De Linux Bootdisk HOWTO

Tom Fawcett

fawcett+BH@croftj.net

Vertaald door: Ellen Bokhorst

bokkie@nl.linux.org

In dit document wordt beschreven hoe je eigen boot/root diskettes voor Linux te ontwerpen. Deze disks kunnen als rescue-disks worden gebruikt of om nieuwe systeemcomponenten te testen. Je zou redelijk bekend moeten zijn met systeembeheertaken voor je een poging gaat wagen je eigen bootdisk te bouwen. Zie Appendix A.1 als je slechts een rescue-disk voor noodgevallen wilt.

Inhoudsopgave

1. Voorwoord

> Het kan zijn dat dit document verouderd is. Als de datum op de titelpagina ouder is dan 6 maanden, kijk dan alsjeblieft op de Bootdisk-HOWTO homepage (http://www.croftj.net/˜fawcett/Bootdisk-HOWTO/index.html) of er een recentere versie is.

Alhoewel dit document in tekstvorm leesbaar zou moeten zijn, ziet het er in Postscript, PDF of HTML, vanwege de gebruikte typografische conventies veel beter uit.

1.1. Versie notities

Graham Chapman schreef de oorspronkelijke Bootdisk-HOWTO en hij ondersteunde het tot aan versie 3.1. Tom Fawcett begon als mede-auteur zo rond de tijd dat kernel v2 werd geïntroduceerd. Hij is de huidige beheerder van het document.

Deze informatie is bedoeld voor Linux op het Intel platform. Veel van deze informatie kan ook toepasbaar zijn voor Linux op andere processors, maar we hebben hiermee geen ervaring uit de eerste hand. Neem alsjeblieft contact met ons op als je ervaring hebt met bootdisks voor andere platformen.

1.2. Nog te doen

Enige vrijwilligers?

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

1.3. Feedback en krediet

Ik verwelkom alle feedback, goed of slecht, over de inhoud van dit document. Ik/we heb(ben) mijn/onze best gedaan er zeker van te zijn dat de instructies en de informatie in dit document accuraat en betrouwbaar zijn. Laat het me alsjeblieft weten als je fouten vindt of als er iets ontbreekt. Geef alsjeblieft het versienummer van het document op waarnaar je refereert als je schrijft.

We bedanken de vele mensen die ons assisteerde met correcties en suggesties. Hun bijdragen hebben het veel beter gemaakt dan we het ooit alleen voor elkaar zouden kunnen hebben krijgen.

Stuur opmerkingen, correcties, en vragen naar de auteur via het bovenstaande e-mailadres. Het maakt me niet uit je proberen antwoord te geven op vragen, maar als je een specifieke vraag hebt over waarom je bootdisk niet werkt, lees dan alsjeblieft eerst de Paragraaf 7 .

1.4. Distributiebeleid

2. Introductie

Linux bootdisks zijn in een aantal situaties van nut, zoals bij het testen van een nieuwe kernel, het herstellen van een diskstoring (alles van een verloren bootsector tot een crash van de diskkoppen), het herstellen van een gedeactiveerd systeem, of het veilig bijwerken van kritieke systeembestanden, (zoals `libc.so`).

Er bestaan diverse manieren om aan bootdisks te komen:

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

Een aantal mensen kiest voor de laatste optie om het zelf te kunnen doen. Dit zodat, als er iets niet goed gaat, ze het zelf op kunnen lossen. Plus dat het een geweldige manier is om te leren hoe een Linux-systeem werkt.

In dit document wordt uitgegaan van een basiskennis van Linuxsysteembeheer concepten. Je zou bijvoorbeeld bekend moeten zijn met directory's, bestandssystemen en diskettes. Je zou moeten weten hoe mount en df te gebruiken. Je zou moeten weten waar de bestanden `/etc/passwd` en `fstab` voor zijn er hoe ze er uitzien. Je zou moeten weten dat de meeste opdrachten in deze HOWTO als root zouden moeten worden uitgevoerd.

Het vanaf het begin samenstellen van je eigen bootdisk kan gecompliceerd zijn. Als je de Linux FAQ en daaraan gerelateerde documenten, zoals de Linux Installation Guide niet hebt gelezen, doe je er beter aan niet te proberen bootdiskettes te bouwen. Als je slechts een bootdisk voor noodgevallen nodig hebt, is het veel eenvoudiger een voorgefabriceerd exemplaar te downloaden. Zie Appendix A.1, waar je deze kunt vinden.

3. Bootdisks en het bootproces

Een bootdisk is eigenlijk een miniatuur, zelfbevattend 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 code in ROM (in het bijzonder de BIOS) uit te voeren om de sector vanaf sector 0, cylinder 0 van de bootdisk te laden. De bootdisk is gewoonlijk de diskette in het eerste diskettestation (toegekend als `A:` in DOS en `/dev/fd0` onder Linux). De BIOS probeert dan deze sector uit te voeren. Op de meeste opstartbare disks, bevat sector 0, cylinder 0 zowel:

- code van een bootloader zoals LILO, die de kernel lokaliseert, het laadt en het uitvoert om de boot zuiver te starten als
- de start van een besturingssysteem zoals Linux.

Als een Linux-kernel raw naar diskette is gekopieerd, zal de eerste sector de eerste sector van de Linux-kernel zelf zijn. Deze 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 een root bestandssysteem vanaf een device te laden en mounten. Een rootbestandssysteem is gewoon een bestandssysteem dat is gemount als “/”. De kernel moet worden verteld waar naar het rootbestandssysteem te zoeken; als het daar geen laadbaar image kan vinden, stopt het.

In een aantal bootsituaties - vaak wanneer van diskette wordt geboot - wordt het rootbestandssysteem in ramdisk geladen, dit is RAM die door het systeem als een disk wordt benaderd. 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.

Als het rootbestandssysteem éénmaal is geladen en gemount, zie je een melding als:

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

Hier vindt het systeem het programma `init` op het rootbestandssysteem (in `/bin` of `/sbin`) en voert het uit. `init` leest zijn configuratiebestand `/etc/inittab` in, zoekt naar een regel aangeduid met `sysinit`, en voert het in die regel genoemde script uit. Het `sysinit` script is gewoonlijk iets als `/etc/rc` of `/etc/init.d/boot`. Dit script bestaat uit een set shell-opdrachten waarmee basissysteemservices worden ingesteld, zoals:

- Het op alle disks uitvoeren van fsck,
- Het laden van de benodigde kernelmodules,
- Het starten van swappen,
- Het initialiseren van het netwerk,
- Het mounten van de disks vermeld in `fstab`.

Dit script roept vaak diverse andere scripts aan voor de modulaire initialisatie. In bijvoorbeeld de structuur van SysVinit, bevat de directory `/etc/rc.d/` een complexe structuur aan subdirectory’s waarvan de bestanden specificeren hoe de meeste systeemservices te activeren of af te sluiten. Op een bootdisk is het sysinit-script echter vaak heel simpel.

Wanneer het sysinit-script klaar is, geeft het de controle terug aan init, die dan overgaat op het standaard runlevel, met het sleutelwoord `initdefault` gespecificeerd in `inittab`. In de regel met het runlevel wordt gewoonlijk een programma als getty gespecificeerd, welke verantwoordelijk is voor het afhandelen van communicaties op de console en tty’s. Het is het getty programma welke de bekende “`login:`” prompt afdrukt. Het getty programma roept op zijn beurt het programma login aan om de loginvalidatie af te handelen en gebruikerssessies in te stellen.

3.2. Disktypes

Na het basisbootproces 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 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 die kan worden geboot. De disk kan worden gebruikt om de kernel te booten, die dan een rootbestandssysteem op een andere disk kan laden. De kernel op een bootdisk moet gewoonlijk worden aangegeven waar het zijn rootbestandssysteem kan vinden.
>
> Vaak laadt een bootdisk een rootbestandssysteem van een andere diskette, maar het is bij een bootdisk mogelijk het zo in te stellen dat het ’t in plaats daarvan een rootbestandssysteem 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 sourcecode 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 rootdisk kan onafhankelijk van enig andere disk worden gebruikt, zodra de kernel is geboot. Gewoonlijk wordt de rootdisk automatisch naar een ramdisk gekopieerd. Dit maakt dat de rootdisk veel sneller te benaderen is, en het maakt het diskettestation vrij voor een utility-disk.

boot/root

> Een disk met zowel de kernel als een rootbestandssysteem. Met andere woorden, het bevat alles wat nodig is om te booten en een Linux-systeem zonder harddisk te draaien. Het voordeel van dit type disk is dat het compact is - alles wat 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 plaatsen.

utility

> Een disk met een bestandssysteem, maar het is niet bedoeld te worden gemount als een rootbestandssysteem. Het is een aanvullende gegevensdisk. Je zou dit type disk kunnen gebruiken om aanvullende utility’s mee te vervoeren voor als je te veel hebt voor op je rootdisk.

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-/rootdisk) als apart (boot + rootdisks) voorkomen. De meest flexibele benadering voor rescue-disks is waarschijnlijk het gebruik van boot- en rootdiskettes, en één of meer utility-diskettes voor datgene wat te veel is voor op deze disks.

4. Bouwen van een rootbestandssysteem

Het aanmaken van het rootbestandssysteem bestaat uit het selecteren van de bestanden die nodig zijn om het systeem te draaien. In deze sectie wordt beschreven hoe een gecomprimeerd rootbestandssysteem te bouwen. Een minder gebruikelijke optie bestaat uit het bouwen van een ongecomprimeerd bestandssysteem op een diskette dat direct als root wordt gemount; dit alternatief wordt beschreven in de Paragraaf 8.2.

4.1. Overzicht

Op een rootbestandssysteem 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 bestandssysteemstructuuur,
- Minimum set directory's: `/dev`, `/proc`, `/bin`, `/etc`, `/lib`, `/usr`, `/tmp`,
- Basisset utility's: `sh`, `ls`, `cp`, `mv`, enz.,
- Minimumset config bestanden: `rc, inittab, fstab`, enz.,
- Devices: `/dev/hd*, /dev/tty*, /dev/fd0`, enz.,
- Runtime library om in basisfuncties die door de utility's worden gebruikt te voorzien.

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

- Het controleren van een bestandssysteem op een andere disk, om bijvoorbeeld je rootbestandssysteem op een harddisk te controleren, moet je Linux van een andere disk kunnen booten, zoals je dat kan met een rootdiskettesysteem. Vervolgens kun je fsck dan uitvoeren op je oorspronkelijke rootdisk terwijl het niet is gemount.
- Het herstellen van je gehele of gedeeltelijke oorspronkelijke rootdisk vanaf een backup door gebruik te maken van archief- en compressie- utility's zoals `cpio`, `tar`, `gzip` en `ftape`.

We zullen beschrijven hoe een gecomprimeerd bestandssysteem op een ramdisk 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 megabyte) op een standaard 1440K diskette. 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 diskette kopiëren.

4.2. Aanmaken van het bestandssysteem

Om een dergelijk rootbestandssysteem te bouwen, heb je een reserve device nodig welke groot genoeg is alle bestanden voor compressie te bevatten. Je hebt een device nodig dat capabel is om ongeveer vier megabyte te bevatten. Je hebt verscheidene keuzes:

- Gebruik een ramdisk (DEVICE = `/dev/ram0`). In dit geval wordt geheugen gebruikt om een diskette te simuleren. De ramdisk moet groot genoeg zijn voor een bestandssysteem van passende grootte. Controleer je configuratiebestand (`/etc/lilo.conf`) op een regel als `RAMDISK = nnn` waarmee het maximum RAM dat in beslag kan worden genomen door een ramdisk als je gebruik maakt van LILO. De standaardwaarde is wat voldoende zou moeten zijn. Het is beter niet te proberen een dergelijke ramdisk te gebruiken op een computer met minder dan 8MB RAM. 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 acceptabel.
- Gebruik een loopback device, waarmee het mogelijk is een diskbestand als een device te laten fungeren. Door gebruik te maken van een loopback-device kun je op je harddisk een bestand van drie megabyte 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 verkrijgen vanuit het package `util-linux` 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`, enz.) hebt, zal je het aan moeten maken met "mknod /dev/loop0 b 7 0". Zodra je deze speciale mount en umount binary's hebt geïnstalleerd, maak je op een hardisk met voldoende capaciteit een tijdelijk bestand aan (bv, `/tmp/fsfile`). Je kunt een opdracht gebruiken als:

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

om een `nnn`-blockbestand aan te maken.

Gebruik hieronder dit bestand in plaats van DEVICE. Wanneer je een mount-opdracht aanroept, moet je de optie `-o loop` aan mount meegeven om aan te geven dat mount een loopback-device gebruikt. Bijvoorbeeld:

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

zal `/tmp/fsfile` via een loopback device op het mount point `/mnt` mounten. Een df zal dit bevestigen.

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

```
dd if=/dev/zero of=DEVICE bs=1k count=4096
```

Deze opdracht vult het device op met nullen.

> Het device met nullen opvullen is van groot belang omdat het bestandssysteem later zal worden gecomprimeerd, dus alle ongebruikte delen zouden met nullen moeten worden opgevuld om een maximum compressie te bereiken. Houd dit feit in gedachten wanneer je bestanden vanaf je rootbestandssysteem verwijdert. Het bestandssysteem zal de blokken correct vrijgeven, *maar het vult ze niet weer met nullen op*. Als je veel verwijdert en kopieert, kan je gecomprimeerde bestandssysteem uiteindelijk veel groter worden dan nodig is.

Maak vervolgens het bestandssysteem aan. De Linux-kernel herkent voor rootdisks twee typen bestandssystemen die automatisch naar ramdisk worden gekopieerd. Dit zijn minix en ext2, waarvan ext2 de voorkeur heeft. Als je ext2 gebruikt, vind je het wellicht handig de optie `-i` mee te geven om meer inodes dan de standaardwaarde op te geven; `-i 2000` wordt aanbevolen zodat je geen inodes te kort komt. Als alternatief kun je op inodes besparen door veel van de onnodige `/dev` bestanden te verwijderen. mke2fs zal op een 1.44Mb diskette standaard 360 inodes aanmaken. Ik merkte dat 120 inodes op mijn huidige rescue rootdiskette ruim voldoende is, maar als je alle devices in de directory `/dev` opneemt, dan zal het de 360 makkelijk overschrijden. Het gebruik van een gecomprimeerd rootbestandssysteem maakt een groter bestandssysteem mogelijk, en vandaar standaard meer inodes, maar mogelijk moet je toch het aantal bestanden nog verminderen of het aantal inodes verhogen.

Dus de opdracht die je gaat gebruiken, ziet er ongeveer zo uit:

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

(Als je van een loopback-device gebruik maakt, moet DEVICE worden vervangen door het diskbestand).

De opdracht mke2fs zal automatisch de beschikbare ruimte detecteren en zichzelf dienovereenkomstig configureren. De parameter “`-m 0`” voorkomt dat er ruimte voor root wordt gereserveerd, en daardoor blijft er meer bruikbare ruimte op de disk over.

Mount dan het device:

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

(Je moet een mountpoint `/mnt` aanmaken als het nog niet voorkomt). In de hiernavolgende secties zal ervan worden uitgegaan dat alle namen van doeldirectory’s zich relatief ten opzichte van `/mnt` bevinden.

4.3. Het bestandssysteem vullen

Hier is een redelijke minimumset directory’s voor je rootbestandssysteem [1]:

- `/dev` -- Devices, vereist voor I/O
- `/proc` -- Directory stub vereist voor het proc-bestandssysteem
- `/etc` -- Systeemconfiguratiebestanden
- `/sbin` -- Kritieke systeembinary’s
- `/bin` -- Essentiële binary’s die worden aangemerkt als onderdeel van het systeem
- `/lib` -- Shared library’s om te voorzien in run-time support
- `/mnt` -- Een mountpoint voor het beheer van andere disks
- `/usr` -- Extra utility’s en applicaties

Drie van deze directory’s zullen op het rootbestandssysteem leeg zijn, dus ze hoeven alleen met mkdir te worden aangemaakt. De directory `/proc` is eigenlijk een stub waaronder het proc-bestandssysteem wordt geplaatst. De directory’s `/mnt` en `/usr` zijn slechts mountpoints voor gebruik nadat het boot/root systeem draait. Vandaar nogmaals, hoeven deze directory’s alleen te worden aangemaakt.

De overblijvende 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 deze kan met `mkdir` op de gebruikelijke wijze worden aangemaakt. De speciale bestanden voor de devices moeten echter op een speciale manier, met de opdracht mknod worden aangemaakt.

Er is echter een kortere weg. Kopieer de inhoud van je bestaande `/dev` directory, en verwijder die bestanden die je niet wilt. Het enige waar je op moet letten is dat je de speciale bestanden voor de devices kopieert met de optie `-R`. Hiermee zal de directory worden gekopieerd zonder dat er zal worden geprobeerd de inhoud van de bestanden te kopiëren. Zorg ervoor dat je de hoofdletter *`R`* gebruikt. De opdracht is:

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

ervan uitgaande dat de diskette is gemount op `/mnt`. De `dp` switches zorgen ervoor dat symbolische links als links worden gekopieerd in plaats van dat het doelbestand wordt gebruikt en dat de oorspronkelijke bestandskenmerken blijven behouden, dus dat de informatie over de eigenaren blijft behouden.

Als je het op een moeilijke manier wilt doen, gebruik je `ls -l` om de major en minor device-nummers voor de gewenste devices weer te geven, en maak je ze aan op de diskette met `mknod`.

Alhoewel de devices zijn gekopieerd, loont het de moeite na te kijken dat alle door jou benodigde devices op de rescue-diskette zijn geplaatst. `ftape` maakt bijvoorbeeld gebruik van tape devices, dus zal je alle tape devices moeten kopiëren als je van plan bent je floppy tapedrive vanaf de bootdisk te benaderen.

Voor ieder speciaal apparaatbestand 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 directory `/dev` van de diskette te verwijderen. Als je bijvoorbeeld geen SCSI-disks hebt, kun je alle apparaatbestanden te beginnen met `sd` gerust verwijderen. Op vergelijkbare wijze kunnen alle apparaatbestanden beginnend met `cua` worden verwijderd, als je niet van plan bent je seriële poort te gaan gebruiken.

Zorg er in ieder geval voor dat je de volgende bestanden in deze directory opneemt: `console`, `kmem`, `mem`, `null`, `ram0` en `tty1`.

4.3.2. /etc

In deze directory staan belangrijke configuratiebestanden. Op de meeste systemen kunnen deze in drie groepen worden onderverdeeld:

1. Ten alle tijden vereist, b.v. `rc`, `fstab`, `passwd`.
2. Mogelijk nodig, maar niemand is daar al te zeker van.
3. Rommel die erin is geslopen.

Niet essentiële bestanden kunnen gewoonlijk worden geïdentificeerd met de opdracht:

```
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 rootdiskette achterwege worden gelaten.

Op mijn rootdiskettes heb ik het aantal configuratiebstanden 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 benodigd bij het wijzigen van het runlevel.
   b. `fstab` -- lijst met te mounten bestandssystemen
   c. `inittab` -- parameters voor het init-proces, het eerste proces dat bij de systeemstart wordt gestart.
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. Wellicht dat je deze niet hebt.
   d. `termcap` -- de terminal capaciteiten database.

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

Zorg ervoor dat in `passwd` om z'n minst `root` voorkomt. Als je van plan bent andere gebruikers in te laten loggen, zorg er dan voor dat de directory's en shells aanwezig zijn.

`termcap`, de terminal database is meestal verscheidene kilobytes groot. De versie die je op je boot-/rootdiskette gebruikt, zou zo moeten worden geoptimaliseerd, dat het slechts de te gebruiken termainal(s) bevat, wat gewoonlijk slechts het `linux` of `linux-console` veld is.

3. De rest. Ze zijn op het 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 juist zijn. Het is niet echt nodig hostname uit te voeren - het ziet er gewoon fraaier uit als je het wel 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 velden vanuit je bestaande `fstab` kopiëren, maar je zou de harddisk-partities niet automatisch moeten mounten; gebruik in plaats daarvan de optie `noauto`. Je harddisk kan beschadigd of onbereikbaar zijn als de bootdisk wordt gebruikt.

Je moet je `inittab` zodanig wijzigen dat de regel `sysinit rc`, of welk bootscript dan ook zal worden gebruikt, wordt uitgevoerd. Als je er tevens zeker van wilt zijn dat gebruikers niet in kunnen loggen via seriële poorten, plaats dan een commentaarteken voor alle velden met `getty` waarin een `ttys` of `ttyS` device aan het einde van de regel is opgenomen. Laat de `tty` poorten staan, zodat je op de console in kunt loggen.

Een minimaal `inittab` bestand ziet er als volgt uit:

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

Het `inittab` bestand definieert wat het systeem in diverse toestanden uit zal voeren, waaronder bij het opstarten, het overgaan naar multi-user mode, enz. Controleer de bestanden vermeld in `inittab` zorgvuldig; als `init` het vermelde programma niet kan vinden, zal de bootdisk blijven hangen, en krijg je mogelijk zelfs geen foutmelding.

Een aantal programma's kan niet naar elders worden verplaatst, omdat andere programma's hun lokaties hebben ingeprogrammeerd. Op mijn systeem bijvoorbeeld is in `/etc/shutdown` `/etc/reboot` ingeprogrammeerd. Als ik `reboot` naar `/bin/reboot` verplaats, en dan de opdracht `shutdown` aanroep, zal de uitvoering ervan mislukken omdat het 't bestand `reboot` niet kan vinden.

Kopieer voor de rest alle tekstbestanden plus alle uitvoerbare bestanden in de directory `/etc` waarvan je niet zeker bent of je ze niet nodig hebt. Raadpleeg als een leidraad de voorbeeldlisting in Aanhangsel C. Waarschijnlijk is het voldoende alleen die bestanden te kopiëren, maar systemen verschillen nogal, dus je kunt 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 bij `inittab` en alles uit te werken wat nodig is.

Op de meeste systemen wordt nu gebruik gemaakt van een `/etc/rc.d/` directory waarin alle shell-scripts voor de verschillende runlevels staan. Het minimum is een enkel `rc` script, maar het kan eenvoudiger door gewoon het bestand `inittab` en de directory `/etc/rc.d` vanaf je bestaande systeem te kopiëren en de shell-scripts in de directory `rc.d` te ontdoen van verwerkingen die niet relevant zijn voor een systeemomgeving voor op diskette.

4.3.3. /bin en /sbin

De directory `/bin` is een prima plaats voor extra utility's die je nodig hebt voor de uitvoering van basisbewerkingen, utility's zoals ls, mv, cat en dd. Zie Aanhangsel C voor een voorbeeldlijst met bestanden die in de directory's bin en `/sbin` worden geplaatst. Hierin zijn geen utility's opgenomen die nodig zijn om gegevens vanaf een backup terug te zetten, zoals cpio, tar en gzip. Dat komt doordat ik die op een aparte utility-diskette plaats, om ruimte te besparen op de boot-/rootdiskette. Zodra de boot-/rootdiskette is geboot, wordt het naar de ramdisk gekopieerd waarbij het diskettestaion vrijkomt om een andere diskette, de utility-diskette te kunnen mounten. Ik mount deze gewoonlijk als `/usr`.

De aanmaak van een utility-diskette wordt hierna beschreven in de Paragraaf 8.3. Waarschijnlijk is het wenselijk een kopie van dezelfde versie backuputitily's, 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.

Verzeker je ervan dat je de volgende programma's opneemt: `init`, `getty` of equivalent, `login`, `mount`, een shell die capabel is voor het uitvoeren van je rc-scripts, een link vanuit `sh` naar je 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` worden gevonden dan zal het systeem niet kunnen booten. Als je geluk hebt, zie je wellicht een foutmelding over wat er aan de hand is.

Bijna ieder programma heeft op z'n minst de library `libc`, `libc.so.`N nodig; de N staat voor het huidige versienummer. Controleer je `/lib` directory. Het bestand `libc.so.N` is gewoonlijk 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 achter de andere library's te komen, neem je alle binaire bestanden door die je van plan bent op de diskette te plaatsen en controleer je daarvan de afhankelijkheden met 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 rechts is vereist. Het bestand mag een symbolische link zijn.

Een aantal library's is nogal groot en ze zullen niet zo gemakkelijk op je rootbestandssysteem passen. De hiervoor genoemde `libc.so` bijvoorbeeld is ongeveer 4 meg. Waarschijnlijk zal je library's moeten strippen wanneer je ze naar je rootbestandssysteem kopieert. Zie Paragraaf 8.1 voor instructies.

In `/lib` moet je tevens een loader voor de library's opnemen. De loader zal óf `ld.so` (voor A.OUT library's, die niet langer algemeen zijn) óf `ld-linux.so` (voor ELF library's) zijn. Nieuwere versies van ldd vertellen je exact welke loader nodig is, zoals in het voorbeeld hiervoor, maar oudere versies mogelijk niet. Als je niet zeker weet welke je nodig hebt, pas dan de opdracht file 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 rootbestandssysteem dat je aan het bouwen bent. Library's en loaders zouden zorgvuldig moeten worden gecontroleerd met de opgenomen library's. Als de kernel een benodigde library niet kan laden, kan het zijn dat de kernel zonder foutmelding blijft hangen.

4.4. Voorziening voor PAM en NSS

Mogelijk zijn er voor je systeem dynamisch laadbare library's nodig die niet zichtbaar zijn voor `ldd`. Als je hierin niet voorziet, kan het zijn dat je problemen krijgt bij het inloggen of het gebruiken van je bootdisk.

4.4.1. PAM (Pluggable Authentication Modules)

Als er op je systeem gebruik wordt gemaakt van PAM (Pluggable Authentication Modules), moet je er voorzieningen voor treffen op je bootdisk. Kort gezegd is PAM een geraffineerde modulaire methode voor de authenticatie van gebruikers en het beheren van de toegang tot de services voor gebruikers. Een eenvoudige manier om vast te stellen of je systeem gebruik maakt van PAM is `ldd` op het uitvoerbare bestand `login` toe te passen; als in de uitvoer `libpam.so` voorkomt, heb je PAM nodig.

Gelukkig heb je bij bootdisks gewoonlijk niets met beveiliging van doen aangezien iedereen die fysiek toegang heeft tot een computer, gewoonlijk alles kan doen wat hij/zij wil. Daarom kun je PAM effectief deactiveren door het aanmaken van een eenvoudig `/etc/pam.conf` bestand in je rootbestandssysteem 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 rootbestandssysteem. Deze library is slechts ongeveer 8K dus heeft het een minimale overhead tot gevolg.

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

Je moet tevens `/lib/libpam.so` op je bootdisk plaatsen. Maar dit wist je al 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 database lookups voor diverse services. Als je van plan bent services vanaf het netwerk te benaderen (bv, DNS of NIS lookups) dan moet je een eenvoudig `nsswitch.conf` bestand 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 wordt geleverd. Je zal ook `/lib/libnss_files.so.`X op moeten nemen; X is 1 voor glibc 2.0 en 2 voor glibc 2.1. Deze library zal dynamisch worden geladen om de file lookups af te handelen.

Als je van plan bent het netwerk vanaf je bootdisk te benaderen, wil je misschien een nauwgezetter bestand `nsswitch.conf` aanmaken. Zie de `nsswitch` manpage voor details. Je moet een bestand `/lib/libnss_` service`.so.1` opgeven voor iedere service die je specificeert.

4.5. Modules

Als je een modulaire kernel hebt, overweeg dan welke modules je na het booten vanaf je bootdisk wilt laden. Wellicht dat je ftape en zftape modules op wilt nemen als je backuptapes op floppytape zijn, modules voor SCSI-devices als je ze hebt, en mogelijk modules voor PPP of SLIP ondersteuning als je het net in noodgeval wilt benaderen.

Deze modules kunnen in `/lib/modules` worden geplaatst. Je zou ook insmod, rmmod en lsmod op moeten nemen. En afhankelijk van of je modules automatisch wilt laden, ook modprobe, depmod en swapout. Als je kerneld gebruikt, neem het dan samen met `/etc/conf.modules` op.

Het belangrijkste voordeel bij het gebruik van modules is echter dat je niet kritieke modules naar een utility-disk kunt verplaatsen en ze kunt laden wanneer ze nodig zijn. Dus gebruik je minder ruimte op je rootdisk. Als je met veel verschillende devices te maken hebt, heeft deze benadering de voorkeur in vergelijking met het bouwen van één zeer grote kernel met veel ingebouwde drivers.

Om een gecomprimeerd ext2 bestandssysteem te kunnen booten, moet je ramdisk en ext2 ondersteuning hebben ingebouwd. *Ze kunnen niet als modules worden toegevoegd.*

4.6. Een paar laatste details

Een aantal systeemprogramma's, zoals login, produceert foutmeldingen 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, pas je ldconfig toe op `/etc/ld.so.cache` op het rootbestandssysteem opnieuw aan te maken. De cache vertelt de loader waar het de library's vindt. Roep de volgende opdrachten aan voor het opnieuw maken van `ld.so.cache`:

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

De chroot is noodzakelijk omdat ldconfig de cache voor het rootbestandssysteem altijd opnieuw aanmaakt.

4.7. Het samenstellen

Zodra je klaar bent met het construeren van het rootbestandssysteem, unmount je het, kopieer je het naar een bestand en comprimeer je het:

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

Wanneer dit klaar is, heb je een bestand genaamd `rootfs.gz`. Dit is je gecomprimeerde rootbestandssysteem. Controleer de grootte ervan om er zeker van te zijn dat het op een diskette zal passen; als dit niet zo is, zal je terug moeten gaan en wat bestanden moeten verwijderen. In de Paragraaf 8.1 staat diverse aanbevelingen voor het terugdringen van de omvang van het rootbestandssysteem.

5. Uitkiezen van een kernel

Je hebt nu een compleet gecomprimeerd rootbestandssysteem. De volgende stap bestaat uit het samenstellen of uitkiezen van een kernel. In de meeste gevallen is het mogelijk je huidige kernel te kopiëren en daarmee vanaf diskette te booten. Er kunnen echter situaties zijn dat je een aparte kernel wenst te bouwen.

Één reden is de omvang. Als je een enkele boot-/rootdiskette aan het bouwen ben, 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 met een minimumset aan faciliteiten die nodig is om het gewenste systeem te ondersteunen. Dit betekent alles achterwege laten wat je niet nodig hebt. Netwerkondersteuning is prima achterwege te laten, als ook de ondersteuning voor eventuele diskettestations en andere drivers voor apparaten die je niet nodig hebt als je je boot-/rootsysteem draait. Zoals eerder uiteengezet, moet de ondersteuning voor de ramdisk en ext2 in je kernel zijn ingebouwd.

Je zal uit moeten werken wat erin 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-/rootdiskette zou zijn een systeem voor het bestuderen en herstellen van een beschadigd rootbestandssysteem, en hiervoor heb je wellicht kernelondersteuning 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 rootdrive en drives met Ftape kwijtraakt. Je zal Linux opnieuw moeten installeren, en ftape moeten downloaden en installeren om vervolgens je backups opnieuw in proberen te lezen.

Waar het hierom gaat is dat welke I/O ondersteuning je ook aan je kernel hebt toegevoegd voor de ondersteuning van backups, dit ook in je boot/root kernel moet worden toegevoegd.

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 eraan de kernel met “make zImage” te comprimeren.

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

Je hebt nu een kernel en een gecomprimeerd rootbestandssysteem. Controleer de grootte als je een boot-/rootdisk aan het maken bent om er zeker van te zijn dat ze beiden op één disk passen. Controleer het rootbestandssysteem 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 bootdiskkernel te booten. Het alternatief is de kernel direct naar de diskette te kopiëren en zonder LILO te booten. Het voordeel van het gebruik van LILO is dat het je de mogelijkheid biedt een aantal parameters aan de kernel op te geven die mogelijk nodig zijn om je hardware te initialiseren. (Controleer het bestand `/etc/lilo.conf` op je systeem. Als het bestaat en er een regel als “`append=...`”in voorkomt, heb je het 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 zal een klein apart bestandssysteem in moeten stellen, wat we het kernelbestandssysteem zullen noemen, waarnaar we de kernel en een paar andere bestanden die LILO nodig heeft, zullen transporteren.

Lees verder als je LILO gaat gebruiken en als je de kernel direct gaat transporteren dan ga je verder met de Paragraaf 6.2.

6.1. Transporteren van de kernel met LILO

Het eerste wat je moet doen is een klein configuratiebestand voor LILO aanmaken. Het zal 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
```

Zie de gebruikersdocumentatie van LILO voor een uitleg van deze parameters. Je zal waarschijnlijk ook de regel `append=...` aan dit bestand toe moeten voegen, kijk hiervoor in het bestand `/etc/lilo.conf` op je harddisk.

Bewaar dit bestand als `bdlilo.conf`.

Je zal nu een klein bestandssysteem aan moeten maken, wat we een kernelbestandssysteem zullen noemen, om het te onderscheiden van het rootbestandssysteem.

Zoek als eerste uit hoe groot het bestandssysteem zal moeten zijn. Neem de omvang van je kernel in blokken (de grootte weergegeven door “ls -l KERNEL” gedeeld door 1024 en afgerond naar boven) en tel hier 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 of gewoon 50 gebruiken. Als je een uit twee disks bestaande set gebruikt, kun je de ruimte net zo goed ruim nemen aangezien de kernel toch alleen voor de kernel wordt gebruikt. Noem dit aantal `KERNEL_BLOCKS`.

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

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

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

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

Maak dan de devices `/dev/null` en `/dev/fd0`. Je kunt in plaats van het opzoeken van de devicenummers, ze vanaf je harddisk kopiëren door gebruik te maken van `-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 samen met de kernel die je in de laatste sectie aanmaakte. Beiden kunnen in de rootdirectory worden geplaatst:

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

Alle benodigdheden voor LILO bevinden zich nu op het kernelbestandssysteem, 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 fouten moeten draaien, waarna het kernelbestandssysteem 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 diskette nu in het diskettestation en ga naar de Paragraaf 6.3.

6.2. Transporteren van de kernel zonder LILO

Transporteer de kernel met de opdracht dd als je LILO niet gebruikt:

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

In dit voorbeeld schreef dd 353 complete + 1 gedeeltelijk record weg, dus de kernel neemt de eerste 354 blokken van de diskette in beslag. Noem dit aantal `KERNEL_BLOCKS` en onthoud het voor gebruik in de volgende sectie.

Stel het rootdevice 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
```

Let erop dat je de hoofdletter `-R` gebruikt in de tweede rdev opdracht.

6.3. Instellen van het ramdisk word

Binnenin de kernelimage bevindt zich het ramdisk word waarin wordt aangegeven waar het rootbestandssysteem is te vinden, plus nog wat andere opties. Het word kan worden benaderd en ingesteld via de rdev opdracht, en de inhoud ervan wordt als volgt geïnterpreteerd:

Bit veld	Beschrijving
0-10	Offset van start ramdisk, in 1024 byte blokken
11-13	ongebruikt
14	Vlag die aangeeft dat ramdisk wordt geladen
15	Vlag die aangeeft een melding te geven alvorens rootfs te laden

Als bit 15 is ingesteld, zal tijdens de systeemstart worden aangegeven een nieuwe diskette in het diskettestation te plaatsen. Dit is nodig voor een uit twee disks bestaande bootset.

Er zijn twee situaties, afhankelijk van of je een enkele boot-/rootdiskette aan het bouwen bent, of een dubbele “boot+root” disketteset.

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

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

Stel na het zorgvuldig te hebben berekend van de waarde voor het ramdisk word het in met rdev -r. 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 je de naam van het diskettedevice (b.v., `/dev/fd0`).

```
rdev -r KERNEL_OR_FLOPPY_DRIVE  VALUE
```

Bij gebruik van LILO unmount je nu de diskette.

6.4. Transporteren van het rootbestandssysteem

De laatste stap bestaat uit het transporteren van het rootbestandssysteem.

- Als het rootbestandssysteem op dezelfde disk zal worden geplaatst als de kernel, dan transporteer je het 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
  ```

- Als het rootbestandssysteem op een tweede disk zal worden geplaatst, verwijder je de eerste diskette, doe je de tweede diskette in het diskettestation, en transporteert dan het rootbestandssysteem naar deze diskette:

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

Gefeliciteerd, je bent klaar!

Test een bootdisk altijd voordat je het opzij legt voor een noodgeval. 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 rootdisk is componenten vanuit je bestaande systeem te assembleren, en het op een diskette gebaseerd systeem te krijgen 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 dat deze 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 devicecodes; is het 't correcte rootdevice? Als dit niet zo is, heb je vermoedelijk rdev -R niet uitgevoerd, of paste je het toe op het verkeerde image. Als de devicecode correct is, controleer dan nauwkeurig de devicedrivers die in je kernel zijn gecompileerd. Overtuig jezelf ervan dat er ingebouwde ondersteuning voor een diskette, ramdisk en het ext2 bestandssysteem in de kernel voorkomt.

- Als je foutmeldingen ziet als:

  ```
  end_request: I/O error, dev 01:00 (ramdisk), sector NNN
  ```

  Dit is een I/O error die door de ramdiskdriver wordt gerapporteerd, waarschijnlijk omdat de kernel voorbij het einde van het device probeert te schrijven. Je ramdisk is te klein om je rootbestandssysteem te bevatten. Controleer de initialisatiemeldingen van je bootdiskkernel op een regel als:

  ```
          Ramdisk driver initialized : 16 ramdisks of 4096K size
  ```

  Vergelijk deze grootte met de ongecomprimeerde grootte van het rootbestandsssysteem. Maak de ramdisk groter als hij niet groot genoeg is.

- Controleer of de rootdisk echt die directory's bevat waarvan je denkt dat ze erop voorkomen. Het is heel eenvoudig op het onjuiste niveau iets te kopiëren, waardoor je op je rootdisk uiteindelijk uitkomt met iets als `/rootdisk/bin` in plaats van `/bin`.
- Controleer of er een `/lib/libc.so` voorkomt met dezelfde link die in je `/lib` directory op je harddisk staat.
- Controleer of alle symbolische links in je `/dev` directory van je bestaande systeem ook voorkomen op je rootbestandssysteem op de diskette, waar die links naar devices verwijzen die je op je rootdiskette hebt opgenomen. In het bijzonder zijn in veel gevallen de `/dev/console` links essentieel.
- Controleer of je de bestanden `/dev/tty1`, `/dev/null`, `/dev/zero`, `/dev/mem`, `/dev/ram` en `/dev/kmem` niet bent vergeten.
- Controleer je kernelconfiguratie -- ondersteuning voor alle bronnen die nodig zijn tot op het punt van inloggen mogen geen modules zijn, maar zijn ingebouwd. Dus ondersteuning voor een ramdisk en ext2 moeten zijn ingebouwd.
- Controleer of je kernel rootdevice en ramdisk instellingen correct zijn.

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

1. Zorg ervoor dat init is opgenomen als `/sbin/init` of `/bin/init`. Wees er zeker van dat het uitvoerbaar is.
2. Voer ldd init uit om te controleren op de library's van init. Gewoonlijk is dit slechts `libc.so`, maar controleer het toch maar. Zorg ervoor dat je de benodigde library's en loaders hebt ingesloten.
3. Verzeker je jezelf ervan dat je de juiste loader voor je library's hebt -- `ld.so` voor a.out of `ld-linux.so` voor ELF.
4. Controleer `/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 aard van het systeem. De enige manier om het aan te pakken is de manpages van init en `inittab` 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 opdracht moeten staan voor het uitvoeren van het systeem initialisatiescript, dat voor moet komen.

5. Pas net als bij getty ldd toe op getty om te zien wat het nodig heeft, en zorg ervoor dat de benodigde library bestanden en loaders in je rootbestandssysteem zijn opgenomen.
6. Wees er zeker van dat je een shell-programma hebt ingesloten (b.v., bash of ash) welke al je rc-scripts kan uitvoeren.
7. Als je een `/etc/ld.so.cache` bestand 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
```

is dat afkomstig van init, waarmee gewoonlijk wordt aangegeven dat getty of login afsluit zodra het opstart. Controleer de uitvoerbare bestanden getty en login en de library's waar ze afhankelijk van zijn. Zorg dat de aanroepen in `/etc/inittab` juist zijn. Als je vreemde meldingen van getty krijgt, kan dit betekenen dat de aanroepende vorm in `/etc/inittab` niet goed is.

Als je een loginprompt krijgt en je een geldige loginnaam invoert, maar het systeem vraagt je onmiddellijk daarna om nog een andere loginnaam, kan het probleem te maken hebben met PAM of NSS. Zie Paragraaf 4.4. Het probleem kan ook zijn 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 rootbestandssysteem

Soms is een rootbestandssysteem zelfs na compressie te groot voor op een diskette. Hier zijn een aantal manieren om de grootte van het bestandssysteem terug te brengen:

1. Verhoog de dichtheid van de diskette. Standaard worden diskette op 1440K geformatteerd, maar er zijn hogere dichtheidsformaten beschikbaar. `fdformat` kan disks met de volgende omvang formatteren: 1600, 1680, 1722, 1743, 1760, 1840, en 1920. De meeste 1440K diskettestations ondersteunen 1722K, en dit is wat ik altijd voor bootdisks gebruik. Zie de manpage van fdformat en `/usr/src/linux/Documentation/devices.txt`.
2. Vervang je shell. Een aantal populaire shells voor Linux, zoals bash en tcsh, is nogal groot en deze shells vereisen veel library's. Er zijn lichtgewicht alternatieven, zoals ash, lsh, kiss en smash, die heel wat kleiner zijn en waarvoor minder (of geen) library's nodig zijn. De meeste vervangende shells zijn beschikbaar vanaf `http://metalab.unc.edu/pub/Linux/system/shells/`. Zorg er in ieder geval voor dat de shell die je kiest de opdrachten in alle `rc` bestanden op je bootdisk uit kan voeren.
3. Strip library's en binary's. Veel library's en binary's worden met debugging informatie gedistribueerd. Als dit zo is krijg je als uitvoer "`not stripped`" als je op deze bestanden de opdracht file toepast. Bij het kopiëren van binary's naar je rootbestandssysteem, is het een goede gewoonte gebruik te maken van:

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

Gebruik bij het kopiëren van library's `strip-debug` in plaats van `strip-all`.

4. Als je bij het aanmaken van het rootbestandssysteem veel bestanden verplaatste of verwijderde, maak het dan opnieuw aan. Zie de NOOT HIERVOOR over het belang van het ontbreken van 'dirty blocks' in het bestandssysteem.
5. Verplaats niet kritieke bestanden naar een utilitydisk. Als een aantal van je binary's na het booten of inloggen niet onmiddellijk nodig is, kun je ze naar een utilitydisk verplaatsen. Zie de Paragraaf 8.3 voor details. Je kunt ook in overweging nemen modules naar een utilitydisk te verplaatsen.

8.2. Niet-ramdisk rootbestandssystemen

In Paragraaf 4 werden instructies gegeven voor het bouwen van een gecomprimeerd rootbestandssysteem die bij het booten van het systeem naar ramdisk wordt geladen. Deze methode heeft veel voordelen en wordt daarom vaak gebruikt. Op een aantal systemen kun je je dit echter niet permitteren vanwege de benodigde RAM, en moet het rootbestandssysteem direct vanaf de diskette worden gemount.

Dergelijke bestandssystemen zijn in wezen eenvoudiger aan te maken dan gecomprimeerde rootbestandssystemen, omdat ze op een diskette kunnen worden gebouwd in plaats van op één of ander ander device, en ze niet hoeven te worden gedecomprimeerd. We zullen deze procedure in zoverre ze 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 rootbestanden. Als je een enkele boot-/rootdisk aan het bouwen bent, moeten alle blokken voor de kernel plus alle blokken voor het rootbestandssysteem op één disk passen.
2. Maak met behulp van mke2fs een rootbestandssysteem op een diskette van de van toepassing zijnde grootte aan.
3. Stel het bestandssysteem samen zoals eerder werd beschreven.
4. Unmount het bestandssysteem en transporteer het naar een diskbestand als je klaar bent, maar comprimeer het niet.
5. Transporteer, zoals eerder beschreven, de kernel naar een diskette. Stel bit 14 in op nul bij het berekenen van het ramdisk word om aan te geven dat het rootbestandssysteem niet naar ramdisk moet worden geladen. Voer zoals eerder beschreven de opdracht rdev uit.
6. Transporteer het rootbestandssysteem als voorheen.

Er zijn verscheidene kortere wegen te bewandelen. Als je een uit twee disks bestaande set aan het bouwen bent, kun je het complete rootbestandssysteem direct op de tweede disk bouwen en is het niet nodig het tijdelijk naar een harddiskbestand te transporteren. Ook kun je een enkel bestandssysteem met de kernel, LILO bestanden en rootbestanden op de gehele disk bouwen als je een enkele boot-/rootdisk aan het bouwen bent en LILO gebruikt en als laatste stap gewoon LILO opstarten.

8.3. Bouwen van een utility-disk

Het bouwen van een utility-disk is relatief gezien eenvoudig -- maak op een geformatteerde disk een bestandssysteem aan en kopieer er de bestanden naar. Mount het handmatig nadat het systeem is geboot bij gebruik met een bootdisk.

In de instructies hiervoor gaven we al aan dat de utility-disk als `/usr` zou kunnen worden gemount. In deze situatie zouden de binary's in de `/bin` directory op je utility-disk kunnen worden geplaatst, zodat het in je path plaatsen van `/usr/bin` ervoor zorgt dat ze kunnen worden benadert. Extra library's benodigd voor de binary's worden geplaatst in `/lib` op de utility-disk.

Er zijn bij het ontwerpen van een utility-disk een aantal aandachtspunten:

1. Plaats kritieke systeembinary's of library's niet op de utility-disk omdat het pas nadat het systeem is geboot te mounten zal zijn.
2. Je kunt een diskette en floppy tapedrive niet gelijktijdig benaderen. Dit betekent dat als je een floppy tapedrive hebt, je het niet zal kunnen benaderen als je utility-disk is gemount.
3. Toegang tot de bestanden op de utility-disk verloopt traag.

In de Aanhangsel D wordt een voorbeeld gegeven van bestanden op een utility-disk. Hier zijn een aantal ideeën betreft bestanden die je wellicht nuttig zal vinden: programma's voor het bestuderen en manipuleren van disks (format, fdisk) en bestandssystemen (mke2fs, fsck, debugfs, isofs.o), een lichtgewicht teksteditor (elvis, jove), comprimeer- en archiefutility's (gzip, bzip, tar, cpio, afio), tape utility's (mt, ftmt, 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 het je is opgevallen dat de bootdisks van 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 omdat hun bootdisks aanvullende vereisten hebben. Ten eerste moeten ze kunnen werken met een brede variëteit aan hardware, dus moet er een interactie met de gebruiker plaats kunnen vinden en moet het mogelijk zijn diverse devicedrivers te laden. Ten tweede moeten ze zodanig zijn geprepareerd dat ze met vele verschillende installatie-opties werken, met diverse graden van automatisering. Als laatste worden in de bootdisks van distributies gewoonlijk installatie en rescue mogelijkheden gecombineerd.

Op een aantal bootdisks wordt gebruik gemaakt van een mogelijkheid genaamd initrd (initiële ramdisk). Deze mogelijkheid werd zo rond 2.0.x geïntroduceerd en deze maakt het mogelijk een kernel in twee fasen te booten. Wanneer de kernel in de eerste fase boot, laadt het een initiële ramdisk image vanaf de disk. Deze initiële ramdisk is een rootbestandssysteem met een programma dat voor het echte root-fs wordt geladen. Dit programma inspecteert gewoonlijk de omgeving en/of vraagt de gebruiker diverse bootopties, zoals het device waarvan de echte rootdisk te laden, te selecteren. Het laadt extra modules die niet in de kernel zijn gebouwd. Wanneer dit initiële programma stopt, laadt de kernel het echte root-image in en wordt het booten normaal gecontinueerd. Zie voor verdere informatie over initrd het lokale bestand `/usr/src/linux/Documentation/initrd.txt` (file:/usr/src/linux/Documentation/initrd.txt) en `ftp://elserv.ffm.fgan.de/pub/linux/loadlin-1.6/initrd-example.tgz`

Hieronder volgen samenvattingen van hoe de installatiedisks van iedere distributie schijnen te werken, gebaseerd op het inspecteren van de bestreffende bestandssysteem en/of sourcecode. We kunnen niet garanderen dat deze informatie volledig accuraat is, of dat ze sinds de vermelde versies niet is gewijzigd.

Slackware (v.3.1) gebruikt een recht-door-zee LILO-boot vergelijkbaar met wat is beschreven in Paragraaf 6.1. 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 rootbestandssysteem geladen vanaf een tweede disk. De gebruiker roept een setup script aan waarmee de installatie wordt gestart. Slackware voorziet in vele verschillende kernels in plaats dat het gebruik maakt van een modulaire kernel en het hangt van de gebruiker af die kernel te selecteren die overeenkomt met zijn of haar hardwarebenodigdheden.

Ook RedHat (v.4.0) maakt gebruik van een LILO boot. Het laadt een gecomprimeerde ramdisk vanaf de eerste disk, waarbij een aangepast init programma wordt uitgevoerd. Dit programma ondervraagt naar drivers en laadt dan zonodig de extra bestanden vanaf een supplemental disk.

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

10. Lijst met veel gestelde vragen (FAQ)

Vraag: Ik boot vanaf mijn boot-/rootdisks en er gebeurt niets. Wat kan ik doen?

Antwoord: Zie Paragraaf 7.

Vraag: Hoe werkt de Slackware/Debian/RedHat bootdisk?

Antwoord: Zie Paragraaf 9.

Vraag: Hoe kan ik een bootdisk met een XYZ-driver aanmaken?

Antwoord: De eenvoudigste manier is door aan een Slackware kernel vanaf je dichtsbijzijnde mirrorsite te komen. Slackware kernels zijn algemene kernels die voor zoveel mogelijk devices drivers op proberen 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, afhankelijk van het type controller dat je hebt, een IDE- of SCSI-kernel. Controleer het bestand xxxxkern.cfg voor de geselecteerde kernel om te bezien welke drivers in die kernel zijn opgenomen. Als het gewenste device in de lijst voorkomt, dan zou je met de corresponderende kernel je computer moeten kunnen booten. Download het bestand xxxxkern.tgz en kopieer het naar je bootdiskette zoals werd beschreven in de sectie over het maken van bootdisks.

Vervolgens moet je met behulp van de opdracht rdev zImage het rootdevice in de kernel controleren. Als deze niet hetzelfde is als het gewenste rootdevice, gebruik je rdev om het te wijzigen. De kernel die ik bijvoorbeeld probeerde was ingesteld op `/dev/sda2`, maar mijn root SCSI-partitie bevindt zich op `/dev/sda8`. Om het op een rootdiskette te gebruiken, zou je de opdracht rdev zImage /dev/fd0 uit moeten voeren.

Als je bovendien wilt weten hoe je een Slackware rootdisk in wilt stellen, dan raad ik je aan hiervoor de Linux Install Guide te lezen of aan de Slackware distributie te komen want dat valt buiten het kader van deze HOWTO. Zie in deze HOWTO de sectie getiteld “Referenties”.

Vraag: Hoe werk ik mijn rootdiskette bij met nieuwe bestanden?

Antwoord: De gemakkelijkste manier is het bestandssysteem vanaf de rootdisk terug naar het eerder gebruikte DEVICE te kopiëren (zie Paragraaf 4.2). Mount vervolgens het bestandssysteem en maak de wijzigingen. Je zal moeten onthouden waar je rootbestandssysteem 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 als voorheen verder (in Paragraaf 4.7) en transporteer je het rootbestandssysteem weer terug naar de disk. Als je de beginpositie van het nieuwe rootbestandssysteem niet wijzigt, hoef je de kernel niet opnieuw te transporteren of het ramdisk word opnieuw te berekenen.

Vraag: Hoe verwijder ik LILO zodat ik DOS weer kan gebruiken om te booten?

Antwoord: Dit is niet echt een Bootdisk onderwerp, maar het wordt vaak gevraagd. Onder Linux doe je het volgende:

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

Je kunt ook gebruik maken van de opdracht dd waarbij je de door LILO opgeslagen backup naar de bootsector kopieert. Raadpleeg hiervoor de LILO documentatie als je het op deze manier wilt doen.

Onder DOS en Windows kun je de volgende DOS-opdracht gebruiken:

```
FDISK /MBR
```

MBR staat voor Master Boot Record. Met deze opdracht vervang je de bootsector door een zuivere DOS MBR, zonder dat dit effect heeft op de partitietabel. Een aantal puristen is het hier niet mee eens, maar zelfs de auteur van LILO, Werner Almesberger, raadt dit aan. Het is makkelijk en het werkt.

Vraag: Hoe kan ik booten als ik mijn kernel- en mijn bootdisk niet meer heb?

Antwoord: Als je geen bootdisk meer bij de hand hebt, is de eenvoudigste methode vermoedelijk voor te zorgen dat je aan een Slackware kernel voor je type diskcontroller (IDE of SCSI) komt zoals hiervoor werd beschreven in “Hoe maak ik een bootdisk met een XXX driver?”. Je kunt je computer dan met behulp van deze kernel booten en vervolgens de opgelopen schade repareren.

Het kan zijn dat het rootdevice in deze kernel niet op het gewenste disktype en partitie is ingesteld. De algemene kernel van Slackware bijvoorbeeld heeft het rootdevice op `/dev/sda2` ingesteld, terwijl mijn Linux rootpartitie op `/dev/sda8` voorkomt. In dit geval zal het rootdevice in de kernel moeten worden gewijzigd.

Je kunt het root-device en de instellingen voor de ramdisk in de kernel nog steeds wijzigen zelfs al heb je alleen een kernel en een ander besturingssysteem zoals DOS.

rdev wijzigt de instellingen van de kernel door de waarden op vaste offsets in het kernelbestand aan te passen, dus je kunt hetzelfde doen als je een hex-editor tot je beschikking hebt. -- je kunt hierbij bijvoorbeeld gebruik maken van de Norton Utilities Disk Editor onder DOS. Je moet dan op de volgende offsets te 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 word werd hiervoor beschreven in Paragraaf 6.3.

De major en minor devicenummers moeten worden ingesteld op het device waarop je het rootbestandssysteem wilt instellen. Een aantal nuttige waarden om te selecteren zijn:

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

Zodra je deze waarden hebt ingesteld, kun je het bestand naar een diskette wegschrijven met behulp van de Norton Utilities Disk Editor of een programma genaamd rawrite.exe. Dit programma wordt met alle distributies meegeleverd. Het is een DOS-programma waarmee een bestand “raw” naar de disk wordt weggeschreven, te beginnen bij de bootsector, in plaats dat het bestand naar het bestandssysteem wordt weggeschreven. Als je Norton Utilities hiervoor gebruikt, moet je het bestand naar een fysieke disk beginnend bij de start van de disk wegschrijven.

Vraag: Hoe kan ik extra kopieën van boot-/rootdiskettes maken?

Antwoord: Omdat magnetische media mettertijd verslechtert, zou je verscheidene kopieën van je rescuedisk moeten bewaren, voor het geval het origineel onleesbaar wordt.

De eenvoudigste wijze om kopieën van een diskette te maken, waaronder opstartbare en utility-diskettes, is gebruik te maken van de opdracht dd om de inhoud van de oorspronkelijke diskette naar een bestand op je harddisk te kopiëren en dan dezelfde opdracht te gebruiken om het bestand terug naar een nieuwe 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 de opdracht:

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

waar DEVICENAME de naam van het device voor de diskette is en FILENAME de naam is van het uitvoerbestand (harddisk). Als je de parameter `count` achterwege laat, maakt dat je met dd de hele diskette kopieert (voor een high-density diskette zijn dit 2880 blokken).

Voor het kopiëren van het resulterende bestand naar een nieuwe diskette, doe je de nieuwe diskette in het diskettestation en geef je de opdracht omgekeerd:

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

In de uitleg hierboven wordt ervan uitgegaan dat je slechts één diskettestation hebt. Als je er twee van hetzelfde type hebt, kun je de diskettes kopiëren met een opdracht als:

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

Vraag: Hoe kan ik zonder het iedere keer weer typen van “ahaxxxx=nn,nn,nn” booten?

Antwoord: Als een diskdevice niet automatisch kan worden gedetecteerd moeten er aan de kernel met een opdracht device parameterstrings worden opgegeven, zoals:

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

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

- Door het iedere keer dat het systeem wordt geboot via LILO op de opdrachtregel in te voeren. Dit is echter nogal vervelend.
- Met behulp van LILO’s `lock` keyword om ervoor te zorgen dat de opdrachtregel als de standaard opdrachtregel wordt opgeslagen, zodat LILO iedere keer dat het boot dezelfde opties gebruikt.

- Met behulp van de opdracht `append=` in het configuratiebestand van LILO. De parameterstring moeten worden omsloten door aanhalingstekens.

Een voorbeeld van een opdrachtregel met de hiervoor genoemde parameterstring die zou worden gebruikt, zou zijn:

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

Hiermee zou de parameterstring voor het device worden doorgegeven en zou de kernel ook worden gevraagd het rootdevice op `/dev/sda1` in te stellen en de gehele opdrachtregel te bewaren en het voor alle toekomstige boots opnieuw te gebruiken.

Een voorbeeld van een APPEND opdracht is:

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

De parameterstring moet op de opdrachtregel niet door aanhalingstekens worden omsloten, maar wel in de opdracht `APPEND`.

In de kernel moet de driver waarop de parameterstring betrekking heeft, zijn ingebouwd. Als dit niet zo is, dan is er niets wat er naar de parameterstring luistert, en zal je de kernel opnieuw moeten bouwen zodat het benodigde device erin is opgenomen. Ga naar `/usr/src/linux` en lees de README, de Linux FAQ en Installatie HOWTO voor het opnieuw bouwen van de kernel. Als alternatief kun je een algemene kernel voor het type disk ophalen en die installeren.

We raden je aan de LILO documentatie goed door te lezen voordat je met de LILO installatie gaat experimenteren. Onvoorzichtig gebruik van de `BOOT` opdracht kan partities beschadigen.

Vraag: Tijdens de systeemstart, krijg ik de foutmelding “*`A: cannot execute B`*”. Waarom?

Antwoord: In een aantal situaties worden programmanamen in diverse utility’s ingeprogrammeerd (hardcoded). Dit is niet altijd het geval, maar het geeft wel een verklaring waarom een uitvoerbaar bestand blijkbaar niet op je systeem kan worden gevonden zelfs al kun je zien dat het er is. Of een gegeven programma de naam van een andere programma heeft ingeprogrammeerd kun je achterhalen met behulp van de opdracht strings en door via een pipe de uitvoer door grep te laten gaan.

Bekende voorbeelden van hardcoding zijn:

- In een aantal versies van shutdown is `/etc/reboot` hardcoded, dus moet reboot in de directory `/etc` worden geplaatst.
- init heeft op z’n minst voor één persoon voor problemen gezorgd waarbij de kernel init niet kon vinden.

Verplaats de programma’s óf naar de juiste directory, óf wijzig de configuratiebestanden (b.v. `inittab`) zodanig dat naar de juiste directory wordt verwezen. Plaats bij twijfel de programma’s in dezelfde directory als waar ze op je harddisk staan, en gebruik dezelfde `inittab` en `/etc/rc.d` bestanden zoals die op je harddisk.

Vraag: Mijn kernel heeft ondersteuning voor een ramdisk van 0K. Waarom?

Antwoord: In die gevallen zal bij het booten een kernelmelding als de volgende worden weergegeven:

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

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

```
ramdisk= 0
```

Dit stond in een aantal oudere distributies in voorbeeldconfiguratiebestanden van LILO, en het werd hier geplaatst om eventuele voorgaande kernelinstellingen te overschrijven. Als er een dergelijke regel in voorkomt, verwijder je het.

Als je een ramdisk ter grootte van 0 probeert te gebruiken, kan de werking onvoorspelbaar zijn en in kernelpanics resulteren.

A. Bronnen en verwijzingen

Zorg bij het ophalen van een package altijd dat je de laatste versie ophaalt, tenzij je goede redenen hebt om dit niet te doen.

A.1. Voorgefabriceerde Bootdisks

Dit zijn bronnen voor distributie-bootdisks. Gebruik alsjeblieft één van de mirror-sites om de load op deze machines te beperken.

- Slackware bootdisks (http://metalab.unc.edu/pub/Linux/distributions/slackware/bootdsks.144/), rootdisks (http://metalab.unc 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 aangegeven, zijn ze te vinden in de directory `http://metalab.unc.edu/pub/Linux/system/recovery/!INDEX.html`

- `tomsrtbt`, door Tom Oehser, is een enkele boot-/rootdisk gebaseerd op kernel 2.0, met een grote set mogelijkheden en ondersteunings programma's. Het biedt ondersteuning voor IDE, SCSI, tape, netwerkadaptors, PCMCIA en meer. Ongeveer 100 utility-programma's en tools zijn opgenomen voor het herstellen van disks. In het package zijn ook scripts opgenomen voor het deassembleren 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, Adaptec 1542 en NCR53C7,8xx. Het maakt gebruik van ELF binary's, maar heeft genoeg opdrachten zodat het op ieder systeem gebruikt kan worden. Er zijn voor alle andere SCSI-kaarten modules die na het booten kunnen worden geladen. Het werkt waarschijnlijk niet met systemen met 4 mb aan ram aangezien het gebruik maakt van een ramdisk van 3 mb.

- `resque_disk-2.0.22`, door Sergei Viznyuk, is een boot-/rootdisk gebaseerd op kernel 2.0.22 met ingebouwde ondersteuning voor IDE, veel verschillende SCSI-controllers, en ELF/AOUT. Tevens zijn veel modules en nuttige utility's voor het herstellen van een harddisk opgenomen.
- `cramdisk` images (http://metalab.unc.edu/pub/Linux/system/recovery/images), gebaseerd op de 2.0.23 kernel, beschikbaar voor 4 meg en 8 meg machines. Hierin is ondersteuning voor de math emulatie en netwerken (PPP en dial-in script, NE2000, 3C509) opgenomen of ondersteuning voor de parallelle poort ZIP-drive. Deze disk-images zullen op een 386'r met 4MB RAM booten. MSDOS ondersteuning is opgenomen dus je kunt vanaf het net naar een DOS-partitie downloaden.

A.2. Rescue packages

Op metalab.unc.edu zijn verscheidene packages voor het aanmaken van rescue-disks beschikbaar. Met deze packages specificeer je een set bestanden 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 packages is verscheidene jaren niet bijgewerkt en zal de aanmaak van een gecomprimeerd rootbestandssysteem die in de ramdisk wordt geladen niet ondersteunen. Zover we weten, is Yard (http://www.croftj.net/~fawcett/yard/index.html) het enige package dat dit wel doet.

A.3. 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 op Metalab en mirrors.

A.4. Linux FAQ en HOWTO's

Deze zijn vanaf veel bronnen beschikbaar. Kijk in 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 van http://linuxdoc.org/HOWTO/HOWT INDEX.html. De meeste documentatie voor Linux is te vinden op de homepage van het Linux Documentatie Project (http://linuxdoc.org/).

A.5. Ramdisk gebruik

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

A.6. Het Linux bootproces

Hier zijn wat verwijzingen voor meer info over het Linux bootproces:

- In de Linux System Administrators' Guide (http://linuxdoc.org/LDP/sag/c1596.html) staat een sectie over het booten.

- In het LILO “Technische overzicht” (http://metalab.unc.edu/pub/Linux/system/boot/lilo/lilo-t-21.ps.gz) staat de definitieve technische, low-level beschrijving van het bootproces, tot aan waar de kernel is gestart.
- De broncode is de definitieve leidraad. Hieronder staan een aantal kernelbestanden gerelateerd aan het bootproces. Als je de broncode van de Linux-kernel hebt, kun je deze op je computer vinden onder `/usr/src/linux`; als alternatief heeft Shigio Yamaguchi (shigio@tamacom.com) voor het lezen van de kernelbronbestanden een zeer fraaie hypertext kernel browser (http://www.tamacom.com/tour/linux/index.html). Dit zijn een aantal relevante te bekijken bestanden:

  `arch/i386/boot/bootsect.S` en `setup.S`

  > Hierin staat assembleercode voor de bootsector zelf.

  `arch/i386/boot/compressed/misc.c`

  > Hierin staat code voor het decomprimeren van de kernel.

  `arch/i386/kernel/`

  > Directory met kernel-initialisatiecode. `setup.c` definieert 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 ramdisk. De procedure identify_ramdisk_image stelt vast welk type bestandssysteem is gevonden en of het is gecomprimeerd.

B. LILO boot foutcodes

Vragen over deze foutmeldingen worden zovaak in Usenet gesteld dat we ze hier als een publieke service hebben opgenomen. Deze samenvatting is onttrokken uit Werner Almsberger’s LILO User Documentation (http://metalab.unc.edu/pub/Linu
u-21.ps.gz).

Wanneer LILO zichzelf laadt, geeft het ’t woord `LILO` weer. Iedere letter wordt voor of na het uitvoeren van een bepaalde actie afgedrukt. Als LILO op een bepaald punt faalt, worden de letters tot zover afgedrukt dat ze kunnen worden gebruikt om het probleem te identificeren.

Uitvoer	**Probleem**
(niets)	Geen enkel onderdeel van LILO werd geladen. LILO is óf niet geïnstalleerd óf de partitie waarop de bootsector voorkomt, is niet actief.
L	De eerste fase bootloader werd geladen, maar het kan de tweede fase bootloader niet laden. De uit twee cijfers bestaande foutcode geeft het type probleem aan. (Zie ook de sectie “Disk foutcodes”.) Dit geeft meestal aan dat er een media storing is óf een onjuiste geometrie (b.v. verkeerde diskparameters).
LI	De eerste fase bootloader kon de tweede fase bootloader laden, maar lukte het niet het uit te voeren. Dit kan óf worden veroorzaakt door een onjuiste geometrie óf door het verplaatsen van `/boot/boot.b` zonder dat de map-installer werd uitgevoerd.
LIL	De tweede fase bootloader is gestart, maar het kan de descriptor tabel vanuit het map-bestand niet laden. Dit wordt meestal veroorzaakt door een storing aan media of door een onjuiste geometrie.

Uitvoer	**Probleem**
LIL?	De tweede fase bootloader is op een onjuist adres geladen. Dit wordt meestal door een subtiel onjuiste geometrie veroorzaakt óf doordat de `/boot/boot.b` werd verplaatst zonder dat de map-installer werd uitgevoerd.
LIL-	De descriptor tabel is beschadigd. Dit kan óf worden veroorzaakt door een onjuiste geometrie of 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 bootimage probeert te laden, wordt de bijbehorende foutcode weergegeven. Deze codes variëren van `0x00` tot en met `0xbb`. Zie de LILO Gebruikersgids voor een uitleg van deze foutcodes.

C. Voorbeeldlisting van een rootbestandssysteem

```
/:
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 -> ttyS1
lrwxrwxrwx   1 root     root           12 Nov  1 15:39 mouse -> 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 slechts die devices voor de IDE-partities opgenomen, waar
***  ik gebruik van maak. Als je gebruik maakt van SCSI, gebruik dan
***  in plaats daarvan 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
-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:
-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:
drwx------   1 root     root         7156 Nov  1 15:39 loop.o

/lib/modules/2.0.35/cdrom:
drwx------   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 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
-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 utilitydisk

```
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
```

Noten

1. De directorystructuur die hier wordt gepresenteerd is alleen voor het gebruik van een rootdiskette. Echte Linux-systemen hebben een complexere en meer gedisciplineerde set gedragslijnen, genaamd de Filesystem Hierarchy Standard (http://www.pathname.com/fhs/2.0/fhs-toc.html), voor het vaststellen waar welke bestanden in staan).