FAQ

DITTO: A USEFUL NEW UPGRADING TOOL

NEXTSTEP 3.3 introduced several new tools to make upgrades a bit easier. One not-too-obvious but quite useful utility introduced in 3.3 is called **ditto**. **ditto** is a command-line utility for copying directories in special ways. If you are familiar with the NEXTSTEP application BuildDisk and found it useful for making bootable system disks, then **ditto** is something to check out. **ditto** serves several purposes. You can use it to make exact duplicates of directories and file systems, ªthin outº fat binaries, and create a duplicate of a package's contents based on its bill of materials (or BOM).

The latter function is particularly interesting because you can specify which bill of materials for particular files get copied. For example, you can specify **/usr/lib/NextStep/BaseSystem.bom**, the bill of materials of the base NEXTSTEP system. This means that given a formatted hard drive with a boot block, you can use **ditto** to create a bootable system disk. Because **ditto** copies all information about the specified file system, you must be **root** to run the command. The only other requirement for **ditto**'s use in this way is that it must have a writable file system for **/tmp**.

BuildDisk isn't supported under NEXTSTEP for Intel processors because of its inability to deal with drivers.

Duplicating entire directories with ditto

The syntax for using **ditto** to duplicate a directory is simple:

ditto *source_directory destination_directory*

For example, the following command makes an exact copy of **/etc** in the directory **/etc_backup**.

ditto /etc /etc_backup

Stripping out unwanted binaries

You can also use ditto to make a copy of the source directory, but also ªthinº any files to the Intel architecture. That is, ditto can remove binaries for other architectures, like NeXT computers or RISC workstations.

ditto -archi386 source_directory destination_directory

This is useful if, for example, you have a set of binaries that have been compiled for several architectures, but you need to use them for only one particular architecture. Current valid architecture types you can specify include **m68k**, **i386**, **hppa**, and **sparc**.

Duplicating from a bill of materials

The syntax for duplicating from a BOM is also straightforward.

ditto -bom /*path*/*BomFile*.bom *source_directory destination_directory*

For example:

    ditto -bom /usr/lib/NextStep/BaseSystem.bom / /Second_Disk

This copies all of the system files specified in the **BaseSystem.bom** to a second hard drive mounted at **/Second_Disk**. You would use this command to duplicate a system disk or installed packages. **BaseSystem.bom** is the bill of materials used for all of the NEXTSTEP system files installed on the current disk. If you want to copy additional packages, specify which BOM to use as the source. BOMs are typically found within package wrappers. (In Workspace Manager, you can use the Open as Folder command or type Command-O to see the contents of a package.)

Note that to make a disk bootable you first need to partition, initialize, and make a file system on the new disk. See ªCreating a bootable NEXTSTEP disk from an existing NEXTSTEP diskº to find out how to do that.

To check the contents of a bill of materials, use the **lsbom** command. Just run **lsbom** *bomfile* to see what's in the BOM.

Refer to the UNIX manual page for **ditto** *for more information.*

Creating a bootable NEXTSTEP disk from an existing NEXTSTEP disk

This section details building a bootable NEXTSTEP hard disk basically from scratch (actually from an already installed system). If you're ever stuck needing to create a bootable disk by hand and cannot install from either the CD or netinstall, but you *do* already have a bootable system, this is the way out of your dilemma. While this isn't a complicated procedure, you should be pretty comfortable with UNIX before you attempt this.

1 Connect the new disk to the computer and boot up using the existing NEXTSTEP disk.

2 Once the system is up, log in as **root**.

3 You may be prompted to initialize the disk if it has not already been initialized. Choose Ignore and start up the Terminal application. (If you boot in single-user mode, you're already   effectively in Terminal.)

4 Use the **umount** command to make sure the disk is unmounted. Below are common valid devices used with **umount**.

   umount /dev/(s,h)d(1-6)a

   For example, if the disk you want to initialize is the second SCSI disk, use this command

   umount /dev/sd1a

   Or, for a second IDE disk, use:

   umount /dev/hd1a

5 Now that the disk is unmounted, you can reinitialize it. If you are working with an Intel-based computer, use the **fdisk** command to partition the disk into whatever partitioning scheme you need. Skip this step if you are using a NeXT Computer or RISC workstation.

6 Use the **disk** command to initialize the disk and add a boot block so the disk will be bootable. The following command is an example:

   disk -i -l "label" -b /dev/rsd1a

   This command initializes the disk (**-i**), writes a label on it (**-l** *label*), and puts the boot block **/usr/standalone/boot** onto the disk.

7 You can now remount the disk and use **ditto** to copy over system files.To remount the disk, choose Check for Disks in the Disk menu for the Workspace Manager or use the **mount** command. To use **mount**, type the following to mount the second SCSI disk as **/Disk** in the file system:

```
mount /dev/sd1a /Disk
```

Or, if you are using the second IDE disk, use this command instead:

```
mount /dev/hd1a /Disk
```

8 Once the disk is mounted, use the **ditto** command to create a duplicate of the root hard drive. Here's an example:

```
ditto -V -bom /usr/lib/NextStep/BaseSystem.bom / /Disk
```

This causes **ditto** to use the bill of materials from the original system to copy the system **(/)** to **/Disk**. The result is an image of the installed system created on **/Disk**. The option **-V** tells **ditto** to use verbose mode so you can see what it's doing.

9 One last step: A link needs to be created for the device drivers on the system, so the system knows which architecture drivers to use. From a Terminal window, run this command:

```
cd /Disk/private
ln -s Drivers/arch Devices
```

arch must be either **m68k**, **i386**, **hppa**, or **sparc**, depending on the architecture of the system disk you are creating.

The disk should now be usable as a NEXTSTEP boot disk. You can set it up as a boot disk (changing SCSI IDs or any pertinent jumper settings) and use it to boot into NEXTSTEP.

To find out more, see the **disk**, **umount**, **lsbom**, **mount**, *and* **ditto** *UNIX manual pages provided with NEXTSTEP.*

Thanks to Brandon Greimann for this useful tip!

_____

http://www.next.com/HotNews/Journal/OSJ/SpringContents95.html