How to transfer a Linux system from one disk to another.

Robert Heller
Deepwoods Software
Wendell, MA, USA *

Jan 24, 2009

Abstract

This document covers the step-by-step process of transferring a Linux system from one disk to another. This process is a simple, straight forward, and foolproof way to upgrade your hard drive or simply rescue a system with an old or failing hard drive. It can also be used to clone a system disk or to simply make a snapshot backup of a working system.

Contents

1 The tools we will use

The tools we will be using consist of pieces of software that are part of the base install of just about every Linux distribution. These tools are fairly simple to use, but are potentially dangerous. These tools are:

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/hda1	966M	569M	348M	63%	/
/dev/hda2	471M	36M	411M	9%	/boot
/dev/hda3	3.8G	2.4G	1.2G	67%	/usr
none	252M	0	252M	0%	/dev/shm
/dev/hda5	3.8G	2.9G	727M	81%	/home
/dev/hda7	16G	9.6G	5.5G	64%	/scratch

Table 1: Output of the df command

fdisk Fdisk is the basic command line / console program for editing the partition table of a mass storage device (such as a hard drive).

mkfs.ext2/3 Mkfs.ext2 and Mkfs.ext3 are programs for creating a fresh Linux file system, either ext2 or ext3.

dump / **restore** Dump is a basic file system backup program. Restore is the reverse of dump.

We will also be using a few other commands, including mkdir, mount, and df.

2 Our disks

In this article we will be copying from one IDE hard drive (/dev/hda) to another IDE hard drive (/dev/hdb). This will be a fictitious system[1] with the partitions shown in Table 1. All of the file systems, except the /boot file system are ext3. The /boot file system is ext2 (no journalizing). Partition 4 is an extended partition and partition 6 is a 1gig swap partition.

After noting down the sizes of the partitions and paying attention to the Used amounts, which imply a minimum size for these partitions on the destination disk, we will shut down the computer, install the new disk as the primary slave (/dev/hdb). Then we will bring the system up into single user mode. We will then be at a shell prompt of a mostly “quiet” system (no major daemons running, no networking, etc.).

3 Step 1: Partitioning the new disk

The first step is to partition the new disk. This will be done with fdisk. We will assume we are working with a new disk that has no partition table on it[2]. We will proceed like this:

[1] One that is based on my laptop, running CentOS 4.7.

[2] A “used” disk can also be used, in which case the fdisk “d” command would be used to delete all existing partitions.

- Enter fdisk's command mode for the new disk by typing the fdisk command at the shell prompt:

  sh$ fdisk /dev/hdb

- Now create the first partition, using fdisk's **n** command. This will be a primary partition and its number will be 1. It will start with cylinder 1 (the default for a disk with no partitions on it). For an end cylinder, enter "1024m". This creates a partition of about 1 gigabyte. Fdisk will pick an end cylinder that results in a partition of about 1 gigabyte.

- Now create the second partition, again using fdisk's **n** command. This will be a primary partition and its number will be 2. Use the default starting cylinder and specify "512m" as the end cylinder. This creates a partition of about 1/2 gigabyte.

- Now create the third partition, again using fdisk's **n** command. This will be a primary partition and its number will be 3. Use the default starting cylinder and specify "4096m" as the end cylinder. This creates a partition of about 4 gigabytes.

- Now create the fourth partition, again using fdisk's **n** command. This will be a extended partition and its number will be 4. Use the defaults for both the starting and ending cylinders. This will make partition 4 use the whole rest of the disk. All of the rest of the partitions will be logical ones carved out of this partition.

- Now create the fifth partition, again using fdisk's **n** command. This will be a logical partition and its number will be 5[3]. Use the default starting cylinder and specify an ending cylinder of "4096m". This creates a 4 gigabyte partition.

- Now create the sixth partition, again using fdisk's **n** command. This will be a logical partition and its number will be 6. Use the default starting cylinder and specify an ending cylinder of "1024m". This creates a 1 gigabyte partition.

- Now change the sixth partition to be a swap partition using the **t** command. Select partition 6 and a partition type of 82.

- Finally, create the last partition, number 7. Use the **n** command and use the default start and end cylinders. This allocates the rest of the disk to partition 7.

- Now write the partition table and exit using the **w** command. Fdisk will write the partition table out and cause the kernel to reread the disk's partition table.

[3] Fdisk won't bother asking you for a partition number at this point.

4 Step 2: Making a fresh set of file systems.

Now we will make the file systems on the new disk.

We will use mkfs.ext2 and mkfs.ext3 to create the file systems on the new disk.

- First the new root file system:

  sh$ mkfs.ext3 -L/ /dev/hdb1

- Then the /boot file system:

  sh$ mkfs.ext2 -L/boot /dev/hdb2

- Then the /usr file system:

  sh$ mkfs.ext3 -L/usr /dev/hdb3

- Then the /home file system:

  sh$ mkfs.ext3 -L/home /dev/hdb5

- Then the /scratch file system:

  sh$ mkfs.ext3 -L/scratch /dev/hdb7

- Finally the swap partition:

  sh$ mkswap /dev/hdb6

5 Step 3: Now we can copy files!

We will use dump and restore to copy the files. I like dump and restore for this since we are using ext2/ext3 file systems and because these programs are fast and fully understand the underlying file systems and will deal properly with “special” files, including symbolic and hard links, block and char special files, and things like pipe and socket files correctly.

The first thing we need to do is mount the new root file system:

```
sh$ mkdir /mnt/newdisk
sh$ mount -v /dev/hdb1 /mnt/newdisk
```

Now we can dump the existing root file system:

```
sh$ dump 0f - / — (cd /mnt/newdisk;restore -rf -)
```

When this completes, we can mount the new /boot file system and copy /boot to the new disk:

```
sh$ mount -v /dev/hdb2 /mnt/newdisk/boot
sh$ dump 0f - /boot — (cd /mnt/newdisk/boot;restore -rf -)
```

We can then proceed to repeat this process for the rest of the file systems:

```
sh$ mount -v /dev/hdb3 /mnt/newdisk/usr
sh$ dump 0f - /usr — (cd /mnt/newdisk/usr;restore -rf -)
sh$ mount -v /dev/hdb5 /mnt/newdisk/home
sh$ dump 0f - /home — (cd /mnt/newdisk/home;restore -rf -)
sh$ mount -v /dev/hdb7 /mnt/newdisk/scratch
sh$ dump 0f - /scratch — (cd /mnt/newdisk/scratch;restore -rf -)
```

We check things at this point using the df command. The old and new file systems should have about the same “Used” sizes[4]. If all is good, we can proceed to the next step.

6 Step 3: Installing the new disk and restoring the boot loader.

We can now do a /sbin/shutdown -h now. Once the machine is down, we can make the necessary hardware changes to make the new file the primary master. Once this is done, we can boot up with a rescue disk and from there, install the boot loader.

[4] Small differences may occur due to slightly different disk sizes and/or geometries.