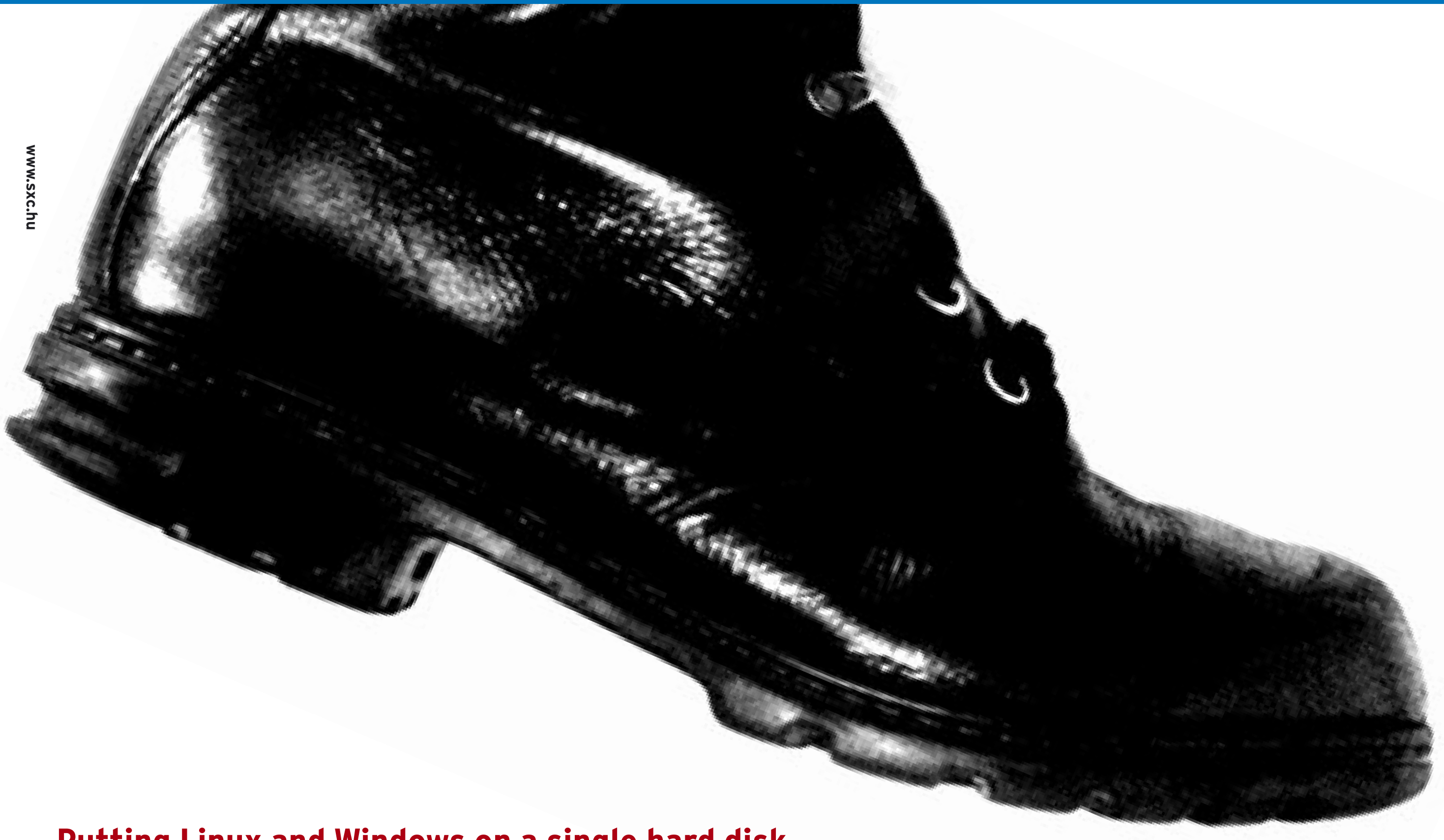


Putting Linux and Windows on a single hard disk

BOOT POLISHING

When two systems share a single computer, a boot manager handles the prompts that determine which system to boot. We'll show you several multiple boot scenarios and describe how to set up your system for dual booting Linux with Windows. **BY ANDREA MÜLLER**

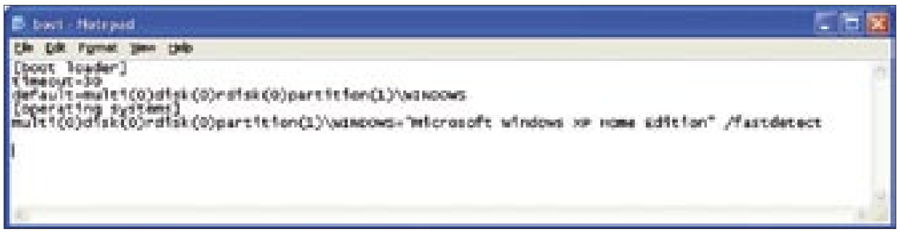
If you need applications such as Dreamweaver, Microsoft Office, or some modern games, and you don't like the idea of using Wine or VMware, you may just have to resort to using a Windows system. And if you run your Windows system along with Linux on a dual boot PC, you need to set up your boot manager to choose which operating system to boot. Luckily, there is no need to purchase commercial software, as more or less any Linux distribution will give you the tools you need for a sucessful dual boot configuration.

The Lilo boot manager, which used to be the principal Linux boot management system, has almost completely surrendered its boot monopoly to Grub (Grand Unified Bootloader) [1]. The Gnu/HURD boot manager has many more features than the simpler Lilo manager. Besides filesystem support, Grub also has an integrated command line, which allow users to boot the installed operating systems even if the menu file is faulty or has been accidentally deleted.

As You Like It

If you are looking to install Linux and Windows on a single computer, you'll find that Linux is the less demanding of the two systems. In fact, Linux feels just as comfortable on an extended partition as on a primary partition. Linux will even install on your second hard disk without complaining. In contrast to this flexibility, the Redmond-based operating system will not play ball unless you put it on your system's first active primary partition.

Whereas Windows 95/98/ME need the first active primary partition as their system drive, the NT-based variants, Windows 2000 and XP just need to store their start files there – the Windows directory and the rest of the system can be installed on a logical drive somewhere on an extended partition. However, the *ntldr* and *ntdetect.com* files, as well as the *boot.ini* configuration file (Figure 1), always need to be located on the first active primary partition. The NT family boot manager does not use the



```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Home Edition" /fastdetect
```

Figure 1: The Windows 2000 and XP boot manager reads its configuration from the boot.ini file.

Master Boot Record (MBR), but the boot sector on the partition with the boot files. If there happens to be another Windows system on the drive where you install Windows XP, the installation routine will automatically add the existing system to the boot menu.

Standard Approach

The most typical type of multiple boot system adds Linux on top of a previously installed Windows system. Most distributions use the Grub boot manager by default and automatically add the Windows installation to the Grub boot menu (Figure 2).

The Grub configuration file, */boot/grub/menu.lst*, tells Grub how to launch Windows. The simplest variant looks like this; it works with Windows 95/98/ME and with the NT family.

```
title Windows
root (hd0,1)
chainloader +1
```

The line starting with *title* specifies the name for the Grub menu entry. The *root* line tells Grub which partition contains the system you wish to launch.

This syntax is a little different from the Linux standard, */dev/hdx*. *hd0* refers to the hard disk; Grub starts counting at *0*. The first disk attached to the first IDE controller, which would be */dev/hda* in Linux notation, is *hd0* for Grub. The partition follows the comma, and again Grub counts from *0*. Thus, our sample entry would boot a Windows system that lives on */dev/hda2*. It is quite common to find Windows installed on the second primary partition, as many manufacturers set up a recovery partition on */dev/hda1*.

Instead of the *root* keyword, some distributions use *rootnoverify*. This keyword tells Grub not to try to read the filesystem belonging to the partition that follows. As long as you are just running the code in the boot sector of the partition, there is no difference between *root* and *rootnoverify*.

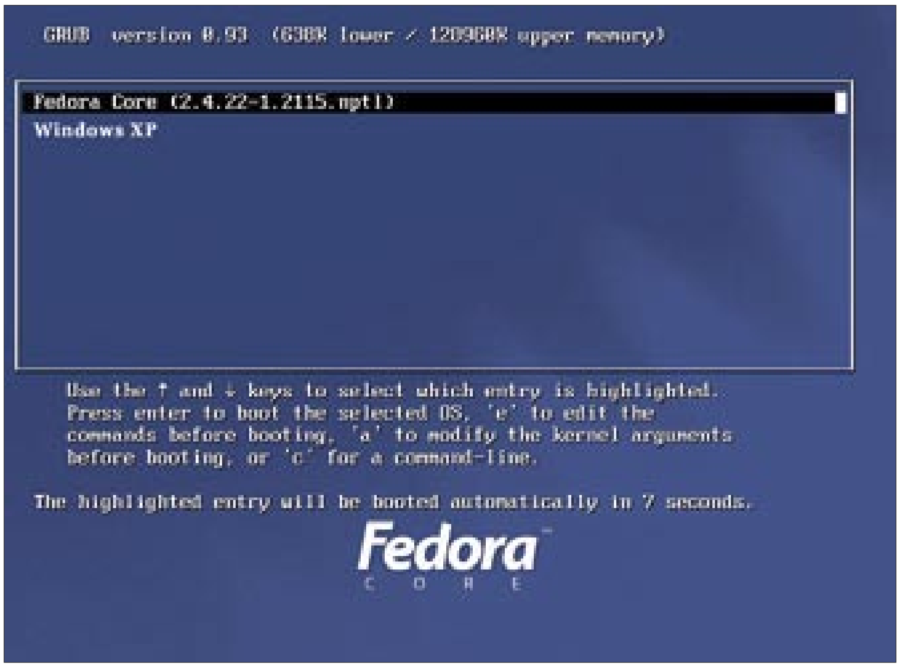
The instructions for booting the Windows system contain a line that reads *chainloader +1*. The *+1* parameter tells Grub to run the boot code following the *root* partition. However, you could instead specify the path to a file with the boot code for the system. You can try this out by getting Linux to write the boot sector of a Windows 2000 or XP partition to a file titled */boot/ntsector* using the following command:

```
dd if=/dev/hda1 ➥
of=/boot/ntsector ➥
bs=512 count=1
```

To run the NTloader from this file on your Linux partition, you then need to add the following entry to your *menu.lst*

```
title Windows
root (hd0,4)
chainloader /boot/➥
ntsector
```

However, you need to specify *root* to read the Linux partition, which is */dev/hda5* in our example, if you want Grub to locate the file. The *rootnoverify* directive will not work in this case, as Grub's file system support is needed to read the boot sector from a file. This approach will only work if the file with the boot code is stored on a partition with a file system that Grub supports. Besides the common Linux file systems, *ext2*, *ext3*, *reiser*, and *xfs*, other file systems could also be the VFAT file system used for



```
GRUB  version 0.93  (638K lower / 1209608K upper memory)

 Fedora Core (2.4.22-1.2115.nptl)
 Windows XP

    Use the ↑ and ↓ keys to select which entry is highlighted.
    Press enter to boot the selected OS, 'e' to edit the
    commands before booting, 'a' to modify the kernel arguments
    before booting, or 'c' for a command-line.

 The highlighted entry will be booted automatically in 7 seconds.
```

Figure 2: Most distributions automatically set up the Grub boot manager and add Windows to the menu.

Windows 95, Windows 98, and Windows ME systems.

It is also quite common to see a *makeactive* option in the Windows boot entries. The line needs to follow the line that passes the partition to Grub. What this option actually does is set the active flag for the partition, and this is necessary because Windows needs the active primary partition to boot. Most distributors put this entry in the *menu.lst* file just to make sure, but you only need it if you have multiple primary partitions on your hard disk, and if one of the non-Windows partitions has set the active flag.

Virtual Swaps

There is another scenario that involves slightly more work for users wanting to launch Windows. The idea behind this scenario is to attach a hard disk to the second IDE controller. Any existing Windows XP or 9x installations would refuse to boot from this disk. But Grub steps in with a *map* option that supports virtual disk swapping. The boot manager expects the disks to be swapped following the *map* option. The following entry would swap */dev/hda* and */dev/hdb* and boot a windows system installed on */dev/hdb1*:

```
title Windows
map (hd0) (hd1)
map (hd1) (hd0)
root (hd1,0)
chainloader +1
```

It is important to note that, after swapping, Grub still refers to the disk connected to the second IDE controller as *hd1* – so as far as Grub is concerned, virtual disk swapping has no after effects.

Windows in Second

Linux users wanting to add Windows to their installation are in for even more trouble. Both 95/98/ME and the NT family overwrite the MBR during a standard install, removing the Grub instance that resides there. If you plan this in advance and have a Grub boot floppy, you should have no trouble reinstating Grub later. To create a Grub install floppy, first run *dd* to write the *stage 1* file to the boot sector of a floppy. Assuming you have installed and set up Grub, the file will be located below the */boot/grub* directory, just like *menu.lst*. Typing

```
dd if=stage1 of=/dev/fd0 ⮐
bs=512 count=1
```

in this directory will send *stage1* to the floppy. The following command

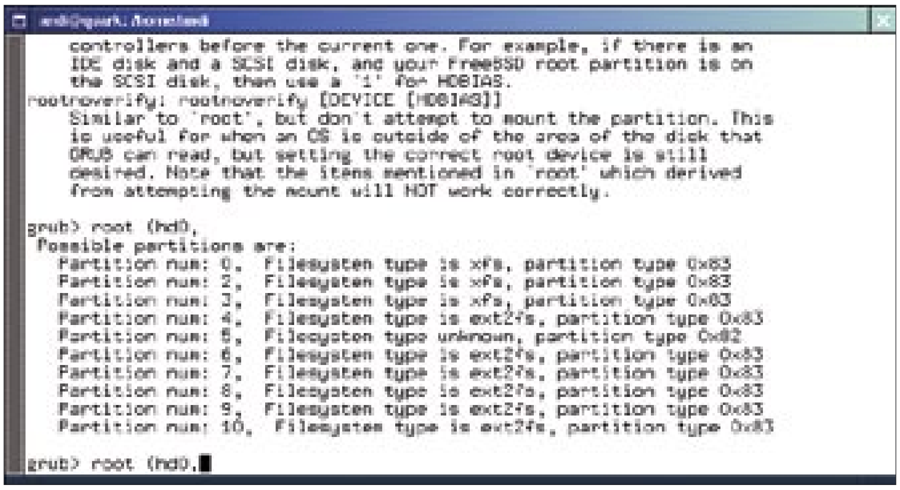
```
dd if=stage2 ⮐
of=/dev/fd0 bs=512 seek=1
```

does the same thing for *stage2*. The *seek = 1* parameter ensures that the command will not overwrite the *stage1* file transferred in the last step, but will instead store *stage2* behind *stage1*.

After completing these steps, your boot floppy now has everything you need to reinstall Grub. After booting, Grub will display the integrated command line without a menu. The prompt supports tab completion and also has online help. Entering *help* displays a list of commands, and *help commandname* gives you an explanation of a command.

Reinstalling Grub

To reinstall Grub, you first have to point Grub to its root partition, that is, the partition with the */boot/grub* directory and the stage files. If */boot* is located on */dev/hda5*, you need to type the following at the Grub prompt: *root (hd0,5)*. Tab completion is supported, so you will not normally need to type the closing parentheses; for example, you can type *root*



```
     controllers before the current one. For example, if there is an
     IDE disk and a SCSI disk, and your FreeBSD root partition is on
     the SCSI disk, then use a '1' for HDBIAS.
rootnoverify: rootnoverify [DEVICE [HDBIAS]]
     Similar to 'root', but don't attempt to mount the partition. This
     is useful for when an OS is outside of the area of the disk that
     GRUB can read, but setting the correct root device is still
     desired. Note that the items mentioned in 'root' which derived
     from attempting the mount will NOT work correctly.

grub> root (hd0,
 Possible partitions are:
   Partition num: 0,  Filesystem type is xfs, partition type 0x83
   Partition num: 2,  Filesystem type is xfs, partition type 0x83
   Partition num: 3,  Filesystem type is xfs, partition type 0x83
   Partition num: 4,  Filesystem type is ext2fs, partition type 0x83
   Partition num: 5,  Filesystem type unknown, partition type 0x82
   Partition num: 6,  Filesystem type is ext2fs, partition type 0x83
   Partition num: 7,  Filesystem type is ext2fs, partition type 0x83
   Partition num: 8,  Filesystem type is ext2fs, partition type 0x83
   Partition num: 9,  Filesystem type is ext2fs, partition type 0x83
   Partition num: 10,  Filesystem type is ext2fs, partition type 0x83

grub> root (hd0,
```

Figure 3: The Grub command line gives you online help and tab completion.

(hd0, and press the tab key to view a list of the partitions on */dev/hda*, including file system information (Figure 3).

If you are not sure which partition */boot* is stored on, you can type *find /boot/grub/stage1* to find out. The command lists any partitions with a file of this name. Now that Grub knows where its root device is located, giving the *setup (hd0)* command will install Grub to the MBR on */dev/hda*.

Chaining the NT Loader

If all of this sounds too dangerous to you, and assuming that you just want to try out Linux, you might like to use the little known NT loader chain loading function. This technique assumes you have Windows NT or any other member of the NT family; the older 95/98 and ME systems do not have a boot manager. Chain loading works in a similar way to the Grub example with the NT boot sector. In contrast to Grub, however, the NT loader can only run boot code from a file and only supports the native Windows file systems, VFAT and NTFS.

During installation, more cautious users will tell the system not to write the boot manager to the MBR but to the boot sector on the Linux partition. But this will not work if you use the XFS file system, which stores management information in this area and would overwrite the boot manager. You can boot Linux in safe mode, using the distribution media, and run *dd* to write the boot sector to a file when you get there. The syntax for a Linux version on */dev/hda5* is as follows

```
dd if=/dev/hda5 of=bootsec.lin ↩
bs=512 count=1
```

The *bootsec.lin* file contains the installed Grub and must be transferred to one of the Windows partitions using a USB stick or floppy if needed. The root directory of drive *c:*, which is where *boot.ini* resides, is a good place to store the file. Add a line to the *boot.ini* file that specifies the location of *bootsec.lin*. The entry points to the file with the boot sector and the name that will appear in the Windows boot menu, for example:

```
c:\bootsec.lin="Launch Grub"
```

Unless you have a *timeout* of *0*, the Windows boot manager will display a menu the next time you start your machine, allowing users to select between the Windows and the *Launch Grub* entry. Selecting the *Launch Grub* entry takes you to the Grub menu, where you can select the Linux system or simply the NT loader again.

Conclusion

Many popular Linux distributions set up a dual boot configuration automatically if you install Linux on top of an existing Windows system. But for those of you looking to install Windows on a Linux machine, or if you want to try anything out of the ordinary, the techniques described in this article will help you get off to a good start. ■

INFO

[1] Grub: *http://www.gnu.org/software/grub*

Advertisment