalam beberapa system untuk men-setup dan memonitor hubungan ppp - script melakukan loop ketika menjalankan ifconfig. Ini memicu kerneld, untuk mencari module net-pf-X (lihat diatas) , menyebabkan system *load* tinggi dan mungkin memunculkan banyak pesan "Cannot locate *module* for net-pf-X" ke dalam log system. Tidak ada pemecahan yang diketahui, selain dengan tidak menggunakan tkPPP, atau mengubahnya untuk melakukan cara lain dalam memonitor hubungan.

12.3 kerneld tidak me*load* driver SCSI-ku!

Tambahkan entry untuk SCSI hostadapter ke /etc/conf.modules. Lihat keterangan di entry 9.2 (scsi_hostadapter) diatas.

12.4 modprobe mengeluh 'gcc2_compiled' being undefined

Ini adalah bug di module-utilities, yang terlihat hanya dengan 2.6.0.9 dan terbaru, ini juga telah di dokumentasikan di releasenote untuk binutils. Bacalah. atau upgrade ke module-utilities yang mengatasi ini, contohnya: modules-2.0.0.

12.5 Driver sound terus melupakan setting untuk volume dll

Setting untuk *module* di simpan didalam *module* itu sendiri ketika di*load* . Sehingga ketika kerneld melakukan auto-un*load* sebuah module, semua setting yang ada dihapus, dan selanjutnya *module* me*load* kembali dengan setting default.

Anda bisa memberitahukan kerneld untuk mengkonfigurasikan *module* dengan menjalankann program stelah *module* di *autoload* . Lihat 9.3 (bagian di atas) dalam entry 'post-install'.

12.6 DOSEMU butuh beberapa *modules* - bagaimana aku bisa membuat kerneld me*load* nya?

Tidak bisa. Tidak satupun versi DOSEMU -official atau development - mendukung proses *load modules* DOSEMU lewat kerneld. Namun, jika Anda menjalankan kernel 2.0.26 atau terbaru, Anda tidak butuh *modules* DOSEMU khusus lagi - upgrade saja DOSEMU ke 0.66.1.

12.7 Mengapa aku mendapat pesan "Ouch, kerneld timed out, message failed"?

Ketika kernel mengirim permintaan ke kerneld, ia berharap untuk menerima pemberitahuan kembali dalam waktu satu detik. Jika kerneld tidak mengirim pemberitahuan ini pesan tidak di log. Permintaan di transmisikan kembali, dan harus dilakukan berulang kali.

Ini biasanya terjadi pada system dengan *load* sangat tinggi - karena kerneld adalah proses yang berada di user-mode, ia dijadwal seperti juga proses lainnya didalam system. Pada saat *load* tinggi, ia mungkin tidak dapat bekerja pada saat itu untuk mengirim pemberitahuan sebelum kernel time out.

Jika ini masih terjadi meskipun ketika *load* rendah, coba restart kerneld. (Kill kerneld proses dan restart dengan perintah `/usr/sbin/kerneld`. Jika masalahnya sama, kirim mail bug report ke *linux-kernel@vger.rutgers.edu* `<mailto:linux-kernel@vger.rutgers.edu>`, tapi **tolong** yakinkan versi kernel dan kerneld up-to-date sebelum mengirimkannya.

12.8 mount tidak menunggu kerneld untuk me*load module* filesystem

Ada beberapa laporan bahwa perintah mount(8) tidak menunggu kerneld untuk me*load module* filesystem. lsmod tidak memperlihatkan bahwa kerneld me*load* module tersebut. Ini adalah sebuah bug di module-utilities versi 1.3.69f yang mempengaruhi beberapa pemakai Debian - bisa di perbaiki dengan memperoleh versi terbaru dari module-utilities.

12.9 kerneld gagal me-*load module* ncpfs

Anda perlu untuk meng-compile ncpfs utilities -DHAVE_KERNELD. Lihat ncpfs Makefile.

12.10 kerneld gagal me*load module* smbfs

Anda menggunakan versi lama dari smbmount utilities. Dapatkan yang terbaru (0.10 atau terbaru) dari *ftp://tsx-11.mit.edu/pub/linux/filesystems/smbfs/* `<ftp://tsx-11.mit.edu/pub/linux/filesystems/smbfs/>`

12.11 Saya telah membangun semuanya sebagai modules, dan sekarang system saya tidak dapat boot

12.12 kerneld gagal me*load module* root filesystem

Anda tidak dapat menjadikan **semuanya** modules: Kernel harus punya cukup driver didalamnya untuk bisa me mount root filesystem, dan menjalankan program penting untuk memulai kerneld. sehingga Anda tidak dapat menjadikannya modules

- driver untuk hard disk dimana root filesystem berada
- driver root filesystem itu sendiri

- binary format *loader* untuk `init`, kerneld dan program lainnya

[Sesungguhnya, ini tidak benar. kernel 1.3.x dan semua 2.0.x mendukung penggunaan ramdisk yang di *load* oleh LILO ata LOADLIN, dan memungkinkan untuk me*load modules* dari "disk" ini di bagian paling awal dari proses boot. Cara melakukannya di jelaskan di file Documentation/initrd.txt yang ada di kernel source-files.]

12.13 kerneld tidak akan *load* saat boot time - mengeluhkan libgdbm

Versi yang lebih baru dari kerneld butuh GNU dbm library, libgdbm.so, untuk bekerja. sebagian besar instalasi memiliki file ini di /usr/lib, tapi Anda mungkin memulai kerneld sebelum /usr filsystem di mount. Salah satu gejala dari ini adalah kerneld tidak akan dimulai selama boot-up (dari rc-scripts), tapi jalan dengan baik jika menjalankan dengan mengetikkannya setelah system boot-up. Pemecahannya adalah pindahkan startup kerneld sesudah /usr di mount, atau pindahkan gdbm library ke root filesystem, misal ke /lib.

12.14 Saya memperoleh pesan "Cannot *load module* xxx" tapi baru saja mengkonfigurasi ulang kernel tanpa xxx support!

Instalasi Slackware (mungkin juga lainnya) membuat default /etc/rc.d/rc.modules yang melakukan modprobe khusus atas beberapa modules. Tepatnya *module* mana yang di modprob tergantung pada konfigurasi asli kernel. Anda mungkin mengkonfigurasi ulang kernel dengan tidak memasukan satu atau lebih madules yang di modprobe di rc.modules, sehingga, muncul pesan kesalahan. Update rc.modules dengan mengcomment semua *modules* yang tidak lagi digunakan, atau hapus rc.modules semuanya dan biarkan kerneld me*load modules* ketika dibutuhkan.

12.15 Aku membangun ulang kernel dan modules, dan mendapat pesan tentang unresolved symbols ketika booting

Mungkin perlu mengkonfigurasi dan membangun ulang kernel dengan tidak memasukan beberapa modules. Beberapa *modules* lama yang tidak lagi digunakan bisa diperoleh dengan bergantung pada direktori /lib/modules. Pemecahan termudah adalah menghapus /lib/modules/x.y.z dan melakukan 'make modules_install' dari direktori kernel source. Catat bahwa hal ini hanya bisa dilakukan ketika mengkonfigurasi ulang kernel pada versi yang sama. Jika Anda melihat kesalahan ini ketika pindah ke kernal yang versi lebih baru, itu adalah kasus yang berbeda.

12.16 Saya telah menginstall Linux 2.1 dan sekarang tidak bisa me*load module* apapun

Linux 2.1 adalah kernel development terbaru. sehingga, bisa diharapkan hal ini dipecahkan dari waktu kewaktu. Satu dari hal yang telah diubah secara nyata adalah cara *module* ditangani, dan di mana kernel dan *modules* di *load* ke memori. Richard Henderson sekarang sedang berusaha dalam pengembangan *module* kernel.

Sebagai permulaan, jika ingin menggunakan *modules* denngan kernel 2.1, maka Anda harus

- membaca file Documentation/Changes dan melihat paket apa yang perlu diupgrade

- Gunakan paket modutils terbaru, tersedia di *ftp://ftp.redhat.com/pub/alphabits/* `<ftp://ftp.redhat.com/pub/alphabits/>` atau mirror site di *ftp://tsx-11.mit.edu/pub/linux/packages/alphabits/* `<ftp://tsx-11.mit.edu/pub/linux/packages/alphabits/>`

penulis akan merekomendasikan penggunaan kernel terbaru 2.1.29, jika Anda ingin menggunakan *modules* dengan kernel 2.1

12.17 Bagaimana dengan *dial-on-demand networking*?

kerneld pada dasarnya mempunyai dukungan untuk mengadakan hubungan jaringan dial-up sesuai dengan permintaan; mencoba mengirim paket ke jaringan tanpa terhubung akan menyebabkan kerneld untuk menjalankan script `/sbin/request_route` untuk men*setup* PPP atau SLIP connection.

Ini akan menjadi ide yang buruk. Alan Cox, ahli linux networking menulis di linux-kernel mailing list, bahwa

```
The request-route stuff is obsolete, broken and not required [...]
Its also removed from 2.1.x trees.
```

Daripada menggunakan script request-route dan kerneld, penulis dengan sepenuh hati menganjurkan untuk menginstall paket *diald* Eric Schenk, tersedia dari http://www.dna.lth.se/ erics/diald.html

13 Pesan Hak Cipta