NAME

installboot − make a device bootable

SYNOPSIS

installboot −i(mage) *image* [*label*:]*kernel mm fs ... init*
installboot −(e)x(tract) *image*
installboot −d(evice) *device bootblock boot* [[*label*:]*image* ...]
installboot −b(oot) *device bootblock boot* [*label*:]*image* ...
installboot −m(aster) [*fix*] *device masterboot*

DESCRIPTION

Installboot may be used to make a device bootable by constructing a kernel image and installing bootstrap code into the boot block of a Minix file system. To understand how this can be done one first has to know what happens when a PC is booted.

When the power is turned on the typical PC will try to read the first sector from the first floppy disk or from the first hard disk into memory and execute it. The code obtained from the hard disk (from the so-called master boot sector) will immediately replace itself by the code found in the first sector of the active partition. Thus the PC is now executing the bootstrap code found in the first sector of /dev/fd0, /dev/hd1, /dev/hd2, /dev/hd3, or /dev/hd4. The bootstrap will locate the operating system on the device it itself was loaded from, load it, and execute it.

To make a Minix file system **/dev/fd0** mounted on **/mnt** bootable, enter the following:

```
cp /usr/mdec/boot /mnt/boot

installboot −i /mnt/minix kernel mm fs init

installboot −d /dev/fd0 /usr/mdec/bootblock boot
```

The "boot" program in the example is named the "boot monitor". It is loaded by the bootblock code placed in the boot sector of /dev/fd0 and it will take care of loading the kernel image "minix" from the root directory of the file system. See **monitor**(8) for a description of the boot monitor. Note that **boot** is a name in the file system on **/dev/fd0** in this example, the same file as **/mnt/boot**. Making **/mnt/minix** is normally not necessary, there is usually a kernel image in the **tools** directory.

OPTIONS

−i(mage) *image* [*label*:]*kernel mm fs ... init*

The **−image** option (or the **−i** shorthand) combines the executable files needed to run Minix in one file. Only the names and a few zero bytes are inserted into the image. The name is for identification and the zeros are used to pad separate pieces to sector boundaries for fast loading.

An executable may be prefixed by a label. The monitor may be instructed to load processes by label. So more than one kernel process may be included in the image, each with a different winchester driver for instance. So if you have compiled two different kernels with an AT or XT driver then

```
installboot −i image AT:at_kernel XT:xt_kernel mm fs init
```

will make an image with two different labeled kernels and one unlabeled set of the other binaries.

−(e)x(tract) *image*

Extract the binaries from *image* under the names stored in the image. (The name includes the optional label.)

−d(evice) *device bootblock boot* [[*label*:]*image* ...]

Installs *bootblock* in the boot sector of *device* together with the disk addresses to *boot*. These disk addresses are needed to load *boot* from the file system at boot time. The argument *boot* is first searched in the file system on *device*. If it is not found then it is read as a normal file and added at the end of the file system. The file system should be smaller than the device it is on to allow this. Any extra images are also added to the end as described under **−boot**. (Make sure you understand all this.)

The device need not be mounted when **installboot** is run, nor does it matter if it is.

Installboot needs to be run again if *boot* is rewritten, because it will then occupy a new place on the disk.

Old boot parameters are kept if there are no images added.

−b(oot) *device bootblock boot* [*label*:]*image ...*

This option fills a blank floppy in *device* with boot code and kernel images. This "boot disk" does not have a root file system, only the boot monitor and Minix kernels. The boot parameters sector is filled with code that enables menu options for selecting an image. After loading an image, the monitor will ask you to insert a root file system diskette before starting Minix.

The labels used on the images should match those on the executables used inside the image. You can put a comma separated list of labels on an image for each label used within the image. For the image created earlier one would create a boot floppy like this:

> **installboot −b /dev/fd0 bootblock boot** *AT,XT:image*

If a label-list is omitted on an image, then that image will be selected by default. (Like in the normal one image, no labels case.)

Note that **−device** and **−boot** together allow you to make a boot floppy with or without a root file system. With the boot code in the file system, attached to the end of it, or after the boot block. And with one or more kernel images in the file system or at the end of the device. Somewhat confusing.

−m(aster) [*fix*] *device masterboot*

This option installs the *masterboot* program into the boot sector of the given device. If another device is given instead of *masterboot* then its bootstrap code is copied to *device*. The master bootstrap on a hard disk boots the active partition on that disk at boot time. The MS-DOS fdisk command normally puts a master bootstrap on the hard disk. Minix has two bootstraps that can be used as a master bootstrap. A fairly normal one named **masterboot** that works as follows:

> If the ALT key is held down while booting then '/dev/hd?' appears and you are expected to type a number key (0 − 9) to select the device to boot.
>
> If *fix* (a small number) is given then the bootstrap is locked into booting the **/dev/hd***fix* disk or primary partition. This is needed if 'boot *hd*N*' is used from the monitor to boot an O.S. that needs the active flag set.
>
> If installed on a Minix floppy then it will try to boot the next floppy or the first hard disk. Ideal for floppies with just data on it, they will no longer obstruct the boot process if left in the drive. Also a very useful trick to boot from floppy drive 1.
>
> If installed on a hard disk then the active partition is selected and booted as usual, unless none of the partitions is marked active, then it will boot the next disk. The latter is useful if you want to boot an operating system from the second disk by default.

The second bootstrap is named **extboot**. It has only one function, to boot the logical partition named by *fix*. *Fix* is not optional for **extboot** and must be a number-letter pair, like **2c** for **/dev/hd2c**.

Extboot or **masterboot** with a fix key need not be installed in the hard disk master bootstrap per se if you don't want to mess with the DOS master bootstrap, or if you want keep the active flag functioning. An extended partition or a non-root Minix partition are better candidates. It seems logical to put **extboot** in the extended partition boot block.

A backup copy of the current master bootstrap (including the partition table) can be made with:

> dd if=*device* of=*backup-file* count=1

A simple 'cat *backup-file* > *device*' will put it back. You can also use **fdisk /mbr** under MS-

DOS 5.0 (or newer) to restore the master bootstrap.

FILES

/usr/mdec/bootblock	Minix bootstrap for the Minix root device. To be placed in the boot sector.
/usr/mdec/boot	Minix Boot Monitor. Can usually be found in the root directory of a bootable device.
/usr/mdec/masterboot	Master bootstrap. Can be placed in the first sector of a disk to select the active partition. In a Minix primary partition it selects the active subpartition.
/usr/mdec/extboot	Extended partition bootstrap.

SEE ALSO

part(8), **monitor**(8).

DIAGNOSTICS

Boot doesn’t fit on *device*

If there is no space on the device to add the boot code. This usually means that there is no boot code in the file system you use **installboot −device** on.

Image doesn’t fit on *device*

If the device is too small for all the images you try to put on it.

BUGS

It has four more options than the SunOS installboot program it is modeled after.

The bootblock code has been crunched to such ugliness that you can use it to scare little kids out of your garden.

AUTHOR

Kees J. Bot (kjb@cs.vu.nl)