De Linux Bootdisk HOWTO

Tom Fawcett (fawcett@croftj.net) Vertaald door: Ellen Bokhorst (bokkie@nl.linux.org)
v3.6, januari 2000

In dit document wordt beschreven hoe je je eigen boot/root diskettes voor Linux kunt ontwerpen en bouwen. Deze disks zouden kunnen worden gebruikt als rescue-disks of om nieuwe systeemcomponenten te testen. Als je de Linux FAQ en gerelateerde documenten, zoals de Linux Installation HOWTO en de Linux Install Guide, nog niet hebt gelezen, zou je niet moeten gaan proberen om bootdiskettes te bouwen. Zie de Appendix A.1 (Voorgefabriceerde bootdisks) als je slechts een rescue-disk voor noodgevallen wilt.

Inhoudsopgave

1 Voorwoord **2**

1.1 Versie noten . . . 3
1.2 Nog te doen . . . 3
1.3 Feedback en credits. . . . 3
1.4 Distributie richtlijnen . . . 3

2 Introductie **4**

3 Bootdisks en het bootproces **4**

3.1 Het bootproces . . . 4
3.2 Disktypen . . . 6

4 Bouwen van een root bestandssysteem **6**

4.1 Overzicht . . . 7
4.2 Aanmaken van het bestandssysteem . . . 7
4.3 Het bestandssysteem vullen. . . . 9
4.3.1 /dev . . . 9
4.3.2 /etc . . . 10
4.3.3 /bin en /sbin . . . 12
4.3.4 /lib . . . 12
4.4 Voorzieningen voor PAM en NSS . . . 13
4.4.1 PAM (Pluggable Authentication Modules). . . . 13
4.4.2 NSS (Name Service Switch). . . . 14
4.5 Modules . . . 14
4.6 Een aantal laatste details . . . 15
4.7 Het samenpakken . . . 15

5 Kiezen van een kernel. **15**

6 Het bijelkaar plaatsen: Het aanmaken van de diskette(s). 16

6.1 De kernel met LILO transporteren. 16

6.2 Transporteren van de kernel zonder LILO. 18

6.3 Instellen van het ramdisk word. 18

6.4 Transporteren van het root-bestandssysteem 19

7 Probleemoplossing 19

8 Diverse onderwerpen 21

8.1 Terugbrengen van de grootte van het root-bestandssysteem 21

8.2 Niet-ramdisk root-bestandssystemen 22

8.3 Bouwen van een utility-disk 23

9 Hoe de pro's het doen 23

10 Lijst met veelgestelde vragen 24

A Bronnen en verwijzingen. 28

A.1 Voorgefabriceerde bootdisks. 28

A.2 Rescue packages. 29

A.3 Graham Chapman's shell scripts 29

A.4 LILO – de Linux loader. 29

A.5 Linux FAQ en HOWTO's 30

A.6 Ramdisk usage. 30

B LILO boot foutcodes. 30

C Voorbeeld rootdisk directory listings 31

D Voorbeeldlisting van een directory op een utility-disk: 37

1 Voorwoord

Noot: Het kan zijn dat dit document is verouderd. Als de datum op de titelpagina van meer dan zes maanden geleden is, controleer dan alsjeblieft de homepage van het Linux Documentatie Project op `<http://linuxdoc.org/HOWTO/Bootdisk-HOWTO.html>` om te zien of er een recentere versie bestaat.

Alhoewel dit document in tekstvorm leesbaar zou moeten zijn, ziet het er in Postscript (`.ps`) of HTML vanwege de gebruikte typografische notatie *veel* beter uit. We moedigen je aan één van deze formaten te gebruiken. De Info versie komt er zo beschadigd uit dat het op moment van schrijven niet bruikbaar is.

1.1 Versie noten

Graham Chapman (grahamc@zeta.org.au) schreef de oorspronkelijke Bootdisk-HOWTO en hij ondersteunde het tot aan versie 3.1. Tom Fawcett (fawcett@croftj.net) voegde heel wat materiaal toe voor kernel 2.0, en hij is sinds versie 3.2 de beheerder van het document. Veel van de oorspronkelijke inhoud geschreven door Chapman is behouden gebleven.

Dit document is bedoeld voor **Linux kernel 2.0 en later**. Als je een oudere kernel hebt (1.2.xx of daarvoor), raadpleeg dan alsjeblieft de vorige versies van de Bootdisk-HOWTO die worden gearchiveerd op *de homepage van Graham Chapman* <http://www.zeta.org.au/~grahamc/linux.html>.

Deze informatie is bedoeld voor Linux op het **Intel** platform. Veel van deze informatie kan toepasbaar zijn voor Linux op andere processoren, maar we hebben hiermee geen ervaring, of informatie uit de eerste hand. Als iemand ervaring heeft met bootdisks op andere platformen, neem dan alsjeblieft contact met ons op.

1.2 Nog te doen

Enige vrijwilligers?

1. Beschrijf (of link naar ander document dat beschrijft) hoe andere opstartbare disk-achtige, zoals CDROM's, ZIP-disks en LS110-disks aan te maken.
2. Beschrijf hoe om te gaan met de zeer grote libc.so shared libraries. De opties zijn voornamelijk om oudere, kleinere library's te krijgen of om te bezuinigen op bestaande library's.
3. Heranalyseer distributie-bootdisks. Upgrade de sectie "Hoe de Pro's het doen".
4. Verwijder sectie welke beschrijft hoe een bestaande distributie-bootdisk te upgraden. Dit geeft gewoonlijk meer problemen dat het waard is.
5. Herschrijf/stroomlijn de sectie Probleemoplossing.

1.3 Feedback en credits.

We verwelkomen iedere feedback, goed of slecht, over de inhoud van dit document. We hebben ons best gedaan je ervan te verzekeren dat de instructies en de informatie hierin accuraat en betrouwbaar zijn. Laat het ons alsjeblieft weten als je fouten vindt of er iets in ontbreekt.

We bedanken de vele mensen die ons hebben geholpen met correcties en suggesties. Hun bijdragen hebben het veel beter gemaakt dan wanneer we het ooit alleen zouden hebben gedaan.

Stuur opmerkingen, correcties en vragen naar de auteur via het email-adres hierboven. Ik vind het niet erg vragen trachten te beantwoorden, maar lees alsjeblieft eerst de sectie 7 (Probleemoplossing).

1.4 Distributie richtlijnen

Copyright © 1995,1996,1997,1998,1999,2000 door Tom Fawcett en Graham Chapman. Dit document mag onder de voorwaarden uiteengezet in de Licentie van het Linux Documentatie Project op <http://linuxdoc.org/copyright.html> worden gedistribueerd. Neem alsjeblieft contact op met de auteurs als het je niet lukt aan de licentie te komen.

Dit is vrije documentatie. Het wordt gedistribueerd in de hoop dat het van nut zal zijn, maar **zonder enige garantie**; zonder zelfs de impliciete garantie van **verkoopbaarheid** of **geschiktheid voor een bepaald doel.**.

2 Introductie

Linux bootdisks zijn in een aantal situaties van nut, zoals:

- Bij het testen van een nieuwe kernel.
- Herstellen van een storing aan de disk – alles van een verloren bootsector tot een crash van de diskkoppen.
- Het herstellen van een gedeactiveerd systeem. Een kleine vergissing als root kan je systeem onbruikbaar achterlaten, en het kan zijn dat je van een diskette moet booten om het te herstellen.
- Upgraden van kritieke systeembestanden, zoals `libc.so`.

Er zijn verscheidene manieren om bootdisks te verkrijgen:

- Gebruik er één van een distributie als Slackware. Hiervan zal je op z'n minst kunnen booten.
- Gebruik een rescuepackage om disks in te stellen die zijn ontworpen om als rescue-disks te worden gebruikt.
- Leer wat nodig is voor de werking van ieder type disk en bouw ze dan zelf.

Een aantal mensen kiest voor de laatste optie zodat ze het zelf kunnen doen. Op die manier, kunnen ze er zelf uitkomen wat ze moeten doen om het te herstellen als er iets misgaat. Plus dat het een geweldige manier is om te leren hoe een Linux-systeem werkt.

In dit document wordt er vanuit gegaan dat je bekend bent met wat van de concepten met betrekking tot Linux systeembeheer. Je zou bijvoorbeeld bekend moeten zijn met directory's, bestandssystemen en diskettes. Je zou moeten weten hoe je gebruik maakt van `mount` en `df`. Je zou moeten weten waar de bestanden `/etc/passwd` en `fstab` voor zijn en hoe ze er uitzien. Je zou moeten weten dat de meeste commando's in deze HOWTO als root moeten worden uitgevoerd.

Het geheel opnieuw construeren van je eigen bootdisk kan gecompliceerd zijn. Als je de Linux FAQ en gerelateerde documenten, zoals de Linux Installation HOWTO en de Linux Installation Guide nog niet hebt gelezen, zou je niet moeten proberen, boot-diskettes te bouwen. Als je slechts een werkende bootdisk voor noodgevallen nodig hebt, dan is het *veel* eenvoudiger een voorgefabriceerde disk te downloaden. Zie Appendix A.1 (Voorgefabriceerde bootdisks), verderop in dit document, voor waar deze te vinden zijn.

3 Bootdisks en het bootproces

Een bootdisk is eigenlijk een miniatuur, zelf-bevattend Linux-systeem op een diskette. Het moet veel dezelfde functies verrichten die door een volledig Linux-systeem worden verricht. Voordat men zijn eigen bootdisk gaat bouwen, zou men de basis van het Linux bootproces moeten begrijpen. We presenteren je hier de basis, wat voldoende is voor het begrijpen van de rest van het document. Veel details en alternatieve opties zijn achterwege gelaten.

3.1 Het bootproces

Alle PC-systemen starten het bootproces door het uitvoeren van code in ROM (in het bijzonder, de BIOS) om de sector vanaf sector 0, cylinder 0 van de bootdrive te laden. De bootdrive is meestal het eerste diskettestation (aangeduid als `A:` onder DOS en `/dev/fd0` onder Linux). De BIOS probeert deze sector dan uit te voeren. Op de meeste opstartbare disks, bevat sector 0, cylinder 0 zowel:

- code van een bootloader zoals LILO, waarmee de kernel wordt gelokaliseerd, het laadt en uitvoert om de boot juist te starten als
- de start van een besturingssysteem, zoals Linux.

Als een Linux-kernel raw naar een diskette is gekopieerd, zal de eerste sector van de disk de eerste sector van de Linux-kernel zelf zijn. De eerste sector zal het bootproces vervolgen door de rest van de kernel vanaf het bootdevice te laden.

Zodra de kernel volledig is geladen, neemt het een basisinitialisatie van devices door. Het probeert vervolgens vanaf een device een **root-bestandssysteem** te laden en mounten. Een root-bestandssysteem is gewoon een bestandssysteem dat als "/" is gemount. De kernel moet worden verteld waar het naar het root-bestandssysteem moet zoeken; als het daar geen laadbaar image kan vinden, stopt het.

In een aantal bootsituaties – vaak wanneer vanaf een diskette wordt geboot –, wordt het root-bestandssysteem in een **ramdisk** geladen, RAM welke door het systeem wordt benaderd alsof het een disk is. Er zijn twee redenen waarom het systeem naar ramdisk laadt. Ten eerste is RAM verscheidene malen sneller dan een diskette, dus de werking van het systeem is snel; en ten tweede kan de kernel een **gecomprimeerd bestandssysteem** vanaf de diskette laden en het in de ramdisk decomprimeren, waardoor het mogelijk is meer bestanden op de diskette te persen.

Zodra het root-bestandssysteem is geladen en gemount, zie je een melding als:

```
VFS: Mounted root (ext2 filesystem) readonly.
```

Op dit punt vindt het systeem het programma `init` op het root-bestandssysteem (in `/bin` of `/sbin`) en voert het uit. `init` leest zijn configuratiebestand `/etc/inittab`, zoekt naar een regel aangeduid met `sysinit`, en voert het genoemde script uit. Het `sysinit` script is meestal iets als `/etc/rc` of `/etc/init.d/boot`. Dit script bestaat uit een set shell-commando's waarmee basissysteemservices worden ingesteld, zoals:

- het op alle disks uitvoeren van `fsck`,
- het laden van de benodigde kernel-modules,
- het starten van swapping,
- het initialiseren van het netwerk.
- het mounten van disks vermeld in `fstab`.

Dit script roept vaak diverse andere scripts aan voor de modulaire initialisatie. Bijvoorbeeld: in de structuur van SysVinit bevat de directory `/etc/rc.d/` een complexe structuur aan subdirectory's waarvan de bestanden aangeven hoe de meeste systeemservices te activeren en af te sluiten. Op een bootdisk is het sysinit-script vaak erg eenvoudig.

Wanneer het sysinit-script klaar is, wordt de controle aan `init` teruggegeven, die dan overgaat op het *standaard runlevel*, aangegeven in `inittab` door middel van het sleutelwoord `initdefault`. In de regel met het runlevel wordt meestal een programma als `getty` gespecifieerd, dat verantwoordelijk is voor het afhandelen van communicaties via de console en tty's. Het is het `getty` programma dat de bekende "`login:`" prompt afdrukt. Het `getty` roept op zijn beurt het programma `login` aan voor de afhandeling van de login-validatie en voor het instellen van gebruikerssessies.

3.2 Disktypen

Na het bootproces te hebben geïnspecteerd, kunnen we nu diverse daarbij betrokken soorten disks definiëren. We classificeren disks in vier typen. Bij de bespreking in dit gehele document maken we, tenzij anders is aangegeven, gebruik van de term "disk" om naar diskettes te refereren, alhoewel het meeste net zo goed ook voor harddisks zou kunnen gelden.

boot

Een disk met een kernel welke kan worden geboot. De disk kan worden gebruikt om de kernel te booten, waarmee dan een root-bestandssysteem op een andere disk kan worden laden. De kernel op een bootdisk moet gewoonlijk worden verteld waar het zijn root-bestandssysteem kan vinden.

Vaak wordt met een bootdisk een root bestandssysteem vanaf een andere diskette geladen, maar het is mogelijk een bootdisk zo in te stellen dat het in plaats daarvan een root bestandssysteem van een harddisk laadt. Dit wordt in het algemeen gedaan bij het testen van een nieuwe kernel. (in feite zal "`make zdisk`" een dergelijke bootdisk automatisch vanuit de kernel-source aanmaken).

root

Een disk met een bestandssysteem met bestanden die nodig zijn om een Linux-systeem te draaien. Een dergelijke disk hoeft niet noodzakelijkerwijs een kernel of een bootloader te bevatten.

Een root-disk kan worden gebruikt om het systeem onafhankelijk van enige andere disks te draaien, zodra de kernel is geboot. Meestal wordt de root-disk automatisch naar een ramdisk gekopieerd. Hierdoor wordt de toegang tot de root-disk veel sneller en geeft het 't diskettestation vrij voor een utility-disk.

boot/root

Een disk met zowel de kernel als een root-bestandssysteem. Met andere woorden: hierop staat alles dat nodig is om een Linux-systeem zonder harddisk te booten en draaien. Het voordeel van dit type disk is dat het compact is – alles dat nodig is op een enkele disk. De van alles geleidelijk toenemende grootte betekent echter dat het, zelfs met compressie, in toenemende mate moeilijker wordt alles op een enkele diskette te passen.

utility

Een disk met een bestandssysteem, maar die niet is bedoeld om als een root-bestandssysteem te worden gemount. Het is een aanvullende gegevensdisk. Je zou dit type disk gebruiken om aanvullende utilities mee te vervoeren voor als je teveel hebt voor op je root-disk.

In het algemeen bedoelen we wanneer we het hebben over "het bouwen van een bootdisk" het aanmaken van zowel de boot (kernel) als de root (bestanden). Ze mogen zowel samen (een enkele boot/root disk) of apart (boot + root disks) voorkomen. De meest flexibele benadering voor rescue-diskettes is waarschijnlijk het gebruik van aparte boot- en rootdiskettes, en één of meer utility-diskettes voor datgene wat teveel is voor deze disks.

4 Bouwen van een root bestandssysteem

Het aanmaken van het root-bestandssysteem bestaat uit het selecteren van de bestanden die nodig zijn om het systeem te draaien. In deze sectie wordt beschreven hoe een *gecomprimeerd bestandssysteem* gebouwd kan worden. Een minder algemene optie bestaat uit het bouwen van een ongecomprimeerd bestandssysteem op een diskette dat direct als root wordt gemount; dit alternatief wordt in de sectie 8.2 (Non-ramdisk Root Filesystem) beschreven.

4.1 Overzicht

Op een root-bestandssysteem moet al datgene voorkomen wat nodig is om een volledig Linux-systeem te ondersteunen. Hiervoor moeten op de disk de minimum-vereisten voor een Linux-systeem worden opgenomen:

- De basis bestandssysteemstructuur,
- Minimum set directories: `/dev, /proc, /bin, /etc, /lib, /usr, /tmp`,
- Basisset utilities: `sh, ls, cp, mv`, enz.,
- Minimum set configuratiebestanden: `rc, inittab, fstab`, enz.,
- Devices: `/dev/hd*, /dev/tty*, /dev/fd0`, enz.,
- Runtime library om te voorzien in basisfuncties die door utilities worden gebruikt.

Uiteraard is ieder systeem slechts dan van nut als je er iets onder kunt draaien, en een root-diskette komt meestal alleen van pas als je iets kunt doen als:

- Het controleren van een bestandssysteem op een andere drive, zoals bijvoorbeeld het controleren van je root-bestandssysteem op je harddrive, je van een andere drive Linux moet kunnen booten, zoals je dat met een root-diskette systeem kunt doen. Vervolgens kun je `fsck` op je oorspronkelijke root-drive uitvoeren zolang het niet is gemount.
- Het terugzetten van alles of delen van je oorspronkelijke root-drive vanaf een backup door gebruik te maken van archief- en compressie-utilities zoals `cpio, tar, gzip` en `ftape`.

We zullen beschrijven hoe een *gecomprimeerd* bestandssysteem te bouwen, het wordt zo genoemd omdat het op disk is gecomprimeerd en bij het booten op een ramdisk wordt gedecomprimeerd. Met een gecomprimeerd bestandssysteem kunnen er veel bestanden (bij benadering zes megabytes) op een standaard 1440K diskette passen. Omdat het bestandssysteem veel groter is dan een diskette, kan het niet op de diskette worden gebouwd. We moeten het elders bouwen, het comprimeren, en dan naar de diskette kopiëren.

4.2 Aanmaken van het bestandssysteem

Om een dergelijk root-bestandssysteem te bouwen, heb je een reserve device nodig welke groot genoeg is om alle bestanden voor compressie te kunnen bevatten. Je zal een device nodig hebben dat capabel is om ongeveer vier megabytes te kunnen bevatten. Er zijn verscheidene mogelijkheden:

- Gebruik een **ramdisk** (DEVICE = `/dev/ram0`). In dit geval wordt geheugen gebruikt om een diskdrive te simuleren. De ramdisk moet groot genoeg zijn om een bestandssysteem van de van toepassing zijnde grootte bijeen te houden. Controleer als je gebruik maakt van LILO, het configuratiebestand (`/etc/lilo.conf`) op een regel als:

  ```
  RAMDISK_SIZE = nnn
  ```

  waarmee het maximum-RAM is vastgesteld dat door een ramdisk in beslag kan worden genomen. De standaardwaarde is 4096K, wat voldoende zou moeten zijn. Je zou waarschijnlijk niet moeten proberen een dergelijke ramdisk op een computer met minder dan 8MB aan Ram te gebruiken.

  Controleer voor de zekerheid of je een device hebt als `/dev/ram0`, `/dev/ram` of `/dev/ramdisk`. Als dit niet zo is, maak `/dev/ram0` dan aan met `mknod` (major nummer 1, minor 0).

- Als je een ongebruikte harddiskpartitie hebt die groot genoeg is (verscheidene megabytes), dan is dit een goede oplossing.
- Gebruik een **loopback device**, waarmee het is toegestaan een diskbestand als een device te laten functioneren. Met het gebruik van een loopback-device kun je een bestand van drie megabyte op je harddisk aanmaken en er het bestandssysteem op bouwen.

  Typ `man losetup` voor instructies over het gebruik van loopback devices. Als je `losetup` niet hebt, kun je het samen met compatibele versies van `mount` en `unmount` ophalen vanuit het `util-linux` package in de directory `<ftp://ftp.win.tue.nl/pub/linux/utils/util-linux/>`.

  Als je op je systeem geen loop-device `/dev/loop0`, `/dev/loop1`, etc.) hebt, zal je het aan moeten maken met "`mknod /dev/loop0 b 7 0`". Zodra je deze speciale `mount` en `umount` binaire bestanden hebt geïnstalleerd, maak je op een harddisk met voldoende capaciteit een tijdelijk bestand (bv, `/tmp/fsfile`). Je kunt gebruik maken van een commando als:

  ```
  dd if=/dev/zero of=/tmp/fsfile bs=1k count=nnn
  ```

  voor het aanmaken van een *nnn*-block bestand.

  Gebruik hieronder de bestandsnaam in plaats van DEVICE. Bij het aanroepen van een mount-commando moet je de optie "`-o loop`" opgeven om mount aan te geven dat het een loopback-device gebruikt. Bijvoorbeeld:

  ```
  mount -o loop -t ext2 /tmp/fsfile /mnt
  ```

  zal `/tmp/fsfile` (via een loopback device) op het mountpoint `/mnt` mounten. Een `df` zal dit bevestigen.

Nadat je voor één van deze opties hebt gekozen, bereid je het DEVICE voor met:

```
dd if=/dev/zero of=DEVICE bs=1k count=3000
```

Dit commando vult het device op met nullen. Deze stap is belangrijk omdat het bestandssysteem op het device later zal worden gecomprimeerd, dus alle ongebruikte delen zouden voor het behalen van een maximale compressie met nullen moeten worden opgevuld.

Maak vervolgens het bestandssysteem aan. De Linux-kernel herkent twee typen bestandssystemen voor root-disks om automatisch naar ramdisk te worden gekopieerd. Dit zijn minix en ext2, waarvan ext2 het voorkeursbestandssysteem is. Bij het gebruik van ext2, kan het zijn dat je het handig vindt, met de optie `-i` meer inodes dan de standaard op te geven; `-i 2000` wordt aanbevolen, zodat je geen tekort zal hebben aan inodes. Als alternatief kan je op veel inodes besparen door veel van de onnodige `/dev`-bestanden te verwijderen. `mke2fs` zal standaard 360 inodes op een 1.44Mb diskette aanmaken. Ik bemerk dat 120 inodes op mijn huidige root-diskette ruim voldoende is, maar als je alle devices in `/dev` opneemt, dan zal het de 360 makkelijk overschrijden. Het gebruik van een gecomprimeerd root-bestandssysteem staat een groter bestandssysteem toe, en vandaar standaard meer inodes, maar mogelijk moet je toch het aantal bestanden nog verminderen of het aantal inodes verhogen.

Dus het commando dat je gebruikt, zal er ongeveer zo uitzien:

```
mke2fs -m 0 -i 2000 DEVICE
```

(Als je een loopback-device gebruikt, zal het te gebruiken diskbestand moeten worden opgegeven in plaats van dit DEVICE, mogelijk vraagt `mke2fs` om bevestiging).

Het commando `mke2fs` zal de beschikbare ruimte automatisch detecteren en zichzelf dienovereenkomstig configureren. De `-m 0` parameter voorkomt dat het ruimte voor root reserveert, en zorgt daarom voor meer bruikbare ruimte op de disk.

Mount vervolgens het device:

```
mount -t ext2 DEVICE /mnt
```

(Je moet een mountpoint `/mnt` aanmaken als het nog niet bestaat). In de resterende secties wordt aangenomen dat alle doeldirectory's zich relatief ten opzichte van `/mnt` bevinden.

4.3 Het bestandssysteem vullen.

Hier is een redelijke minimumset aan directory's voor je root-bestandssysteem:

- `/dev` – Devices, vereist voor I/O
- `/proc` – Directory stub vereist voor het proc bestandssysteem
- `/etc` – Systeem configuratiebestanden
- `/sbin` – Kritieke systeembinary's
- `/bin` – Basis binary's aangemerkt als onderdeel van het systeem
- `/lib` – Shared library's voor run-time ondersteuning
- `/mnt` – Een mountpoint voor beheer op andere disks
- `/usr` – Aanvullende utilities en applicaties

(De hier gepresenteerde directory-structuur is alleen voor het gebruik van een root-diskette. Echte Linux systemen hebben een complexere en gedisciplineerde set gedragslijnen, genaamd de *Filesystem Hierarchy Standard*, voor het vaststellen waar de bestanden thuishoren).

Drie van deze directory's zullen leeg zijn op het root-bestandssysteem, dus die hoeven alleen maar met `mkdir` te worden aangemaakt. De `/proc` directory is in wezen een stub waaronder het proc bestandssysteem wordt geplaatst. De directory's `/mnt` en `/usr` zijn slechts mountpoints voor gebruik als het boot/root systeem éénmaal draait. Vandaar nogmaals, hoeven deze directory's slechts te worden aangemaakt.

De resterende vier directory's worden in de volgende secties beschreven.

4.3.1 /dev

Een `/dev` directory met voor alle devices een speciaal bestand om door het systeem te worden gebruikt is voor ieder Linux-systeem verplicht. De directory zelf is een normale directory en kan met `mkdir` op gebruikelijke wijze worden aangemaakt. De speciale bestanden voor de devices moeten echter op een speciale manier, met het commando `mknod`, worden aangemaakt.

Er is echter een kortere weg – het kopiëren van de inhoud van je bestaande `/dev`-directory, waarbij je de niet gewenste bestanden verwijdert. De enige vereiste hierbij is dat je de speciale bestanden voor de devices met de optie `-R` kopieert. Hiermee zal de directory worden gekopieerd, zonder dat zal worden getracht de inhoud van de bestanden te kopiëren. *Zorg ervoor dat je de hoofdletter R gebruikt.* Als je de kleine letter `-r` switch gebruikt, zal dit waarschijnlijk tot gevolg hebben dat je de gehele inhoud van alle harddisks kopieert – of in ieder geval zoveel als op een diskette past! Pas daarom op en maak gebruik van het commando:

```
cp -dpR /dev /mnt
```

uitgaande van een op `/mnt` gemounte diskette. De `dp` switches zorgen ervoor dat symbolische links als links worden gekopieerd in plaats van als doelbestand, en dat de oorspronkelijke kenmerken behouden blijven, dus dat de informatie over de eigenaar behouden blijft.

Als je het op de moeilijke manier wilt doen, gebruik je `ls -l` om de major en minor device-nummers voor de services, die je wilt, te tonen, en maak je ze met behulp van `mknod` aan.

Alhoewel de devices zijn gekopieerd, loont het de moeite na te kijken dat eventuele door jou benodigde devices op de rescue-diskette zijn geplaatst. `ftape` maakt bijvoorbeeld gebruik van tape-devices, dus je zal hiervan een kopie nodig hebben als je van plan bent je floppy tape-drive vanaf de bootdisk te benaderen.

Voor ieder speciaal bestand voor een device is een inode vereist, en inodes kunnen zo nu en dan een schaarse bron vormen, vooral op diskette bestandssystemen. Het heeft daarom zin alle niet benodigde speciale bestanden voor de devices uit de `/dev`-directory van de diskette te verwijderen. Veel devices op specifieke systemen zijn klaarblijkelijk overbodig. Je kunt bijvoorbeeld met een gerust hart alle device-bestanden die beginnen met `sd` verwijderen als je geen SCSI-disks hebt. Op vergelijkbare wijze kunnen alle device-bestanden beginnend met `cua` worden verwijderd, als je niet van plan bent je seriële poort te gaan gebruiken.

Wees er zeker van dat je in ieder geval de volgende bestanden in deze directory opneemt: `console, kmem, mem, null, ram, tty1`.

4.3.2 /etc

In deze directory moeten een aantal configuratiebestanden voorkomen. Op de meeste systemen kunnen deze in drie groepen worden onderverdeeld:

1. Ten alle tijden vereist, *b.v.* `rc, fstab, passwd`.
2. Mogelijk vereist, maar niemand is daar echt zeker van.
3. Rommel die er in is geslopen.

Niet essentiële bestanden kunnen worden geïndentificeerd met het commando:

```
ls -ltru
```

Hiermee worden bestanden in omgekeerde volgorde op de laatst benaderde datum weergegeven, dus als eventuele bestanden niet zijn benaderd, kunnen ze op een root-diskette achterwege worden gelaten.

Op mijn root-diskettes, heb ik het aantal configuratiebestanden onder de 15 weten te houden. Dit reduceert mijn werk tot 3 sets bestanden:

1. Degenen die ik voor een boot/root-systeem moet configureren:
   (a) `rc.d/*` – systeem opstartscripts en scripts voor bij het wijzigen van het runlevel
   (b) `fstab` – lijst met te mounten bestandssystemen
   (c) `inittab` – parameters voor het `init` proces, het eerst gestarte proces tijdens het booten.
2. Degenen die ik voor een boot/root systeem op zou kunnen knappen:
   (a) `passwd` – lijst met gebruikers, homedirectory's, enz.
   (b) `group` – gebruikersgroepen.
   (c) `shadow` – wachtwoorden van gebruikers. Mogelijk heb je deze niet.
   (d) `termcap` – de terminal capaciteiten database.

`passwd` en `shadow` zouden moeten worden geoptimaliseerd, als de beveiliging van belang is, om het kopiëren van gebruikerswachtwoorden van het systeem te voorkomen, en zodat wanneer je vanaf de diskette boot, ongewenste logins worden verworpen.

Zorg ervoor dat in `passwd` op z'n minst `root` voorkomt. Zorg ervoor dat de directory's en shells voorkomen, als je van plan bent andere gebruikers in te laten loggen.

`termcap`, de terminal database, is kenmerkend verscheidene honderden kilobytes. De versie die je op je boot/root-diskette gebruikt zou zo moeten worden geoptimaliseerd dat het slechts de te gebruiken terminal(s) bevat, wat gewoonlijk slechts de `linux-console` entry is.

3. De rest. Ze zijn op 't moment actief, dus laat ik ze met rust.

Daarbuiten hoefde ik slechts twee bestanden te configureren en de inhoud daarvan is verbazingwekkend weinig.

- In `rc` zou moeten staan:

```
#!/bin/sh
/bin/mount -av
/bin/hostname Kangaroo
```

  Zorg ervoor dat de directory's goed zijn. Het is niet echt nodig `hostname` uit te voeren – het ziet er gewoon beter uit als je hetwel doet.

- In `fstab` zou op z'n minst moeten staan:

```
/dev/ram0       /               ext2    defaults
/dev/fd0        /               ext2    defaults
/proc           /proc           proc    defaults
```

  Je kunt de regels vanuit de bestaande `fstab` kopiëren, maar je zou geen van je harddisk-partities automatisch moeten mounten; gebruik hiervoor het sleutelwoord `noauto`. Het kan zijn dat je harddisk beschadigd is of niet meer werkt als de bootdisk wordt gebruikt.

Je `inittab` zou zodanig moeten worden gewijzigd dat de `sysinit` regel `rc`, of welk basisbootscript ook zal worden gebruikt, uitvoert. Als je er tevens verzekerd van wilt zijn, dat gebruikers niet via seriële poorten in kunnen loggen, maak je commentaar van alle regels voor `getty` waarin een `ttys` of `ttyS` device aan het einde van de regel is opgenomen. Laat de `tty` poorten erin, zodat je op de console in kunt loggen.

Een minimaal `inittab` bestand ziet er ongeveer zo uit:

```
id:2:initdefault:
si::sysinit:/etc/rc
1:2345:respawn:/sbin/getty 9600 tty1
2:23:respawn:/sbin/getty 9600 tty2
```

Het bestand `inittab` definieert wat het systeem in diverse toestanden uit zal voeren, waaronder bij het opstarten, het overgaan naar multi-user mode, enz. Zorg ervoor de bestandsnamen in `inittab` zorgvuldig te controleren; als `init` het vermelde programma niet kan vinden, zal de bootdisk blijven hangen, en zal je wellicht zelfs geen foutmelding krijgen.

Een aantal programma's kunnen niet naar elders worden verplaatst, omdat andere programma's hun locaties hebben ingeprogrammeerd. Op mijn systeem is in `/etc/shutdown`, `/etc/reboot` hardgecodeerd. Als ik

`reboot` naar `/bin/reboot` verplaats, en het `shutdown`-commando aanroep, dan zal de uitvoering ervan mislukken omdat het 't bestand `reboot` niet kan vinden.

Kopieer voor de rest alle tekstbestanden in je `/etc` directory, plus alle uitvoerbare bestanden in je `/etc/` directory waarvan je niet zeker bent of je ze niet nodig hebt. Raadpleeg als leidraad de listing in de Sectie C (Voorbeeld rootdisk directory listings). Waarschijnlijk volstaat het alleen die bestanden te kopiëren, maar systemen verschillen nogal, dus je kan er niet zeker van zijn dat dezelfde set bestanden op je systeem equivalent is aan de bestanden in de lijst. De enige zekere methode is te beginnen met `inittab` en uit te werken wat nodig is.

De meeste systemen maken nu gebruik van een `/etc/rc.d/` directory met shell-scripts voor verschillende run-levels. Het minimum is een enkel `rc` script, maar mogelijk is het eenvoudiger gewoon `inittab` en de directory `/etc/rc.d` vanaf je bestaande systeem te kopiëren, en de directory `rc.d` te ontdoen van de shell-scripts omdat de verwerking niet relevant is voor een systeemomgeving voor op een diskette.

4.3.3 /bin en /sbin

De `/bin` directory is een prima plaats voor extra utilities die je nodig hebt voor het verrichten van basis-bewerkingen, utilities zoals `ls`, `mv`, `cat` en `dd`. Zie Appendix C (Voorbeeld rootdisk directory listings) voor een voorbeeldlijst met bestanden voor in de directory's `/bin` en `/sbin`. Hierin staan geen utilities die nodig zijn om bestanden vanuit een backup terug te plaatsen, zoals `cpio`, `tar` en `gzip`. Dat komt doordat ik deze op een aparte utility-diskette plaats, om ruimte op de boot/root-diskette te besparen. Zodra de boot/root-diskette is geboot, wordt het naar de ramdisk gekopieerd waarbij het diskettestation vrij blijft om een andere diskette, de utility-diskette, te mounten. Ik mount deze meestal als `/usr`.

De aanmaak van een utility diskette wordt hierna in de sectie 8.3 (Bouwen van een utility-disk) beschreven. Waarschijnlijk is het wenselijk een kopie van dezelfde versie backuputilities, die worden gebruikt om de backups te schrijven, te beheren, zodat je geen tijd verspilt bij het proberen te installeren van versies die je backuptapes niet in kunnen lezen.

Zorg er in ieder geval voor dat je de volgende programma's opneemt: `init`, `getty` of equivalent, `login`, `mount`, één of andere shell die je rc-scripts uit kan voeren, een link van `sh` naar de shell.

4.3.4 /lib

In `/lib` plaats je de benodigde shared library's en loaders. Als de benodigde library's niet in de directory `/lib` zijn te vinden, zal het systeem niet kunnen booten. Als je geluk hebt, zie je misschien een foutmelding die aangeeft waarom niet.

Bijna ieder programma heeft op z'n minst de library `libc` `libc.so.`*N* nodig, waar *N* het huidige versienummer is. Kijk in de directory `/lib`. `libc.so.N` is meestal een symlink naar een bestandsnaam met een volledig versienummer:

```
% ls -l /lib/libc*
-rwxr-xr-x   1 root     root      4016683 Apr 16 18:48 libc-2.1.1.so*
lrwxrwxrwx   1 root     root           13 Apr 10 12:25 libc.so.6 -> libc-2.1.1.so*
```

In dit geval gebruik je `libc-2.1.1.so`. Om andere library's op te zoeken ga je door alle binaire bestanden die je van plan bent op te nemen en controleer je daarvan de afhankelijkheden met het commando `ldd`. Bijvoorbeeld:

```
% ldd /sbin/mke2fs
libext2fs.so.2 => /lib/libext2fs.so.2 (0x40014000)
```

```
libcom_err.so.2 => /lib/libcom_err.so.2 (0x40026000)
libuuid.so.1 => /lib/libuuid.so.1 (0x40028000)
libc.so.6 => /lib/libc.so.6 (0x4002c000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Ieder bestand aan de rechterkant is vereist. Het bestand mag een symbolische link zijn.

Een aantal library's is *nogal groot* en zal niet zo eenvoudig op je root-bestandssysteem passen. De hierboven genoemde `libc.so` bijvoorbeeld, is zo ongeveer 4 meg. Je zal library's bij het kopiëren naar je root-bestandssysteem waarschijnlijk moeten strippen. Zie de sectie 8.1 (Terugbrengen van de grootte van het root-bestandssysteem) voor instructies.

In `/lib` moet je ook een loader voor de library's opnemen. De loader zal óf `ld.so` (voor a.out libraries) óf `ld-linux.so` (voor ELF libraries) zijn. Nieuwere versies van `ldd` geven je exact aan welke loader nodig is, zoals in het voorbeeld hiervoor, maar oudere versies doen dit wellicht niet. Als je niet zeker weet welke je nodig hebt, pas je het `file` commando toe op de library. Bijvoorbeeld:

```
% file/lib/libc.so.4.7.2 /lib/libc.so.5.4.33 /lib/libc-2.1.1.so
/lib/libc.so.4.7.2: Linux/i386 demand-paged executable (QMAGIC), stripped
/lib/libc.so.5.4.33: ELF 32-bit LSB shared object, Intel 80386, version 1, stripped
/lib/libc-2.1.1.so: ELF 32-bit LSB shared object, Intel 80386, version 1, not stripped
```

De `QMAGIC` geeft aan dat `4.7.2` voor a.out library's is, en `ELF` geeft aan dat `5.4.33` en `2.1.1` voor ELF zijn.

Kopieer de specifieke loader(s) die je nodig hebt naar het root bestandssysteem dat je aan het bouwen bent. Library's en loaders zouden *zorgvuldig* moeten worden gecontroleerd met de opgenomen binaire bestanden. Als de kernel een benodigde library niet kan laden, zal de kernel meestal zonder foutmelding afbreken.

4.4 Voorzieningen voor PAM en NSS

Mogelijk dat je systeem dynamisch geladen library's nodig heeft, die voor `ldd` niet zichtbaar zijn.

4.4.1 PAM (Pluggable Authentication Modules).

Als er op je systeem gebruik wordt gemaakt van PAM (Pluggable Authentication Modules), moet je er een voorziening voor treffen op je bootdisk, ander zal je niet in kunnen loggen. PAM, is in 't kort een geraffineerde modulaire methode voor de authenticatie van gebruikers en het beheren van de toegang aan gebruikers tot services. Een eenvoudige manier om vast te stellen of op je systeem gebruik wordt gemaakt van PAM is de directory `/etc` op je harddisk te controleren op een bestand met de naam `pam.conf` of op een directory `pam.d`; als één van beiden bestaat, moet je een voorziening treffen voor een minimale ondersteuning van PAM. (Als alternatief voer je `ldd` uit op het uitvoerbare bestand `login`; als in de uitvoer `libpam.so` voorkomt, heb je PAM nodig).

Gelukkig heb je op bootdisks vaak niets te maken met beveiliging aangezien iedereen die fysieke toegang tot een computer heeft, meestal sowieso alles ermee kan doen. Daarom kun je PAM effectief deactiveren door het aanmaken van een eenvoudig `/etc/pam.conf` bestand in je root-bestandssysteem dat er ongeveer zo uitziet:

```
OTHER   auth       optional     /lib/security/pam_permit.so
OTHER   account    optional     /lib/security/pam_permit.so
OTHER   password   optional     /lib/security/pam_permit.so
OTHER   session    optional     /lib/security/pam_permit.so
```

Kopieer ook het bestand `/lib/security/pam_permit.so` naar je root-bestandssysteem. Deze library is slechts zo'n 8K dus heeft het een minimale overhead tot gevolg.

Deze configuratie staat iedereen volledige toegang tot de bestanden en services op je machine toe. Als beveiliging op je bootdisk je om één of andere reden lief is, zal je wat van je of je volledige harddisk's PAM setup naar je root bestandssysteem moeten kopiëren. Lees in ieder geval zorgvuldig de documentatie van PAM door, en kopieer alle benodigde library's in `/lib/security` naar je root-bestandssysteem.

Je moet tevens `/lib/libpam.so` op je bootdisk plaatsen. Maar je weet dit reeds aangezien je ldd op `/bin/login` toepaste, waarmee deze afhankelijkheid werd getoond.

4.4.2 NSS (Name Service Switch).

Als je glibc (ala libc6) gebruikt, zal je voorzieningen moeten treffen voor name services anders zal je niet in kunnen loggen. Het bestand `/etc/nsswitch.conf` bestuurt voor diverse services de database lookups. Als je niet van plan bent services te benaderen vanaf het netwerk (bv DNS of NIS lookups), zal je een eenvoudig `nsswitch.conf` moeten prepareren dat er ongeveer zo uitziet:

```
passwd:     files
shadow:     files
group:      files
hosts:      files
services:   files
networks:   files
protocols:  files
rpc:        files
ethers:     files
netmasks:   files
bootparams: files
automount:  files
aliases:    files
netgroup:   files
publickey:  files
```

Hiermee wordt aangegeven dat iedere service alleen door lokale bestanden zal worden geleverd. Je zal ook `/lib/libnss_files.so.1` op moeten nemen, welke dynamisch zal worden geladen om de bestands lookups af te handelen.

Als je van plan bent het netwerk vanaf je bootdisk te benaderen, wil je wellicht een nauwgezetter `nsswitch.conf` bestand aanmaken. Zie de `nsswitch` manpage voor details. Houd in gedachten dat je voor iedere *service* die je specificeert een bestand `/lib/libnss_`*service*`.so.1` opneemt.

4.5 Modules

Als je een modulaire kernel hebt, zou je moeten overwegen welke modules je na het booten vanaf je bootdisk wilt laden. Wellicht dat je de `ftape` en `zftape` modules op wilt nemen, als je backup-tapes op floppy tape staan, modules voor SCSI-devices als je ze hebt, en mogelijk modules voor PPP of SLIP ondersteuning als je in een noodgeval een verbinding wilt kunnen maken met het net.

Deze modules mogen in `/lib/modules` worden geplaatst. Je zou ook `insmod`, `rmod` en `lsmod` op moeten nemen. Afhankelijk van of je modules automatisch wil laden, wil je wellicht ook `modprobe`, `depmod` en `swapout` insluiten. Als je `kerneld` gebruikt, doe deze dan samen met `/etc/conf.modules` op de diskette.

Het belangrijkste voordeel bij het gebruik van modules is echter dat je niet-kritieke modules op een utility-disk kunt plaatsen en dat je ze wanneer nodig laadt, waarbij je dus minder ruimte verbruikt op je root-disk. Deze benadering verdient de voorkeur boven het bouwen van één zeer grote kernel met veel ingebouwde drivers, als je veel verschilende devices hebt.

Om een gecomprimeerd ext2 bestandssysteem te booten, heb je ingebouwde ondersteuning nodig voor een ramdisk en ext2. Hier kan niet als modules in worden voorzien.

4.6 Een aantal laatste details

Een aantal systeemprogramma's, zoals `login`, klaagt als het bestand `/var/run/utmp` en de directory `/var/log` niet voorkomen. Dus:

```
mkdir -p /mnt/var/{log,run}
touch /mnt/var/run/utmp
```

Nadat je tenslotte alle benodigde library's hebt ingesteld, voer je `ldconfig` uit om `/etc/ld.so.cache` op het root-bestandssysteem opnieuw aan te maken. De cache vertelt de loader waar de library's te vinden zijn. Voor het opnieuw aanmaken van `ld.so.cache` voer je de volgende commando's uit:

```
chdir /mnt; chroot /mnt /sbin/ldconfig
```

De `chroot` is nodig omdat `ldconfig` altijd de cache voor het root-bestandssysteem opnieuw aanmaakt.

4.7 Het samenpakken

Zodra je klaar bent met het constructureren van een root-bestandssysteem, unmount je het, en kopieer je het naar een bestand en comprimeert dat:

```
umount /mnt
dd if=DEVICE bs=1k | gzip -v9 > rootfs.gz
```

Wanneer dit klaar is, zal je een bestand `rootfs.gz` hebben, wat je gecomprimeerde root-bestandssysteem is. Je zou de grootte ervan moeten controleren om er zeker van te zijn dat het op een diskette past; als dit niet zo is, zal je terug moeten gaan en wat bestanden moeten verwijderen. In Sectie 8.1 (Terugbrengen van de grootte van het root-bestandssysteem) staan een aantal hints voor het terugbrengen van de grootte van het root-bestandssysteem.

5 Kiezen van een kernel.

Op dit punt heb je een compleet gecomprimeerd root-bestandssysteem. De volgende stap bestaat uit het bouwen of uitkiezen van een kernel. In de meeste gevallen zal het mogelijk zijn om je huidige kernel te kopiëren en je diskette daar vanaf te booten. Er kunnen echter situaties zijn dat je een aparte kernel wilt bouwen.

Één reden is de grootte. Als je een enkele boot/root diskette bouwt, zal de kernel één van de grootste bestanden op de diskette zijn, dus zal je de grootte van de kernel zoveel mogelijk willen beperken. Bouw het om de grootte van de kernel te beperken met een minimum set aan faciliteiten die nodig zijn het gewenste systeem te ondersteunen. Dit betekent er alles uit te laten wat je niet nodig hebt. Ondersteuning voor

netwerken als ook voor eventuele diskdrives en andere devices die je niet nodig hebt voor het draaien van je boot/root-systeem zijn prima achterwege te laten. Zoals eerder uiteengezet, *moet* je kernel ingebouwde ondersteuning hebben voor ramdisk en ext2.

Je zult uit moeten werken wat er in terug te plaatsen als je een minimum set faciliteiten hebt uitgewerkt om in een kernel op te nemen. Waarschijnlijk het meest algemene gebruik voor een boot/root-diskette zou zijn een systeem te bestuderen en een beschadigd root-bestandssysteem te herstellen, en hiervoor heb je wellicht kernel-ondersteuning nodig. Als bijvoorbeeld je backups allen op tape worden bewaard door gebruik te maken van Ftape om je tapedrive te benaderen, dan zal het niet mogelijk zijn vanaf je backuptapes een herstelprocedure uit te voeren als je je huidige root-drive en drives met Ftape kwijtraakt. Je zult Linux opnieuw moeten installeren, ftape moeten downloaden en opnieuw moeten installleren om vervolgens je backups opnieuw in proberen te lezen.

Het punt hier is dat, dat welke I/O ondersteuning je ook aan je kernel hebt toegevoegd om backups te ondersteunen dit ook in je boot/root kernel toegevoegd zou moeten zijn.

De procedure voor het werkelijk bouwen van de kernel is beschreven in de documentatie die met de kernel wordt geleverd. Het is tamelijk eenvoudig te volgen, dus begin door het kijken in `/usr/src/linux`. Als je bij het bouwen van een kernel problemen ondervindt, zou je eigenlijk niet moeten proberen boot/root systemen te bouwen. Denk er aan de kernel met "`make zImage`" te comprimeren.

6 Het bijelkaar plaatsen: Het aanmaken van de diskette(s).

Op dit punt heb je een kernel en een gecomprimeerd root-bestandssysteem. Controleer de grootte als je een boot/root-disk aan het maken bent om er zeker van te zijn dat ze beiden op één disk passen. Controleer het root-bestandssysteem als je er zeker van wilt zijn dat het op een enkele diskette past als je een uit twee disks bestaande boot+root set aan het maken bent.

Je zou een beslissing moeten nemen of je LILO wilt gebruiken om de bootdisk kernel te booten. Het alternatief is de kernel direct naar de diskette te kopiëren en zonder LILO te booten. Het voordeel bij gebruik van LILO is dat het je de mogelijkheid biedt een aantal parameters aan de kernel te leveren die mogelijk nodig zijn om je hardware te initialiseren (Controleer op je systeem het bestand `/etc/lilo.conf`. Als het voorkomt en er een regel in staat als "`append=...`", heb je deze feature waarschijnlijk nodig). Het nadeel van het gebruik van LILO is dat het bouwen van de bootdisk wat gecompliceerder is en wat meer ruimte in beslag neemt. Je zult een klein apart bestandssysteem in moeten stellen, wat we het **kernel bestandssysteem** zullen noemen, waarnaar we de kernel en een paar andere bestanden die LILO nodig heeft zullen transporteren.

Lees verder, als je LILO gaat gebruiken; ga naar sectie 6.2 (Zonder gebruik van LILO) als je de kernel direct gaat transporteren.

6.1 De kernel met LILO transporteren.

Het eerste wat je doen moet is een klein configuratiebestand voor LILO aanmaken. Het zou er ongeveer zo uit moeten komen te zien:

```
boot      =/dev/fd0
install   =/boot/boot.b
map       =/boot/map
read-write
backup    =/dev/null
compact
image     = KERNEL
```

```
label     = Bootdisk
root      =/dev/fd0
```

Kijk in de gebruikersdocumentatie van LILO, voor een uitleg van deze parameters. Je zal waarschijnlijk tevens een `append=...` regel aan dit bestand toe willen voegen vanaf je harddisk's bestand `/etc/lilo.conf`.

Bewaar dit bestand als `bdlilo.conf`.

Je zal nu een klein bestandssysteem aan moeten maken, wat we een **kernel bestandssysteem** zullen noemen, om het te onderscheiden van het root-bestandssysteem.

Zoek als eerste uit hoe groot het bestandssysteem zou moeten zijn. Neem de grootte van je kernel in blokken (de grootte weergegeven door "`ls -l KERNEL`" gedeeld door 1024 en naar boven afgerond) en tel daar 50 bij op. Vijftig blokken is bij benadering de ruimte die nodig is voor inodes plus nog wat andere bestanden. Je kunt dit aantal exact berekenen als je dit wilt, of gewoon 50 gebruiken. Als je een diskset bestaande uit twee disks aan het maken bent, kun je de ruimte net zo goed te hoog ramen, aangezien de eerste disk toch slechts alleen wordt gebruikt voor de kernel. Noem dit aantal `KERNEL_BLOCKS`.

Plaats een diskette in de drive (ter vereenvoudiging gaan we uit van `/dev/fd0`) en maak er een ext2 kernel bestandssysteem op aan:

```
mke2fs -i 8192 -m 0 /dev/fd0 KERNEL_BLOCKS
```

De "`-i 8192`" geeft aan dat we één inode per 8192 bytes willen. Mount vervolgens het bestandssysteem, verwijder de directory `lost+found`, en maak de `dev` en `boot` directory's voor LILO aan:

```
mount /dev/fd0 /mnt
rm -rf /mnt/lost+found
mkdir /mnt/{boot,dev}
```

Maak vervolgens de devices `/dev/null` en `/dev/fd0` aan. Je kunt in plaats van de device-nummers op te zoeken, ze vanaf je harddisk kopiëren door gebruik te maken van de `-R`:

```
cp -R /dev/{null,fd0} /mnt/dev
```

LILO heeft een kopie van de bootloader `boot.b` nodig, die je van je harddisk kan halen. Het wordt gewoonlijk in de directory `/boot` bewaard.

```
cp /boot/boot.b /mnt/boot
```

Kopieer tenslotte het configuratiebestand van LILO dat je in de laatste sectie aanmaakte samen met je kernel. Beiden kunnen in de root-directory worden geplaatst:

```
cp bdlilo.conf KERNEL /mnt
```

Alle benodigdheden voor LILO bevinden zich nu op het kernel-bestandssysteem, dus je bent er klaar voor het uit te voeren. LILO's `-r` vlag wordt gebruikt voor het installeren van de bootloader op een andere root:

```
lilo -v -C bdlilo.conf -r /mnt
```

LILO zou zonder fout moeten draaien, waarna het kernel-bestandssysteem er ongeveer zo uit zou moeten zien:

```
total 361
  1 -rw-r--r--   1 root     root          176 Jan 10 07:22 bdlilo.conf
  1 drwxr-xr-x   2 root     root         1024 Jan 10 07:23 boot/
  1 drwxr-xr-x   2 root     root         1024 Jan 10 07:22 dev/
358 -rw-r--r--   1 root     root       362707 Jan 10 07:23 vmlinuz
boot:
total 8
  4 -rw-r--r--   1 root     root         3708 Jan 10 07:22 boot.b
  4 -rw-------   1 root     root         3584 Jan 10 07:23 map
dev:
total 0
  0 brw-r-----   1 root     root       2,   0 Jan 10 07:22 fd0
  0 crw-r--r--   1 root     root       1,   3 Jan 10 07:22 null
```

Maak je geen zorgen als de bestandsgroottes bij jou iets anders uitpakken.

Laat de disk nu in de drive en ga naar sectie 6.3 (Instellen van het ramdisk word).

6.2 Transporteren van de kernel zonder LILO.

Transporteer de kernel met het `dd`-commanod als je `geen` gebruik maakt van LILO:

```
% dd if=KERNEL of=/dev/fd0 bs=1k
353+1 records in
353+1 records out
```

In dit voorbeeld schreef `dd` 353 volledige records + 1 gedeeltelijk record weg, dus de kernel neemt de eerste 354 blokken van de diskette in beslag. Noem dit aantal `KERNEL_BLOCKS` en denk er aan het in de volgende sectie te gebruiken.

Stel het root-device zo in dat het de diskette zelf is, en stel de root dan in dat het read/write zal worden geladen:

```
rdev /dev/fd0 /dev/fd0
rdev -R /dev/fd0 0
```

Wees voorzichting door de hoofdletter `-R` te gebruiken in het tweede `rdev` commando.

6.3 Instellen van het ramdisk word.

Binnenin de kernel-image bevindt zich het **ramdisk word** dat aangeeft waar het root-bestandssysteem te vinden is, samen met nog wat andere opties. Het word kan worden benaderd en ingesteld via het commando `rdev`, en de inhoud ervan wordt als volgt geïnterpreteerd:

```
bits  0-10:     Offset start ramdisk, in 1024 byte blokken
bits 11-13:     ongebruikt
bit     14:     Vlag aangevend dat ramdisk wordt geladen
bit     15:     Vlag aangevend een melding te geven alvorens rootfs
                te laden
```

Als bit 15 is ingesteld, zal je tijdens het booten worden gevraagd een nieuwe diskette in de drive te doen. Dit is nodig voor een uit twee disk bestaande bootset.

Er zijn twee situaties, afhankelijk van of je een enkele boot/root-diskette aan het bouwen bent of een dubbele "boot+root" diskette-set.

1. Als je een enkele disk aan het bouwen bent, zal het gecomprimeerde root-bestandssysteem direct achter de kernel worden geplaatst, dus zal de offset het eerste vrije blok zijn (wat hetzelfde zou moeten zijn als `KERNEL_BLOCKS`). Bit 14 zal worden ingesteld op 1, en bit 15 zal nul zijn.

   Stel bijvoorbeeld dat je een enkele disk aan het bouwen bent en dat het root-bestandssysteem begint op blok 253 (decimaal). De ramdisk word waarde zou 253 (decimaal) moeten zijn met bit 14 ingesteld op 1 en bit 15 ingesteld op 0. Voor het berekenen van de waarde kun je de decimale waarden eenvoudigweg bij elkaar optellen. 253 + (2^14) = 253 + 16384 = 16637. Als je niet geheel begrijpt waar dit nummer vandaan komt, tik het dan in in een wetenschappelijk rekenmachine en converteer het naar binair.

2. Als je een diskset bestaande uit twee disks aan het bouwen bent, zal het root-bestandssysteem beginnen op blok nul van de tweede disk, dus zal het offset nul zijn. Bit 14 zal op 1 zijn ingesteld en bit 15 zal 1 zijn. De decimale waarde zal in dit geval 2^14 + 2^15 = 49152 zijn.

Stel het met `rdev -r` in na het zorgvuldig te hebben berekend van de waarde voor het ramdisk word. Wees er zeker van de *decimale* waarde te gebruiken. Het argument aan `rdev` zou hier het *gemounte kernel path*, b.v. `/mnt/vmlinuz`, moeten zijn, als je LILO gebruikte; als je in plaats daarvan de kernel met `dd` kopieerde, gebruik dan de naam van het diskette-device (*b.v.*, `/dev/fd0`).

```
rdev -r KERNEL_OR_FLOPPY_DRIVE  VALUE
```

Unmount de diskette nu als je LILO gebruikte.

6.4 Transporteren van het root-bestandssysteem

De laatste stap bestaat uit het transporteren van het root-bestandssysteem.

- Als het root-bestandssysteem op *dezelfde* disk zal worden geplaatst als de kernel, transporteer het dan met behulp van `dd` met de optie `seek`, waarmee wordt opgegeven hoeveel blokken over te slaan:

  ```
  dd if=rootfs.gz of=/dev/fd0 bs=1k seek=KERNEL_BLOCKS
  ```

- Verwijder de eerste diskette en plaats de tweede diskette in de drive als het root-bestandssysteem op een *tweede* disk zal worden geplaatst, en tranporteer het root-bestandssysteem er dan naar:

  ```
  dd if=rootfs.gz of=/dev/fd0 bs=1k
  ```

Gefeliciteerd, je bent klaar! *Je zou een bootdisk voor dat je het opzijlegt voor een noodgeval altijd moeten proberen!* Lees verder als het niet lukt ervan te booten.

7 Probleemoplossing

Bij het bouwen van bootdisks, zal het systeem bij de eerste pogingen waarschijnlijk niet booten. De algemene benadering bij het bouwen van een root-disk is componenten vanuit je bestaande systeem te assembleren, en het op een diskette gebaseerd systeem te krijgen en uit te proberen tot op het punt waar het berichten op

de console weergeeft. Zodra het éénmaal met je begint te communiceren, is het halve leed geleden omdat je dan kunt zien waar het problemen mee heeft, en kun je individuele problemen herstellen net zolang tot het systeem soepel werkt. Als het systeem zonder verklaring hangt, kan het uitzoeken van de oorzaak moeilijk zijn. Om een systeem geboot te krijgen tot die fase waarin het met je zal communiceren, is het vereist dat verscheidene componenten aanwezig zijn en correct zijn geconfigureerd. De aanbevolen procedure voor het onderzoeken van het probleem waar het systeem niet met je zal communiceren is als volgt:

- Mogelijk zie je een melding als:

  ```
  Kernel panic: VFS: Unable to mount root fs on XX:YY
  ```

  Dit is een algemeen probleem en het kan slechts door een paar dingen worden veroorzaakt. Vergelijk allereerst het device *XX:YY* met de lijst device-codes; is het 't juiste root-device? Als dit niet zo is, heb je waarschijnlijk de `rdev -R` niet uitgevoerd, of paste je het toe op het verkeerde image. Als de device-code correct is, controleer dan nauwkeurig de device-drivers die in je kernel zijn gecompileerd. Overtuig jezelf ervan dat er ingebouwde ondersteuning voor een diskette, ramdisk en ext2 bestandssysteem in de kernel voorkomt.

- Controleer of de root-disk echt die directory's bevat waarvan je denkt dat ze er op voorkomen. Het is heel eenvoudig op het verkeerde niveau iets te kopiëren waardoor je op je root-diskette uiteindelijk eindigt met iets als `/rootdisk/bin` in plaats van `/bin`.

- Controleer of er een `/lib/libc.so` met dezelfde link die voorkomt in je `/lib` directory op je harddisk is.

- Controleer of alle symbolische links in je `/dev` directory op je bestaande systeem ook voorkomen in het bestandssysteem op je root-diskette, waar die links naar devices zijn die je op je root-diskette hebt opgenomen. In het bijzonder zijn de `/dev/console` links in veel gevallen essentieel.

- Controleer of je `/dev/tty1`, `/dev/null`, `/dev/zero`, `/dev/mem`, `/dev/ram` en `/dev/kmem` bestanden hebt opgenomen.

- Controleer je kernelconfiguratie – ondersteuning voor alle bronnen die nodig zijn tot op het punt van inloggen mogen geen modules zijn, maar zijn ingebouwd. Dus *ramdisk en ext2 ondersteuning moeten zijn ingebouwd.*

- Controleer of je kernel root-device en ramdisk instellingen correct zijn.

Nu dat we deze algemene aspecten éénmaal hebben gehad, zijn hier nog een aantal specifiekere bestanden om te controleren:

1. Zorg ervoor dat `init` is opgenomen als `/sbin/init` of `/bin/init`. Zorg ervoor dat het uitvoerbaar is.

2. Voer `ldd init` uit om op de library's van init te controleren. Meestal is dit gewoon `libc.so`, maar controleer het toch maar. Zorg ervoor dat je de benodigde library's en loaders hebt opgenomen.

3. Zorg ervoor dat je de juiste loader voor je library's hebt – `ld.so` voor a.out of `ld-linux.so` voor ELF.

4. Controller `/etc/inittab` op het bestandssysteem van je bootdisk op aanroepen naar `getty` (of een op `getty`-lijkend programma, zoals `agetty`, `mgetty` of `getty_ps`). Controleer deze tweemaal met `inittab` op je harddisk. Controleer de manpages van het te gebruiken programma om er zeker van te zijn dat deze zin hebben. `inittab` is mogelijk het lastigste onderdeel omdat de syntax en inhoud ervan afhangen van het in gebruik zijnde init programma en de natuur van het systeem. De enige manier om het aan te pakken is de manpages van `init` en `inittab` te lezen en exact uit te werken wat je bestaande systeem

doet wanneer het boot. Controleer voor de zekerheid of `/etc/inittab` een systeeminitialisatie-entry heeft. Hierin zou een commando moeten staan voor het uitvoeren van het systeem initialisatiescript, dat voor moet komen.

5. Voer net als met `init`, `ldd` uit op `getty` om te zien wat het nodig heeft, en zorg ervoor dat de benodigde library-bestanden en loaders in je root-bestandssysteem zijn opgenomen.
6. Zorg ervoor dat je een shell-programma hebt ingesloten (b.v., `bash` of `ash`) die capabel is in het uitvoeren van al je rc-scripts.
7. Als je een bestand `/etc/ld.so.cache` op je rescue-disk hebt, maak het dan opnieuw aan.

Als `init` start, maar je een melding krijgt als:

```
Id xxx respawning too fast: disabled for 5 minutes
```

komt het vanuit `init`, waarmee gewoonlijk wordt aangegeven dat `getty` of `login` afsterft zodra het opstart. Controleer de `getty` en `login` uitvoerbare bestanden en de library's waar ze afhankelijk van zijn. Zorg ervoor dat de aanroepingen in `/etc/inittab` juist zijn. Als je vreemde meldingen krijgt van `getty`, kan het zijn dat de aanroepende vorm in `/etc/inittab` onjuist is. De opties van de *getty* programma's zijn variabel; zelfs van verschillende versies van `agetty` wordt gerapporteerd dat ze verschillende incompatibele aanroepende vormen hebben.

Als je een login-prompt krijgt, en je een geldige loginnaam intikt, maar het systeem vraagt je onmiddellijk om een andere loginnaam, kan het probleem te maken hebben met PAM of NSS. Zie Sectie 4.4 (PAM en NSS). Misschien dat het probleem tevens is dat je shadow passwords gebruikt en `/etc/shadow` niet naar je bootdisk kopieerde.

Als je één of ander uitvoerbaar bestand, zoals `df` probeert uit te voeren, wat zich op je rescue-disk bevindt, maar het levert je een bericht op als: `df: not found`, controleer dan op twee zaken: (1) Verzeker je ervan dat de directory met het binaire bestand zich in je PATH bevindt, en (2) zorg ervoor dat de library's (en loaders) die het programma nodig heeft er zijn.

8 Diverse onderwerpen

8.1 Terugbrengen van de grootte van het root-bestandssysteem

Soms is een root-bestandssysteem zelfs na compressie te groot voor op een diskette. Hier zijn een aantal manieren om de grootte van je bestandssysteem te beperken, opgesomd in aflopende volgorde van effectiviteit:

Verhoog de dichtheid van de disk

Standaard worden diskettes op 1440K geformatteerd, maar formaten met hogere dichtheden zijn beschikbaar. `fdformat` zal disks met de volgende formaten formatteren: 1600, 1680, 1722, 1743, 1760, 1840, en 1920. De meeste 1440K drives zullen 1722K ondersteunen en dit is wat ik altijd voor bootdisks gebruik. Zie de manpage van `fdformat` en `/usr/src/linux/Documentation/devices.txt`.

Vervang je shell

Een aantal van de populaire shells voor Linux, zoals `bash` en `tcsh`, zijn groot en vereisten heel wat library's. Lichtgewicht alternatieven bestaan, zoals `ash`, `lsh`, `kiss` en `smash`, die heel veel kleiner zijn en waarvoor minder (of geen) library's nodig zijn. De meeste van deze vervangingen van de shell zijn beschikbaar vanaf `<http://metalab.unc.edu/pub/Linux/system/shells/>`. Zorg ervoor dat iedere shell die je gebruikt capabel is commando's in alle `rc` bestanden uit te voeren die je op je bootdisk opneemt.

Strip library's en binary's

Veel library's en binary's zijn kenmerkend niet gestript (waaronder debugging symbolen). Het uitvoeren van '`file`' op deze bestanden zal je aangeven '`not stripped`' als dit zo is. Het is bij het kopiëren van binaire bestanden naar je root-bestandssysteem een goede gewoonte gebruik te maken van:

```
objcopy --strip-all FROM TO
```

Gebruik bij het kopiëren van library's:

```
objcopy --strip-debug FROM TO
```

Verplaats niet-kritieke bestanden naar een utility-disk

Als een aantal van je binary's niet onmiddellijk nodig zijn om te booten of in te loggen, kun je ze naar een utility-disk verplaatsen. Zie voor details sectie 8.3 (Bouwen van een utility-disk). Bovendien zou je kunnen overwegen tevens modules naar een utility-disk te verplaatsen.

8.2 Niet-ramdisk root-bestandssystemen

In de sectie 4 (Bouwen van een root-bestandssysteem) werden instructies gegeven voor het bouwen van een gecomprimeerd root-bestandssysteem die naar ramdisk wordt geladen zodra het systeem boot. Deze methode heeft vele voordelen, de reden dat het in het algemeen wordt gebruikt. Een aantal systemen met weinig geheugen kunnen het zich echter niet de RAM die hiervoor nodig is veroorloven, en hierop moet het root-bestandssysteem direct vanaf de diskette worden gemount.

Dergelijke bestandssystemen zijn in feite eenvoudiger te bouwen dan gecomprimeerde root-bestandssystemen omdat ze op een diskette kunnen worden gebouwd in plaats van op een ander device, en ze niet hoeven te worden gecomprimeerd. We zullen de procedure waarin het verschilt van de instructies hiervoor uiteenzetten. Houd in gedachten dat als je hiervoor kiest, je *veel minder ruimte* beschikbaar zal hebben.

1. Bereken hoeveel ruimte je beschikbaar zal hebben voor root-bestanden.

   Als je een enkele boot/root-disk aan het bouwen bent, moet je alle blokken voor de kernel plus alle blokken voor het root-bestandssysteem op die ene disk passen.

2. Door gebruik te maken van `mke2fs`, maak je een root-bestandssystem op een diskette van passende grootte.

3. Stel het bestandssysteem samen zoals hiervoor beschreven.

4. Unmount het bestandssysteem wanneer je klaar bent en transporteeer het naar een diskbestand, maar *comprimeer het niet*.

5. Transporteer de kernel naar een diskette, zoals hiervoor is beschreven. Bij het berekenen van het ramdisk word, **stel je bit 14 op nul**, om aan te geven dat het root-bestandssysteem niet naar ramdisk zal worden geladen. Voer `rdev`'s zoals beschreven uit.

6. Transporteer als voorheen het root-bestandssysteem.

Er zijn verscheidene shortcuts te nemen. Als je een uit twee disk bestaande set aan het bouwen bent, kun je het complete root-bestandssysteem direct op de tweede disk bouwen en hoef je het niet naar een harddiskbestand en dan weer terug te transporteren. Ook is het zo dat als je een enkele boot/root-disk aan het bouwen bent en gebruik maakt van LILO, je een *enkel* bestandssysteem op de gehele disk, met de kernel, LILO en root-bestanden kunt bouwen en als laatste stap gewoonweg LILO kunt uitvoeren.

8.3 Bouwen van een utility-disk

Het bouwen van een utility-disk is relatief eenvoudig — maak gewoon een bestandssysteem aan op een geformatteerde disk en kopieer er bestanden naar. Mount het handmatig nadat het systeem is geboot om het met een bootdisk te gebruiken.

In de instructies hiervoor, gaven we al aan dat de utility-disk als `/usr` gemount zou kunnen worden. In dit geval, zouden binaire bestanden in een `/bin` directory op je utility-disk geplaatst kunnen worden, zodat het plaatsen van `/usr/bin` in je path ze zal benaderen. Extra library's benodigd voor de binary's worden op de utility-disk geplaatst in `/lib`.

Er zijn verscheidene belangrijke punten om in gedachten te houden wanneer je een utility-disk aan het ontwerpen bent:

1. Plaats kritieke systeembinary's of library's niet op de utility-disk, aangezien het niet te mounten zal zijn totdat het systeem is geboot.
2. Je kunt een diskette en een floppy tape-drive niet simultaan benaderen. Dit betekent dat als je een floppy tape-drive hebt, het niet mogelijk zal zijn het te benaderen als je utility-disk is gemount.
3. Toegang tot de bestanden op de utility-disk zal traag zijn.

In appendix D (Voorbeeld utility disk directory listing) worden voorbeeldbestanden op een utility-disk getoond. Hier zijn een aantal ideeën voor bestanden die je wellicht van nut vindt: programma's voor het bestuderen en manipuleren van disks (`format, fdisk`) en bestandssystemen (`mke2fs, fsck, debugfs, isofs.o`), een lichtgewicht teksteditor (`elvis, jove`), compressie en archief- utility's (`gzip, tar, cpio, afio`), tape utility's (`mt, tob, taper`), communicatie-utility's (`ppp.o, slip.o, minicom`) en utility's voor devices (`setserial, mknod`).

9 Hoe de pro's het doen

Misschien dat je opmerkt dat de bootdisks die door belangrijke distributies zoals Slackware, RedHat of Debian geraffineerder lijken dan wat in dit document is beschreven. Professionele distributie bootdisks zijn op dezelfde principes gebaseerd als hierin is uiteengezet, maar investeren in diverse truuks omdate hun bootdisks aanvullende vereisten hebben. Ten eerste moeten ze kunnen werken met een brede variëteit aan hardware, dus moeten er een interactie met de gebruiker plaats kunnen vinden en moet het mogelijk zijn diverse device-drivers te laden. Ten tweede moeten ze zodanig zijn geprepareerd dat ze met vele verschillende installatie-opties werken, met diverse graden van automatisering. Als laatste combineren distributie bootdisks gewoonlijk installatie en rescue mogelijkheden.

Een aantal bootdisks gebruikt een mogelijkheid genaamd **initrd** (**initial ramdisk**). Dit mogelijkheid werd zo rond 2.0.x geïntroduceerd en maakt het mogelijk een kernel in twee fasen te booten. Wanneer de kernel als eerste boot, laadt het een initiële ramdisk image vanaf de bootdisk. Deze initiële ramdisk is een root-bestandssysteem waarop een programma staat dat wordt uitgevoerd voordat het feitelijke root fs wordt geladen. Dit programma inspecteert gewoonlijk de omgeving en/of vraagt de gebruiker diverse bootopties te selecteren, zoals het device van waaraf de echte rootdisk te booten. Het laadt typisch extra modules die niet in de kernel zijn gebouwd. Wanneer dit initiële programma beëindigt, laadt de kernel het feitelijke root-image en wordt het booten normaal Zie `/usr/src/linux/Documentation/initrd.txt` en `<ftp://elserv.ffm.fgan.de/pub/linux/loadlin-1.6/initrd-example.tgz>` voor verdere informatie over `initrd`.

Wat volgt zijn samenvattingen van hoe de installatiedisks van iedere distributie schijnen te werken, gebaseerd op het inspecteren van hun bestandssystemen en/of source-code. We garanderen niet dat deze informatie volledig accuraat is, of dat ze niet zijn gewijzigd sinds de vermelde versies.

Slackware (v.3.1) gebruikt een recht-door-zee LILO-boot vergelijkbaar met wat is beschreven in sectie 6.1 (Transporteren van de kernel met LILO). De Slackware bootdisk drukt een opstartmelding af ("`Welcome to the Slackware Linux bootkernel disk!`") door gebruik te maken van LILO's `message` parameter. Hiermee wordt de gebruiker geïnstrueerd zonodig een bootparameterregel in te voeren. Na het booten wordt een root-bestandssysteem geladen vanaf een tweede disk. De gebruiker roept een `setup` script aan waarmee de installatie wordt gestart. Slackware vorziet in vele verschillende kernels in plaats dat het gebruik maakt van een modulaire kernel en het hangt van de gebruiker af degene te selecteren die overeenkomt met zijn of haar hardwarebenodigdheden.

RedHat (v.4.0) maakt ook gebruik van een LILO-boot. Het laadt een gecomprimeerde ramdisk op de eerste disk, waarmee een aangepast `init` programma wordt uitgevoerd. Dit programma ondervraagt voor drivers en laadt vervolgens extra bestanden vanaf een supplemental disk als dit nodig is.

Debian (v.1.3) is waarschijnlijk het meest geraffineerd van de installatie diskssets. Het maakt gebruik van de SYSLINUX loader om diverse load opties te regelen, vervolgens gebruikt het een `initrd` image om de gebruiker door de installatie te leiden. Het schijnt zowel gebruik te maken van een aangepaste `init` als een aangepaste shell.

10 Lijst met veelgestelde vragen

V. Ik boot vanaf mijn boot/root-disks en er gebeurt niks. Wat kan ik doen?

Zie hiervoor de sectie 7 (Probleemoplossing).

V. Hoe werkt de Slackware/Debian/RedHat bootdisk?

Zie sectie 9 (Hoe de pro's het doen), above.

V. Hoe kan ik een bootdisk maken met een XYZ-driver?

De eenvoudigste manier is een Slackware kernel vanaf je dichtsbijzijnde Slackware mirror site te verkrijgen. Slackware kernels zijn algemene kernels waarin geprobeerd wordt drivers voor zoveel mogelijk devices op te nemen, dus als je een SCSI of IDE controller hebt, bestaat de kans dat er een driver voor in de Slackware kernel is opgenomen.

Ga naar de directory `a1` en selecteer óf IDE óf een SCSI-kernel afhankelijk van het type controller dat je hebt. Controleer het bestand xxxxkern.cfg voor de geselecteerde kernel om de drivers die in die kernel zijn opgenomen te zien. Als het device in die lijst voorkomt, dan zou met de corresponderende kernel de computer moeten booten. Download het bestand xxxxkern.tgz en kopieer het naar je bootdiskette zoals in de sectie over het maken van bootdisks hiervoor is beschreven.

Je moet vervolgens met behulp van het commando rdev het root-device in de kernel controleren:

```
rdev zImage
```

`rdev` zal dan het huidige root-device in de kernel weergeven. Als dit niet hetzelfde is als het gewenste root-device, gebruik dan `rdev` om het te wijzigen. De kernel die ik bijvoorbeeld probeerde was ingesteld op `/dev/sda2`, maar mijn root SCSI-partitie is `/dev/sda8`. Om een root-diskette te gebruiken, zou je het volgende commando hebben gebruikt:

```
rdev zImage /dev/fd0
```

Als je tevens wilt weten hoe je een Slackware root-disk in kunt stellen, nou ja, jammer dan, dat valt buiten het kader van deze HOWTO, dus raad ik je aan de Linux Install Guide te raadplegen of de Slackware distributie op te halen. Zie de sectie in deze HOWTO getiteld "Referenties".

V. Hoe update ik mijn boot-diskette met een nieuwe kerenl?

Kopieer de kernel voor een boot-diskette zonder bestandssysteem met het dd commando naar je boot-diskette of gebruik het cp commando voor een boot/root-disk. Raadpleeg in deze HOWTO de sectie getiteld "Boot" voor details over het aanmaken van een boot-disk. De beschrijving geldt evenzo voor het updaten van een kernel op een bootdisk.

V. Hoe werk ik mijn root-diskette bij met nieuwe bestanden?

De eenvoudigste manier is het bestandssysteem van de rootdisk naar het gebruikte DEVICE terug te kopiëren (vanuit de sectie 4.2 (Aanmaken van het bestandssysteem) van hiervoor). Mount vervolgens het bestandssysteem en maak de wijzigingen. Je zult moeten onthouden waar je root-bestandssysteem begon en hoeveel blokken het in beslag nam:

```
dd if=/dev/fd0 bs=1k skip=ROOTBEGIN count=BLOCKS | gunzip > DEVICE
mount -t ext2 DEVICE /mnt
```

Na het maken van de wijzigingen ga je verder als voorheen (in Sectie 4.7 (Het samenpakken)) en transporteer het root-bestandssysteem terug naar de disk. Het is niet nodig de kernel te hertranspoteren of het ramdisk word te herberekenen als je de startpositie van het nieuwe root-bestandssysteem niet wijzigt.

V. Hoe verwijder ik LILO zodat ik DOS weer kan gebruiken om te booten?

Dit is niet echt een Bootdisk onderwerp, maar het wordt vaak gevraagd. Onder Linux, kun je opgeven:

```
/sbin/lilo -u
```

Je kunt ook het commando `dd` gebruiken om de door LILO bewaarde backup naar de bootsector te kopiëren. Raadpleeg de LILO-documentatie als je dit wilt doen.

Onder DOS en Windows kun je het DOS-commando gebruiken:

```
FDISK /MBR
```

MBR staat voor Master Boot Record, en het vervangt de bootsector door een opgeschoond DOS-exemplaar, zonder effect te hebben op de partitie-tabel. Een aantal puristen zijn het hier niet mee eens, maar zelfs de auteur van LILO, Werner Almesberger raadt het aan. Het is eenvoudig en het werkt.

V. Hoe kan ik booten als ik mijn kernel *en* mijn bootdisk niet meer heb?

Als je geen bootdisk bij de hand hebt, is waarschijnlijk de eenvoudigste methode een Slackware kernel voor je type diskcontroller (IDE of SCSI) te verkrijgen zoals hiervoor is beschreven in "Hoe maak ik een bootdisk met een XXX driver?". Je kunt je computer dan met behulp van deze kernel booten, en de opgelopen schade herstellen.

De kernel die je krijgt is mogelijk niet ingesteld op het root-device voor het type disk en de partitie die je wilt. Slackware's algemene SCSI-kernel heeft het root-device bijvoorbeeld ingesteld op `/dev/sda2`, terwijl mijn root Linux-partitie zich bevindt op `/dev/sda8`. In dit geval zal het root-device in de kernel moeten worden gewijzigd.

Je kunt nog steed het root-device en de ramdisk instellingen in de kernel wijzigeen ook als heb je alleen een kernel en een ander besturingssysteem zoals DOS.

`rdev` wijzigt kernel-instellingen door het wijzigen van de waarden op vastgestelde offsets in het kernelbestand, dus je kunt hetzelfde doen als je een hex-editor beschikbaar hebt onder welk systeem je dan ook draait – zoals bijvoorbeeld met de Norton Utilities Disk Editor onder DOS. Je moet dan op de volgende offsets de waarden in de kernel controleren en zonodig wijzigen:

```
HEX     DEC  DESCRIPTION
0x01F8  504  Low byte van RAMDISK word
0x01F9  505  High byte van RAMDISK word
0x01FC  508  Root minor device nummer - zie hieronder
0X01FD  509  Root major device nummer - zie hieronder
```

De interpretatie van het ramdisk is hiervoor beschreven in Sectie 6.3 (Instellen van het ramdisk word).

De major en minor device-nummers moeten worden ingesteld naar het device waarop je je root-bestandssysteem wilt mounten. Een aantal nuttige uit te kiezen waarden zijn:

```
DEVICE          MAJOR MINOR
/dev/fd0            2     0   1e diskettestation
/dev/hda1           3     1   partitie 1 op 1e IDE-disk
/dev/sda1           8     1   partitie 1 op 1e SCSI-disk
/dev/sda8           8     8   partitie 8 op 1e SCSI-disk
```

Zodra je deze waarden hebt ingesteld, kun je het bestand naar een diskette met behulp van óf de Norton Utilities Disk Editor, óf een programma genaamd `rawrite.exe` wegschrijven. Dit programma is in alle distributies opgenomen. Het is een DOS-programma waarmee een bestand naar "raw" disk wordt weggeschreven, te beginnen bij de bootsector, in plaats dat het naar het bestandssysteem wordt weggeschreven. Als je Norton Utilities gebruikt, moet je het bestand naar een fysieke disk wegschrijven te beginnen aan het begin van de disk.

V. Hoe kan ik extra kopieën maken van boot/root-diskettes?

Omdat magnetiche media mettertijd verslechteren, zou je verscheidene kopiën van je rescue-disk moeten bewaren, voor het geval het origineel onleesbaar is.

De eenvoudigste wijze om kopiën van een diskette te maken, waaronder opstarbare en utility-diskettes, is gebruik te maken van het commando `dd` om de inhoud van de oorspronkelijke diskette naar een bestand op je harddisk te kopiëren en dan hetzelfde commando te gebruiken om het bestand terug naar een nieuw diskette te kopiëren. Het is niet nodig de diskettes te mounten en je zou dit ook niet moeten doen, omdat `dd` gebruik maakt van de raw device interface.

Typ voor het kopiëren van het origineel het commando:

```
dd if=DEVICENAME of=FILENAME
waar    DEVICENAME is de device naam van de diskette drive
en      FILENAME de naam is van het (hard-disk) uitvoerbestand
```

Het achterwege laten van de parameter `count` zorgt ervoor dat dd de gehele diskette kopieert (2880 blokken voor een high-density disk).

Doe de nieuwe diskette in het station en tik het omgekeerde commando in om het resulterende bestand naar een nieuwe diskette terug te kopiëren:

```
dd if=FILENAME of=DEVICENAME
```

In deze beschrijving wordt ervan uitgegaan dat je slechts één diskettestation hebt. Als je er twee van hetzelfde type hebt, kun je diskettes met een commando kopiëren als:

```
dd if=/dev/fd0 of=/dev/fd1
```

V. Hoe kan ik booten zonder ieder keer ähaxxxx=nn,nn,nnïn te tikken?

Waar een diskdevice niet automatisch kan worden gedetecteerd is het nodig de kernel met een commando een device parameterstring op te geven, zoals:

```
aha152x=0x340,11,3,1
```

Deze parameterstring kan op verscheidene manieren met behulp van LILO worden aangeleverd:

- Door het iedere keer op de commandoregel in te voeren als het systeem wordt geboot via LILO. Dit is echter vervelend.
- Door gebruik te maken van het LILO "`lock`" sleutelwoord om te maken dat de commandoregel als de standaard commandoregel wordt bewaard, zodat LILO iedere keer dat het boot dezelfde opties zal gebruiken.
- Door het gebruik van het `append=` statement in het LILO configuratiebestand. De parameterstring moet worden omsloten door aanhalingstekens.

Een voorbeeld van een commandoregel door gebruik te maken van de parameterstring van hierboven zou zijn:

```
zImage  aha152x=0x340,11,3,1 root=/dev/sda1 lock
```

Hiermee zou de parameterstring van het device worden doorgegeven, en ook de kernel worden gevraagd het rootdevice op `/dev/sda1` in te stellen en de gehele commandoregel op te slaan en het voor alle toekomstige boots te hergebruiken.

Een voorbeeldopdracht van APPEND is:

```
APPEND = "aha152x=0x340,11,3,1"
```

De parameterstring moet op de commandoregel NIET door dubbele aanhalingstekens worden omsloten, maar in de APPEND-opdracht MOET het wel door dubbele aanhalingstekens worden omsloten.

Merk ook op dat de kernel de driver voor dat type disk moet bevatten waarop de parameterstring betrekking heeft. Als dit niet zo is, dan is er niets luisterend voor de parameterstring, en zal je de kernel opnieuw moeten bouwen waarbij je de benodigde driver insluit. cd naar `/usr/src/linux` en lees de README en lees de Linux FAQ en Installatie HOWTO voor details over het opnieuw bouwen van de kernel. Als alternatief zou je een algemene kernel voor het type disk kunnen verkrijgen en dat installeren.

Lezers worden sterk aangespoord de LILO documentatie te lezen voordat zij met de LILO-installatie gaan experimenteren. Onzorgvuldig gebruik van het `BOOT` statement kan partities beschadigen.

V. Tijdens het booten krijg ik de fout "`A: cannot execute B`". Waarom?

Er zijn verscheidene gevallen waarbij namen van programma's in diverse utilities zijn ingeprogrammeerd (hardcoded). Deze gevallen vinden niet overal plaatst, maar het kan een verklaring geven waarom een uitvoerbaar bestand blijkbaar niet op je systeem is te vinden, zelfs al zie je dat het er is. Je kunt er achter komen of een gegeven programma de naam van een ander programma heeft ingeprogrammeerd door gebruik te maken van het commando `strings` en de uitvoer door `grep` te sturen.

Bekende voorbeelden van hardcoding zijn:

- Shutdown heeft in een aantal versies `/etc/reboot` ingeprogrammeerd, dus moet `reboot` in de directory `/etc` directory worden geplaatst.
- `init` heeft voor tenminste één persoon problemen veroorzaakt, waarbij de kernel niet in staat is `init` op te sporen.

Om deze problemen te herstellen, verplaats je deze programma's óf naar de juiste directory, of wijzig je de configuratiebestanden (b.v. `/inittab`) dat ze naar de juiste directory verwijzen. Plaats bij twijfel de programma's in dezelfde directory's als waar ze op je harddisk staan, en gebruik dezelfde `inittab` en `/etc/rc.d` bestanden zoals ze op je harddisk voorkomen.

V. Mijn kernel heeft ondersteuning voor een ramdisk, maar initialiseert ramdisks van 0K

Waar dit plaatsvindt, zal tijdens het booten van de kernel een vergelijkbare melding als de volgende verschijnen:

```
Ramdisk driver initialized : 16 ramdisks of 0K size
```

Dit komt waarschijnlijk doordat de grootte tijdens de systeemstart door kernelparameters op 0 is ingesteld. Dit zou kunnen zijn veroorzaakt door een over het hoofd geziene parameter in het configuratiebestand van LILO:

```
ramdisk= 0
```

Dit kwam voor in voorbeelden van configuratiebestanden van LILO in een aantal oudere distributies, en was daar geplaatst om een eventuele voorgaande kernelinstelling te overschrijven. Verwijder een dergelijke regel als 't voorkomt.

Als je probeert een ramdisk te gebruiken die is ingesteld op 0K, het gedrag onvoorspelbaar kan zijn, en het in kernel panics kan resulteren.

A Bronnen en verwijzingen.

Neem altijd de laatste versie wanneer je een package ophaalt, tenzij je er een goede reden voor hebt dit niet te doen.

A.1 Voorgefabriceerde bootdisks.

Deze bronnen zijn voor distributie-bootdisks. *Maak alsjeblieft gebruik van één van de mirror-sites om de load op deze computer te beperken.*

- *Slackware bootdisks* <http://metalab.unc.edu/pub/Linux/distributions/slackware/bootdsks.144/>, *rootdisks* <http://metalab.unc.edu/pub/Linux/distributions/slackware/rootdsks/>
  en *Slackware mirror sites* <http://www.slackware.com/getslack/>
- *RedHat bootdisks* <http://metalab.unc.edu/pub/Linux/distributions/redhat/current/i386/images/> en *Red Hat mirror sites* <http://www.redhat.com/mirrors.html>
- *Debian bootdisks* <ftp://ftp.debian.org/pub/debian/dists/stable/main/disks-i386/current/> en *Debian mirror sites* <ftp://ftp.debian.org/pub/debian/README.mirrors.html>

In aanvulling op de distributie-bootdisks, zijn de volgende rescue-diskimages beschikbaar. Tenzij anders is aangegeven, zijn ze beschikbaar in de directory <http://metalab.unc.edu/pub/Linux/system/recovery/!INDEX.html>

- `tomsrtbt`, door Tom Oehser, is een uit enkele disk bestaande boot/root-disk gebaseerd op kernel 2.0, met een grote set mogelijkheden en ondersteuningsprogramma's. Hiermee worden IDE, SCSI, tape,

netwerk-adapters, PCMCIA en meer ondersteunt. Er zijn ongeveer 100 utility-programma's en tools opgenomen voor het herstellen en terugzetten van disks. In het package zijn ook scripts opgenomen voor het de-assembleren en herconstrueren van de images zodat zonodig nieuw materiaal kan worden toegevoegd.

- `rescue02`, door John Comyns, is een rescue-disk gebaseerd op kernel 1.3.84, met ondersteuning voor IDE en Adaptec 1542 en NCR53C7,8xx. Het maakt gebruik van ELF binaries, maar het heeft genoeg commando's voor gebruik op ieder systeem. Er zijn modules die na het booten kunnen worden geladen voor alle andere SCSI-kaarten. Het zal waarschijnlijk niet werken op systemen met 4 mb ram aangezien het gebruik maakt van een 3 mb ramdisk.

- `resque_disk-2.0.22`, door Sergei Viznyuk, is een full-featured boot/root-disk gebaseerd op kernel 2.0.22 met ingebouwde ondersteuning voor, veel verschillende SCSI-controllers, en ELF/AOUT. Ook opgenomen zijn veel modules en handige utilities voor het repareren en herstellen van een harddisk.

- `cramdisk` images, gebaseerd op de 2.0.23 kernel, beschikbaar voor 4 meg en 8 meg computers. Hierin zijn math emulatie en netwerkondersteuning opgenomen (PPP en dialin script, NE2000, 3C509), of ondersteuning voor de parallelle poort ZIP-drive. Deze diskette-images zullen op een 386'r booten met 4MB RAM. MSDOS-ondersteuning is opgenomen dus je kunt het downloaden vanaf het net naar een DOS-partitie. `<http://metalab.unc.edu/pub/Linux/system/recovery/images>`

A.2 Rescue packages.

Verscheidene packages voor het aanmaken van rescue-disks zijn beschikbaar op metalab.unc.edu. Met deze packages kan je een set bestanden aangeven die moeten worden opgenomen en de software automatiseert (in verschillende mate) de aanmaak van een bootdisk. Zie `<http://metalab.unc.edu/pub/Linux/system/recovery/!INDEX.html>` voor meer informatie. **Controleer de bestandsdata zorgvuldig** – een aantal van deze packages is verscheidene jaren niet bijgewerkt en zal de aanmaak van een gecomprimeerd root-bestandssysteem naar ramdisk niet ondersteunen. Zover we weten, is Yard het enige package die dat wel doet.

A.3 Graham Chapman's shell scripts

Graham Chapman heeft een set scripts geschreven die van pas kunnen komen als voorbeelden van hoe bootdisks kunnen worden aangemaakt. In vorige versies van deze HOWTO verschenen de scripts in een bijlage, maar ze zijn uit het document verwijderd en op een webpage geplaatst:

`<http://www.zeta.org.au/grahamc/linux.html>`

Misschien dat je het prettig vindt van deze scripts gebruik te maken, maar lees de instructies zorgvuldig door, als je dit doet – als je bijvoorbeeld het verkeerde swap-device specificeert, zal je bemerken dat je root-bestandssysteem geheel en permanent is verwijderd. Overtuig jezelf ervan dat je het correct hebt geconfigureerd, voordat je het gebruikt!

A.4 LILO – de Linux loader.

Geschreven door Werner Almesberger. Uitstekende bootloader, en de documentatie bevat informatie over de inhoud van de bootsector en de beginfasen van het bootproces.

Ftp vanaf `<ftp://tsx-11.mit.edu/pub/linux/packages/lilo/>`. Het is ook beschikbaar van Metalab en mirrors.

A.5 Linux FAQ en HOWTO's

Deze zijn vanaf veel bronnen beschikbaar. Kijk naar de usenet nieuwsgroepen `news.answers` en `comp.os.linux.announce`.

De FAQ is beschikbaar vanaf `<http://linuxdoc.org/FAQ/Linux-FAQ.html>` en de HOWTO's vanaf `<http://linuxdoc.org/HOWTO/HOWTO-INDEX.html>`. De meeste documentatie voor Linux is te vinden op *de homepage van het Linux Documentatie Project* `<http://linuxdoc.org/>`.

A.6 Ramdisk usage.

Een uitstekende beschrijving van hoe de nieuwe ramdisk code werkt is te vinden in de documentatie die met de Linux-kernel wordt geleverd. Zie `/usr/src/linux/Documentation/ramdisk.txt`. Het is geschreven door Paul Gortmaker en bevat een sectie over het aanmaken van een gecomprimeerde ramdisk.

Voor meer details over het Linux-bootproces zijn hier wat verwijzingen:

- De Linux System Administrators' Guide heeft een sectie over het booten, zie `<http://linuxdoc.org/LDP/sag/c1596.html>`.
- De LILO "Technical overview" `<http://metalab.unc.edu/pub/Linux/system/boot/lilo/lilo-t-21.ps.gz>` heeft de definitieve technische, low-level beschrijving van het bootproces, tot aan waar de kernel wordt gestart.
- De broncode is de definitieve leidraad. Hieronder vind je een aantal kernelbestanden gerelateerd aan het bootproces. Als je de sourcecode van de Linux-kernel hebt, vind je deze op je computer onder `/usr/src/linux`; als alternatief heeft Shigio Yamaguchi (shigio@tamacom.com) een zeer fraaie kernelbrowser gebaseerd op hypertekst op `<http://www.tamacom.com/tour/linux/index.html>`. Hier zijn een aantal van de relevante bestanden:

  arch/i386/boot/bootsect.S,setup.S
  : Bevat assembleercode voor de bootsector.

  arch/i386/boot/compressed/misc.c
  : Bevat code voor het decomprimeren van de kernel.

  arch/i386/kernel/
  : Directory met kernel-initialisatiecode. `setup.c` bevat het ramdisk word.

  drivers/block/rd.c
  : Bevat de ramdisk-driver. De procedures `rd_load` en `rd_load_image` laden blokken vanaf een device naar een ramdisk. De procedure `identify_ramdisk_image` stelt het gevonden bestandssysteem vast en of het is gecomprimeerd.

B LILO boot foutcodes.

Vragen over deze fouten worden zo vaak in Usenet gesteld, dat we ze hier als een publieke service hebben opgenomen. Deze samenvatting is ontleend uit Werner Almsberger's LILO User Documentatie, beschikbaar op `<http://metalab.unc.edu/pub/Linux/system/boot/lilo/lilo-u-21.ps.gz>`.

Als LILO zichzelf laadt, geeft het 't woord "`LILO`" weer. Iedere letter wordt voor of na het uitvoeren van een specifieke actie afgedrukt. Als LILO op een bepaald punt faalt, kunnen de tot dusverre afgedrukte letters worden gebruikt om het probleem te identificeren.

(niets)

Geen enkel onderdeel van LILO werd geladen. LILO is óf niet geïnstalleerd, óf de partitie waarop de bootsector is gelokaliseerd is niet actief.

L

De eerste fase van de bootloader is geladen en gestart, maar het kan de tweede fase van de bootloader niet laden. De foutcode bestaande uit twee cijfers geeft het type probleem aan. (Zie ook de sectie "Disk foutcodes".) Deze situatie geeft meestal een media failure of niet passende geometrie aan (b.v. slechte diskparameters)

LI

Het lukte de eerste fase van de bootloader de tweede fase van de bootloader te laden, maar het faalde in het uitvoeren ervan. Dit kan óf door een niet passende geometrie zijn veroorzaakt óf door het verplaatsen van /boot/boot.b zonder het uitvoeren van de map-installer.

LIL

De tweede fase van de bootloader is gestart, maar het kan de descriptor tabel vanuit het map-bestand niet laden. Dit wordt typisch veroorzaakt door een media-failure of door een een niet passende geometrie.

LIL?

De tweede fase van de bootloader is op een onjuist adres geladen. Dit wordt typisch veroorzaakt door een subtiel niet passende geometrie of door het verplaatsen van `/boot/boot.b` zonder het uitvoeren van de map-installer.

LIL-

De descriptor tabel is corrupt. Dit kan óf door een niet passende geometrie worden veroorzaakt óf door het verplaatsen van /boot/map zonder de map-installer uit te voeren.

LILO

Alle onderdelen van LILO zijn succesvol geïnstalleerd.

Als de BIOS een fout signaleert wanneer LILO een boot-image probeert te laden, dan wordt de daarop betrekking hebbende foutcode weergegeven. Deze codes liggen in het bereik van `0x00` tot en met `0xbb`. Zie de LILO User Guide voor een uitleg hierover.

C Voorbeeld rootdisk directory listings

Hieronder vind je de inhoud van een voorbeeld root-bestandssysteem en een utility-diskette.

```
Root directory:
drwx--x--x   2 root     root         1024 Nov  1 15:39 bin
drwx--x--x   2 root     root         4096 Nov  1 15:39 dev
drwx--x--x   3 root     root         1024 Nov  1 15:39 etc
drwx--x--x   4 root     root         1024 Nov  1 15:39 lib
drwx--x--x   5 root     root         1024 Nov  1 15:39 mnt
drwx--x--x   2 root     root         1024 Nov  1 15:39 proc
drwx--x--x   2 root     root         1024 Nov  1 15:39 root
drwx--x--x   2 root     root         1024 Nov  1 15:39 sbin
drwx--x--x   2 root     root         1024 Nov  1 15:39 tmp
drwx--x--x   7 root     root         1024 Nov  1 15:39 usr
```

```
drwx--x--x   5 root     root         1024 Nov  1 15:39 var

/bin:
-rwx--x--x   1 root     root        62660 Nov  1 15:39 ash
-rwx--x--x   1 root     root         9032 Nov  1 15:39 cat
-rwx--x--x   1 root     root        10276 Nov  1 15:39 chmod
-rwx--x--x   1 root     root         9592 Nov  1 15:39 chown
-rwx--x--x   1 root     root        23124 Nov  1 15:39 cp
-rwx--x--x   1 root     root        23028 Nov  1 15:39 date
-rwx--x--x   1 root     root        14052 Nov  1 15:39 dd
-rwx--x--x   1 root     root        14144 Nov  1 15:39 df
-rwx--x--x   1 root     root        69444 Nov  1 15:39 egrep
-rwx--x--x   1 root     root          395 Nov  1 15:39 false
-rwx--x--x   1 root     root        69444 Nov  1 15:39 fgrep
-rwx--x--x   1 root     root        69444 Nov  1 15:39 grep
-rwx--x--x   3 root     root        45436 Nov  1 15:39 gunzip
-rwx--x--x   3 root     root        45436 Nov  1 15:39 gzip
-rwx--x--x   1 root     root         8008 Nov  1 15:39 hostname
-rwx--x--x   1 root     root        12736 Nov  1 15:39 ln
-rws--x--x   1 root     root        15284 Nov  1 15:39 login
-rwx--x--x   1 root     root        29308 Nov  1 15:39 ls
-rwx--x--x   1 root     root         8268 Nov  1 15:39 mkdir
-rwx--x--x   1 root     root         8920 Nov  1 15:39 mknod
-rwx--x--x   1 root     root        24836 Nov  1 15:39 more
-rws--x--x   1 root     root        37640 Nov  1 15:39 mount
-rwx--x--x   1 root     root        12240 Nov  1 15:39 mt
-rwx--x--x   1 root     root        12932 Nov  1 15:39 mv
-r-x--x--x   1 root     root        12324 Nov  1 15:39 ps
-rwx--x--x   1 root     root         5388 Nov  1 15:39 pwd
-rwx--x--x   1 root     root        10092 Nov  1 15:39 rm
lrwxrwxrwx   1 root     root            3 Nov  1 15:39 sh -> ash
-rwx--x--x   1 root     root        25296 Nov  1 15:39 stty
-rws--x--x   1 root     root        12648 Nov  1 15:39 su
-rwx--x--x   1 root     root         4444 Nov  1 15:39 sync
-rwx--x--x   1 root     root       110668 Nov  1 15:39 tar
-rwx--x--x   1 root     root        19712 Nov  1 15:39 touch
-rwx--x--x   1 root     root          395 Nov  1 15:39 true
-rws--x--x   1 root     root        19084 Nov  1 15:39 umount
-rwx--x--x   1 root     root         5368 Nov  1 15:39 uname
-rwx--x--x   3 root     root        45436 Nov  1 15:39 zcat

/dev:
lrwxrwxrwx   1 root     root            6 Nov  1 15:39 cdrom -> cdu31a
brw-rw-r--   1 root     root      15,   0 May  5  1998 cdu31a
crw-------   1 root     root       4,   0 Nov  1 15:29 console
crw-rw-rw-   1 root     uucp       5,  64 Sep  9 19:46 cua0
crw-rw-rw-   1 root     uucp       5,  65 May  5  1998 cua1
crw-rw-rw-   1 root     uucp       5,  66 May  5  1998 cua2
crw-rw-rw-   1 root     uucp       5,  67 May  5  1998 cua3
brw-rw----   1 root     floppy     2,   0 Aug  8 13:54 fd0
brw-rw----   1 root     floppy     2,  36 Aug  8 13:54 fd0CompaQ
brw-rw----   1 root     floppy     2,  84 Aug  8 13:55 fd0D1040
brw-rw----   1 root     floppy     2,  88 Aug  8 13:55 fd0D1120
brw-rw----   1 root     floppy     2,  12 Aug  8 13:54 fd0D360
brw-rw----   1 root     floppy     2,  16 Aug  8 13:54 fd0D720
```

```
brw-rw----   1 root     floppy     2, 120 Aug  8 13:55 fd0D800
brw-rw----   1 root     floppy     2,  32 Aug  8 13:54 fd0E2880
brw-rw----   1 root     floppy     2, 104 Aug  8 13:55 fd0E3200
brw-rw----   1 root     floppy     2, 108 Aug  8 13:55 fd0E3520
brw-rw----   1 root     floppy     2, 112 Aug  8 13:55 fd0E3840
brw-rw----   1 root     floppy     2,  28 Aug  8 13:54 fd0H1440
brw-rw----   1 root     floppy     2, 124 Aug  8 13:55 fd0H1600
brw-rw----   1 root     floppy     2,  44 Aug  8 13:55 fd0H1680
brw-rw----   1 root     floppy     2,  60 Aug  8 13:55 fd0H1722
brw-rw----   1 root     floppy     2,  76 Aug  8 13:55 fd0H1743
brw-rw----   1 root     floppy     2,  96 Aug  8 13:55 fd0H1760
brw-rw----   1 root     floppy     2, 116 Aug  8 13:55 fd0H1840
brw-rw----   1 root     floppy     2, 100 Aug  8 13:55 fd0H1920
lrwxrwxrwx   1 root     root           7 Nov  1 15:39 fd0H360 -> fd0D360
lrwxrwxrwx   1 root     root           7 Nov  1 15:39 fd0H720 -> fd0D720
brw-rw----   1 root     floppy     2,  52 Aug  8 13:55 fd0H820
brw-rw----   1 root     floppy     2,  68 Aug  8 13:55 fd0H830
brw-rw----   1 root     floppy     2,   4 Aug  8 13:54 fd0d360
brw-rw----   1 root     floppy     2,   8 Aug  8 13:54 fd0h1200
brw-rw----   1 root     floppy     2,  40 Aug  8 13:54 fd0h1440
brw-rw----   1 root     floppy     2,  56 Aug  8 13:55 fd0h1476
brw-rw----   1 root     floppy     2,  72 Aug  8 13:55 fd0h1494
brw-rw----   1 root     floppy     2,  92 Aug  8 13:55 fd0h1600
brw-rw----   1 root     floppy     2,  20 Aug  8 13:54 fd0h360
brw-rw----   1 root     floppy     2,  48 Aug  8 13:55 fd0h410
brw-rw----   1 root     floppy     2,  64 Aug  8 13:55 fd0h420
brw-rw----   1 root     floppy     2,  24 Aug  8 13:54 fd0h720
brw-rw----   1 root     floppy     2,  80 Aug  8 13:55 fd0h880
brw-rw----   1 root     disk       3,   0 May  5  1998 hda
brw-rw----   1 root     disk       3,   1 May  5  1998 hda1
brw-rw----   1 root     disk       3,   2 May  5  1998 hda2
brw-rw----   1 root     disk       3,   3 May  5  1998 hda3
brw-rw----   1 root     disk       3,   4 May  5  1998 hda4
brw-rw----   1 root     disk       3,   5 May  5  1998 hda5
brw-rw----   1 root     disk       3,   6 May  5  1998 hda6
brw-rw----   1 root     disk       3,  64 May  5  1998 hdb
brw-rw----   1 root     disk       3,  65 May  5  1998 hdb1
brw-rw----   1 root     disk       3,  66 May  5  1998 hdb2
brw-rw----   1 root     disk       3,  67 May  5  1998 hdb3
brw-rw----   1 root     disk       3,  68 May  5  1998 hdb4
brw-rw----   1 root     disk       3,  69 May  5  1998 hdb5
brw-rw----   1 root     disk       3,  70 May  5  1998 hdb6
crw-r-----   1 root     kmem       1,   2 May  5  1998 kmem
crw-r-----   1 root     kmem       1,   1 May  5  1998 mem
lrwxrwxrwx   1 root     root          12 Nov  1 15:39 modem -> ../dev/ttyS1
lrwxrwxrwx   1 root     root          12 Nov  1 15:39 mouse -> ../dev/psaux
crw-rw-rw-   1 root     root       1,   3 May  5  1998 null
crwxrwxrwx   1 root     root      10,   1 Oct  5 20:22 psaux
brw-r-----   1 root     disk       1,   1 May  5  1998 ram
brw-rw----   1 root     disk       1,   0 May  5  1998 ram0
brw-rw----   1 root     disk       1,   1 May  5  1998 ram1
brw-rw----   1 root     disk       1,   2 May  5  1998 ram2
brw-rw----   1 root     disk       1,   3 May  5  1998 ram3
brw-rw----   1 root     disk       1,   4 May  5  1998 ram4
brw-rw----   1 root     disk       1,   5 May  5  1998 ram5
```

```
brw-rw----   1 root     disk       1,   6 May  5  1998 ram6
brw-rw----   1 root     disk       1,   7 May  5  1998 ram7
brw-rw----   1 root     disk       1,   8 May  5  1998 ram8
brw-rw----   1 root     disk       1,   9 May  5  1998 ram9
lrwxrwxrwx   1 root     root            4 Nov  1 15:39 ramdisk -> ram0
***  Ik heb alleen devices opgenomen voor de IDE-partities die ik gebruik.
***  Als je SCSI gebruikt, gebruik in plaats daarvan dan de /dev/sdXX devices.
crw-------   1 root     root       4,   0 May  5  1998 tty0
crw--w----   1 root     tty        4,   1 Nov  1 15:39 tty1
crw-------   1 root     root       4,   2 Nov  1 15:29 tty2
crw-------   1 root     root       4,   3 Nov  1 15:29 tty3
crw-------   1 root     root       4,   4 Nov  1 15:29 tty4
crw-------   1 root     root       4,   5 Nov  1 15:29 tty5
crw-------   1 root     root       4,   6 Nov  1 15:29 tty6
crw-------   1 root     root       4,   7 May  5  1998 tty7
crw-------   1 root     tty        4,   8 May  5  1998 tty8
crw-------   1 root     tty        4,   9 May  8 12:57 tty9
crw-rw-rw-   1 root     root       4,  65 Nov  1 12:17 ttyS1
crw-rw-rw-   1 root     root       1,   5 May  5  1998 zero

/etc:
-rw-------   1 root     root          164 Nov  1 15:39 conf.modules
-rw-------   1 root     root          668 Nov  1 15:39 fstab
-rw-------   1 root     root           71 Nov  1 15:39 gettydefs
-rw-------   1 root     root          389 Nov  1 15:39 group
-rw-------   1 root     root          413 Nov  1 15:39 inittab
-rw-------   1 root     root           65 Nov  1 15:39 issue
-rw-r--r--   1 root     root          746 Nov  1 15:39 ld.so.cache
***  ld.so.cache is door ldconfig aangemaakt en caches library locaties.
***  Er is veel dat tijdens het booten niet werkt als ld.so.cache ontbreekt.
***  Je kunt het na het aanmaken van de bootdisk óf opnieuw aanmaken,
***  óf ldconfig op de bootdisk opnemen en het vanuit een rc.x-script
***  uitvoeren om de cache bij te werken.
-rw-------   1 root     root           32 Nov  1 15:39 motd
-rw-------   1 root     root          949 Nov  1 15:39 nsswitch.conf
drwx--x--x   2 root     root         1024 Nov  1 15:39 pam.d
-rw-------   1 root     root          139 Nov  1 15:39 passwd
-rw-------   1 root     root          516 Nov  1 15:39 profile
-rwx--x--x   1 root     root          387 Nov  1 15:39 rc
-rw-------   1 root     root           55 Nov  1 15:39 shells
-rw-------   1 root     root          774 Nov  1 15:39 termcap
-rw-------   1 root     root           78 Nov  1 15:39 ttytype
lrwxrwxrwx   1 root     root           15 Nov  1 15:39 utmp -> ../var/run/utmp
lrwxrwxrwx   1 root     root           15 Nov  1 15:39 wtmp -> ../var/log/wtmp

/etc/pam.d:
-rw-------   1 root     root          356 Nov  1 15:39 other

/lib:
*** Ik heb een ELF-systeem met glibc, dus heb ik de ld-2.so loader nodig.
-rwxr-xr-x   1 root     root        45415 Nov  1 15:39 ld-2.0.7.so
lrwxrwxrwx   1 root     root           11 Nov  1 15:39 ld-linux.so.2 -> ld-2.0.7.so
-rwxr-xr-x   1 root     root       731548 Nov  1 15:39 libc-2.0.7.so
lrwxrwxrwx   1 root     root           13 Nov  1 15:39 libc.so.6 -> libc-2.0.7.so
lrwxrwxrwx   1 root     root           17 Nov  1 15:39 libcom_err.so.2 -> libcom_err.so.2.0
```

```
-rwxr-xr-x   1 root     root         6209 Nov  1 15:39 libcom_err.so.2.0
-rwxr-xr-x   1 root     root       153881 Nov  1 15:39 libcrypt-2.0.7.so
lrwxrwxrwx   1 root     root           17 Nov  1 15:39 libcrypt.so.1 -> libcrypt-2.0.7.so
-rwxr-xr-x   1 root     root        12962 Nov  1 15:39 libdl-2.0.7.so
lrwxrwxrwx   1 root     root           14 Nov  1 15:39 libdl.so.2 -> libdl-2.0.7.so
lrwxrwxrwx   1 root     root           16 Nov  1 15:39 libext2fs.so.2 -> libext2fs.so.2.4
-rwxr-xr-x   1 root     root        81382 Nov  1 15:39 libext2fs.so.2.4
-rwxr-xr-x   1 root     root        25222 Nov  1 15:39 libnsl-2.0.7.so
lrwxrwxrwx   1 root     root           15 Nov  1 15:39 libnsl.so.1 -> libnsl-2.0.7.so
-rwx--x--x   1 root     root       178336 Nov  1 15:39 libnss_files-2.0.7.so
lrwxrwxrwx   1 root     root           21 Nov  1 15:39 libnss_files.so.1 -> libnss_files-2.0.7.so
lrwxrwxrwx   1 root     root           14 Nov  1 15:39 libpam.so.0 -> libpam.so.0.64
-rwxr-xr-x   1 root     root        26906 Nov  1 15:39 libpam.so.0.64
lrwxrwxrwx   1 root     root           19 Nov  1 15:39 libpam_misc.so.0 -> libpam_misc.so.0.64
-rwxr-xr-x   1 root     root         7086 Nov  1 15:39 libpam_misc.so.0.64
-r-xr-xr-x   1 root     root        35615 Nov  1 15:39 libproc.so.1.2.6
lrwxrwxrwx   1 root     root           15 Nov  1 15:39 libpwdb.so.0 -> libpwdb.so.0.54
-rw-r--r--   1 root     root       121899 Nov  1 15:39 libpwdb.so.0.54
lrwxrwxrwx   1 root     root           19 Nov  1 15:39 libtermcap.so.2 -> libtermcap.so.2.0.8
-rwxr-xr-x   1 root     root        12041 Nov  1 15:39 libtermcap.so.2.0.8
-rwxr-xr-x   1 root     root        12874 Nov  1 15:39 libutil-2.0.7.so
lrwxrwxrwx   1 root     root           16 Nov  1 15:39 libutil.so.1 -> libutil-2.0.7.so
lrwxrwxrwx   1 root     root           14 Nov  1 15:39 libuuid.so.1 -> libuuid.so.1.1
-rwxr-xr-x   1 root     root         8039 Nov  1 15:39 libuuid.so.1.1
drwx--x--x   3 root     root         1024 Nov  1 15:39 modules
drwx--x--x   2 root     root         1024 Nov  1 15:39 security

/lib/modules:
drwx--x--x   4 root     root         1024 Nov  1 15:39 2.0.35

/lib/modules/2.0.35:
drwx--x--x   2 root     root         1024 Nov  1 15:39 block
drwx--x--x   2 root     root         1024 Nov  1 15:39 cdrom

/lib/modules/2.0.35/block:
-rw-------   1 root     root         7156 Nov  1 15:39 loop.o

/lib/modules/2.0.35/cdrom:
-rw-------   1 root     root        24108 Nov  1 15:39 cdu31a.o

/lib/security:
-rwx--x--x   1 root     root         8771 Nov  1 15:39 pam_permit.so

***  Directory stubs voor het mounten
/mnt:
drwx--x--x   2 root     root         1024 Nov  1 15:39 SparQ
drwx--x--x   2 root     root         1024 Nov  1 15:39 cdrom
drwx--x--x   2 root     root         1024 Nov  1 15:39 floppy

/proc:

/root:
-rw-------   1 root     root          176 Nov  1 15:39 .bashrc
-rw-------   1 root     root          182 Nov  1 15:39 .cshrc
-rw-------   1 root     root           47 Nov  1 15:39 .glintrc
```

```
-rwx--x--x   1 root     root          455 Nov  1 15:39 .profile
-rw-------   1 root     root         4014 Nov  1 15:39 .tcshrc

/sbin:
-rwx--x--x   1 root     root        23976 Nov  1 15:39 depmod
-rwx--x--x   2 root     root       274600 Nov  1 15:39 e2fsck
-rwx--x--x   1 root     root        41268 Nov  1 15:39 fdisk
-rwx--x--x   1 root     root         9396 Nov  1 15:39 fsck
-rwx--x--x   2 root     root       274600 Nov  1 15:39 fsck.ext2
-rwx--x--x   1 root     root        29556 Nov  1 15:39 getty
-rwx--x--x   1 root     root         6620 Nov  1 15:39 halt
-rwx--x--x   1 root     root        23116 Nov  1 15:39 init
-rwx--x--x   1 root     root        25612 Nov  1 15:39 insmod
-rwx--x--x   1 root     root        10368 Nov  1 15:39 kerneld
-rwx--x--x   1 root     root       110400 Nov  1 15:39 ldconfig
-rwx--x--x   1 root     root         6108 Nov  1 15:39 lsmod
-rwx--x--x   2 root     root        17400 Nov  1 15:39 mke2fs
-rwx--x--x   1 root     root         4072 Nov  1 15:39 mkfs
-rwx--x--x   2 root     root        17400 Nov  1 15:39 mkfs.ext2
-rwx--x--x   1 root     root         5664 Nov  1 15:39 mkswap
-rwx--x--x   1 root     root        22032 Nov  1 15:39 modprobe
lrwxrwxrwx   1 root     root            4 Nov  1 15:39 reboot -> halt
-rwx--x--x   1 root     root         7492 Nov  1 15:39 rmmod
-rwx--x--x   1 root     root        12932 Nov  1 15:39 shutdown
lrwxrwxrwx   1 root     root            6 Nov  1 15:39 swapoff -> swapon
-rwx--x--x   1 root     root         5124 Nov  1 15:39 swapon
lrwxrwxrwx   1 root     root            4 Nov  1 15:39 telinit -> init
-rwx--x--x   1 root     root         6944 Nov  1 15:39 update

/tmp:

/usr:
drwx--x--x   2 root     root         1024 Nov  1 15:39 bin
drwx--x--x   2 root     root         1024 Nov  1 15:39 lib
drwx--x--x   3 root     root         1024 Nov  1 15:39 man
drwx--x--x   2 root     root         1024 Nov  1 15:39 sbin
drwx--x--x   3 root     root         1024 Nov  1 15:39 share
lrwxrwxrwx   1 root     root           10 Nov  1 15:39 tmp -> ../var/tmp

/usr/bin:
-rwx--x--x   1 root     root        37164 Nov  1 15:39 afio
-rwx--x--x   1 root     root         5044 Nov  1 15:39 chroot
-rwx--x--x   1 root     root        10656 Nov  1 15:39 cut
-rwx--x--x   1 root     root        63652 Nov  1 15:39 diff
-rwx--x--x   1 root     root        12972 Nov  1 15:39 du
-rwx--x--x   1 root     root        56552 Nov  1 15:39 find
-r-x--x--x   1 root     root         6280 Nov  1 15:39 free
-rwx--x--x   1 root     root         7680 Nov  1 15:39 head
-rwx--x--x   1 root     root         8504 Nov  1 15:39 id
-r-sr-xr-x   1 root     bin          4200 Nov  1 15:39 passwd
-rwx--x--x   1 root     root        14856 Nov  1 15:39 tail
-rwx--x--x   1 root     root        19008 Nov  1 15:39 tr
-rwx--x--x   1 root     root         7160 Nov  1 15:39 wc
-rwx--x--x   1 root     root         4412 Nov  1 15:39 whoami
```

```
/usr/lib:
lrwxrwxrwx   1 root     root           17 Nov  1 15:39 libncurses.so.4 -> libncurses.so.4.2
-rw-r--r--   1 root     root       260474 Nov  1 15:39 libncurses.so.4.2

/usr/sbin:
-r-x--x--x   1 root     root        13684 Nov  1 15:39 fuser
-rwx--x--x   1 root     root         3876 Nov  1 15:39 mklost+found

/usr/share:
drwx--x--x   4 root     root         1024 Nov  1 15:39 terminfo

/usr/share/terminfo:
drwx--x--x   2 root     root         1024 Nov  1 15:39 l
drwx--x--x   2 root     root         1024 Nov  1 15:39 v

/usr/share/terminfo/l:
-rw-------   1 root     root         1552 Nov  1 15:39 linux
-rw-------   1 root     root         1516 Nov  1 15:39 linux-m
-rw-------   1 root     root         1583 Nov  1 15:39 linux-nic

/usr/share/terminfo/v:
-rw-------   2 root     root         1143 Nov  1 15:39 vt100
-rw-------   2 root     root         1143 Nov  1 15:39 vt100-am

/var:
drwx--x--x   2 root     root         1024 Nov  1 15:39 log
drwx--x--x   2 root     root         1024 Nov  1 15:39 run
drwx--x--x   2 root     root         1024 Nov  1 15:39 tmp

/var/log:
-rw-------   1 root     root            0 Nov  1 15:39 wtmp

/var/run:
-rw-------   1 root     root            0 Nov  1 15:39 utmp

/var/tmp:
```

D Voorbeeldlisting van een directory op een utility-disk:

```
total 579
-rwxr-xr-x   1 root     root        42333 Jul 28 19:05 cpio*
-rwxr-xr-x   1 root     root        32844 Aug 28 19:50 debugfs*
-rwxr-xr-x   1 root     root       103560 Jul 29 21:31 elvis*
-rwxr-xr-x   1 root     root        29536 Jul 28 19:04 fdisk*
-rw-r--r--   1 root     root       128254 Jul 28 19:03 ftape.o
-rwxr-xr-x   1 root     root        17564 Jul 25 03:21 ftmt*
-rwxr-xr-x   1 root     root        64161 Jul 29 20:47 grep*
-rwxr-xr-x   1 root     root        45309 Jul 29 20:48 gzip*
-rwxr-xr-x   1 root     root        23560 Jul 28 19:04 insmod*
-rwxr-xr-x   1 root     root          118 Jul 28 19:04 lsmod*
lrwxrwxrwx   1 root     root            5 Jul 28 19:04 mt -> mt-st*
-rwxr-xr-x   1 root     root         9573 Jul 28 19:03 mt-st*
lrwxrwxrwx   1 root     root            6 Jul 28 19:05 rmmod -> insmod*
-rwxr-xr-x   1 root     root       104085 Jul 28 19:05 tar*
```

```
lrwxrwxrwx   1 root     root            5 Jul 29 21:35 vi -> elvis*
```