Linux Partition HOWTO

Tony Harris

Kristian Koehntopp

Revision History

Revision 3.3	10 July 2001
Corrected Section 6, calculation of cylinder numbers	
Revision 3.2	1 September 2000
Dan Scott provides sgml conversion 2 Oct. 2000. Rewrote Introduction. Rewrote discussion on device names in Logical Devices. Reorganized Partition Types. Edited Partition Requirements. Added Recovering a deleted partition table.	
Revision 3.1	12 June 2000
Corrected swap size limitation in Partition Requirements, updated various links in Introduction, added submitted example in How to Partition with fdisk, added file system discussion in Partition Requirements.	
Revision 3.0	1 May 2000
First revision by Tony Harris based on Linux Partition HOWTO by Kristian Koehntopp.	
Revision 2.4	3 November 1997
Last revision by Kristian Koehntopp.	

This Linux Mini−HOWTO teaches you how to plan and create partitions on IDE and SCSI hard drives. It discusses partitioning terminology and considers size and location issues. Use of the fdisk partitioning utility for creating and recovering of partition tables is covered. The most recent version of this document is here.

Table of Contents

1. Introduction

1.1. What is a partition?

Partitioning is a means to divide a single hard drive into many logical drives. A partition is a contiguous set of blocks on a drive that are treated as an independant disk. A partition table (the creation of which is the topic of this HOWTO) is an index that relates sections of the hard drive to partitions.

Why have multiple partitions?

- Encapsulate your data. Since file system corruption is local to a partition, you stand to lose only some of your data if an accident occurs.
- Increase disk space efficiency. You can format partitions with varying block sizes, depending on your usage. If your data is in a large number of small files (less than 1k) and your partition uses 4k sized blocks, you are wasting 3k for every file. In general, you waste on average one half of a block for every file, so matching block size to the average size of your files is important if you have many files.
- Limit data growth. Runaway processes or maniacal users can consume so much disk space that the operating system no longer has room on the hard drive for its bookkeeping operations. This will lead to disaster. By segregating space, you ensure that things other than the operating system die when allocated disk space is exhausted.

1.2. Constraints

- Partitions must not overlap. This will cause data corruption and other spooky stuff.
- There ought to be be no gap between adjacent partitions. While this is not harmful, you are wasting precious disk space by leaving space between partitions.
- A disk need not be partitioned completely. You may decide to leave some unpartitioned space at the end of your disk and partition it later.
- Partitions cannot be moved but they can be resized and copied using special software. This HOWTO only covers the use of the **fdisk** utility, which does not permit any of these operations.

1.3. Other Partitioning Software:

- **sfdisk**: a command−line version of fdisk
- **cfdisk**: a curses−based version of fdisk
- **parted**: Gnu partition editor
- Partition Magic: a commercial utility to create, resize, merge and convert partitions, without destroying data.
- Disk Drake: a Perl/Gtk program to create, rsize, and delete partitions

1.4. Related HOWTOs

Table 1. Related HOWTOs

Title	Author	Description
Linux Multiple Disk System Tuning	Gjoen Stein	How to estimate the various size and speed requirements for different parts of the filesystem.
Linux Large Disk	Andries Brouwer	Instructions and considerations regarding disks with more than 1024 cylinders
Linux Quota	Albert M.C. Tam	Instructions on limiting disk space usage per user (quotas)
Partition−Rescue mini−HOWTO	Jean−Daniel Dodin	How to restore linux partitions after they have been deleted by a Windows install. Does not appear to preserve data.
Linux ADSM Backup	Thomas Koenig	Instructions on integrating Linux into an IBM ADSM backup environment.
Linux Backup with MSDOS	Christopher Neufeld	Information about MS−DOS driven Linux backups.
Linux HOWTO Index	Tim Bynum	Instructions on writing and submitting a HOWTO document

1.5. Additional information on your system:

- `/usr/src/linux/Documentation`
    - `ide.txt`: Info about your IDE drivers
    - `scsi.txt`: Info about your SCSI drivers

2. Devices

There is a special nomenclature that linux uses to refer to hard drive partitions that must be understood in order to follow the discussion on the following pages.

In Linux, partitions are represented by device files. These are phoney files located in `/dev`. Here are a few entries:

```
brw-rw----    1 root     disk       3,   0 May  5  1998 hda
brw-rw----    1 root     disk       8,   0 May  5  1998 sda
crw-------    1 root     tty        4,  64 May  5  1998 ttyS0
```

A device file is a file with type c (for "character" devices, devices that do not use the buffer cache) or b (for "block" devices, which go through the buffer cache). In Linux, all disks are represented as block devices only.

2.1. Device names

2.1.1. Naming Convention

By convention, IDE drives will be given device names `/dev/hda` to `/dev/hdd`. The first drive is 'a' the second drive 'b' and so on. For example, `/dev/hda` is the first drive on the first IDE controller and `/dev/hdd` is the second drive on the second controller (the fourth IDE drive in the computer). You can write to these devices directly (using cat or dd). However, since these devices represent the entire disk, starting at the first block, you can mistakenly overwrite the master boot record and the partition table, which will render the drive unusable.

Once a drive has been partitioned, the partitions will be represented as numbers on the end of the names. For example, the second partition on the second drive will be `/dev/hdb2`. SCSI drives follow a similar pattern; They are represented by 'sd' instead of 'hd'. The first partition of the second SCSI drive would therefore be `/dev/sdb1`.

Primary partitions (Section 3.3) on a disk are 1, 2, 3 and 4. Logical partitions (Section 3.4) have numbers 5 and up, for reasons explained later (Section 5.1.3).

2.1.2. Name Assignment

Under (Sun) Solaris and (SGI) IRIX, the device name given to a SCSI drive has some relationship to where you plug it in. Under linux, there is only wailing and gnashing of teeth. Lower SCSI ID numbers are assigned lower−order letters, so if you remove one drive from the chain, the names of the higher ID number drives will change. If you have two SCSI controllers in your linux box, you will need to examine the output of `/bin/dmesg` in order to see what name each drive was assigned. If you remove one of two controllers, the remaining controller might have all its drives renamed. Grrr...

This is all you have to know to deal with linux disk devices. For the sake of completeness, see Kristian's discussion of device numbers below.

2.2. Device numbers

The only important thing with a device file are its major and minor device numbers, which are shown instead of the file size:

```
$ ls -l /dev/hda
```

Table 2. Device file attributes

brw−rw−−−−	1	root	disk	3,	0	Jul 18 1994	/dev/hda
permissions		owner	group	major device number	minor device number	date	device name

When accessing a device file, the major number selects which device driver is being called to perform the input/output operation. This call is being done with the minor number as a parameter and it is entirely up to the driver how the minor number is being interpreted. The driver documentation usually describes how the driver uses minor numbers. For IDE disks, this documentation is in `/usr/src/linux/Documentation/ide.txt`. For SCSI disks, one would expect such documentation in `/usr/src/linux/Documentation/scsi.txt`, but it isn't there. One has to look at the driver source to be sure (`/usr/src/linux/driver/scsi/sd.c`:184−196). Fortunately, there is Peter Anvin's list of device numbers and names in `/usr/src/linux/Documentation/devices.txt`; see the entries for block devices, major 3, 22, 33, 34 for IDE and major 8 for SCSI disks. The major and minor numbers are a byte each and that is why the number of partitions per disk is limited.

3. Partition Types

3.1. Partition Types

A partition is labeled to host a certain kind of file system. Such a file system could be the linux standard ext2 file system or linux swap space, or even foreign file systems like (Microsoft) NTFS or (Sun) UFS. There is a numerical code associated with each partition type. For example, the code for ext2 is `0x83` and linux swap is 0x82.

3.2. Foreign Partition Types

The partition type codes have been arbitrarily chosen (you can't figure out what they should be) and they are particular to a given operating system. Therefore, it is theoretically possible that if you use two operating systems with the same hard drive, the same code might be used to designate two different partition types.

OS/2 marks its partitions with a `0x07` type and so does Windows NT's NTFS. MS−DOS allocates several type codes for its various flavors of FAT file systems: 0x01, `0x04` and `0x06` are known. DR−DOS used `0x81` to indicate protected FAT partitions, creating a type clash with Linux/Minix at that time, but neither Linux/Minix nor DR−DOS are widely used any more.

3.3. Primary Partitions

The number of partitions on an Intel−based system was limited from the very beginning: The original partition table was installed as part of the boot sector and held space for only four partition entries. These partitions are now called primary partitions.

3.4. Logical Partitions

One primary partition of a hard drive may be subpartitioned. These are logical partitions. This effectively allows us to skirt the historical four partition limitation.

The primary partition used to house the logical partitions is called an extended partition and it has its own file system type (0x05). Unlike primary partitions, logical partitions must be contiguous. Each logical partition contains a pointer to the next logical partition, which implies that the number of logical partitions is unlimited. However, linux imposes limits on the total number of any type of prtition on a drive, so this effectively limits the number of logical partitions. This is at most 15 partitions total on an SCSI disk and 63 total on an IDE disk.

3.5. Swap Partitions

Every process running on your computer is allocated a number of blocks of RAM. These blocks are called pages. The set of in−memory pages which will be referenced by the processor in the very near future is called a "working set." Linux tries to predict these memory accesses (assuming that recently used pages will be used again in the near future) and keeps these pages in RAM if possible.

If you have too many processes running on a machine, the kernel will try to free up RAM by writing pages to disk. This is what swap space is for. It effectively increases the amount of memory you have available. However, disk I/O is very slow compared to reading from and writing to RAM.

If memory becomes so scarce that the kernel pages out from the working set of one process in order to page in for another, the machine is said to be thrashing. Expect performance to drop by approximately the ratio between memory access speed and disk access speed. Swap space is something you need to have, but it is no substitute for sufficient RAM. See Section 4.4.1 for tips on determining the size of swap space you need.

4. Partitioning requirements

4.1. What Partitions do I need?

Boot Drive: If you want to boot your operating system from the drive you are about to partition, you will need:

- A primary partition
- One or more swap partitions
- Zero or more primary/logical partitions

Any other drive:

- One or more primary/logical partitions
- Zero or more swap partitions

4.2. Discussion:

Boot Partition:

Your boot partition ought to be a primary partition, not a logical partition. This will ease recovery in case of disaster, but it is not technically necessary. It must be of type `0x83` "Linux native". If you are using **lilo**, your boot partition must be contained within the first 1024 cylinders of the drive. (Typically, the boot partition need only contain the kernel image.)

If you have more than one boot partition (from other OSs, for example,) keep them all in the first 1024 cylinders (*All* DOS partitions must be within the first 1024). If you are using a means other than lilo loading your kernel (for example, a boot disk or the **LOADLIN.EXE** MS−DOS based Linux loader), the partition can be anywhere. See the Large−disk HOWTO for details.

Swap Partition:

Unless you swap to files you will need a dedicated swap partition. It must be of type `0x82` "Linux swap". It may be positioned anywhere on the disk (but see notes on placement: Section 4.4.2). Either a primary or logical partition can be used for swap. More than one swap partition can exist on a drive. 8 total (across drives) are permitted. See notes on swap size: Section 4.4.1.

Logical Partition:

A single primary partition must be used as a container (extended partition) for the logical partitions. The extended partition can go anywhere on the disk. The logical partitions must be contiguous, but needn't fill the extended partition.

4.3. File Systems

4.3.1. Which file systems need their own partitions?

Everything in your linux file system can go in the same (single) partition. However, there are circumstances when you may want to restrict the growth of certain file systems. For example, if your mail spool was in the same partition as your root fs and it filled the remaining space in the partition, your computer would basically hang.

/var

This fs contains spool directories such as those for mail and printing. In addition, it contains the error log directory. If your machine is a server and develops a chronic error, those msgs can fill the partition. Server computers ought to have /var in a different partition than /.

/usr

This is where most executable binaries go. In addition, the kernel source tree goes here, and much documentation.

/tmp

Some programs write temporary data files here. Usually, they are quite small. However, if you run computationally intensive jobs, like science or engineering applications, hundreds of megabytes could be required for brief periods of time. In this case, keep /tmp in a different partition than /.

/home

This is where users home directories go. If you do not impose quotas on your users, this ought to be in its own partition.

/boot

This is where your kernel images go. If you use MSDOS, which must go in the first 1024 cylinders, you need to at least get this partition in there in order to ensure that lilo can see it. If you have a drive larger than 1024 cylinders, making this your first partition guarantees that it will be visible to lilo.

4.3.2. File lifetimes and backup cycles as partitioning criteria

With ext2, partitioning decisions should be governed by backup considerations and to avoid external fragmentation (Section 7.3) from different file lifetimes.

Files have lifetimes. After a file has been created, it will remain some time on the system and then be removed. File lifetime varies greatly throughout the system and is partly dependent on the pathname of the file. For example, files in `/bin`, `/sbin`, `/usr/sbin`, `/usr/bin` and similar directories are likely to have a very long lifetime: many months and above. Files in `/home` are likely to have a medium lifetime: several weeks or so. File in `/var` are usually short lived: Almost no file in `/var/spool/news` will remain longer than a few days, files in `/var/spool/lpd` measure their lifetime in minutes or less.

For backup it is useful if the amount of daily backup is smaller than the capacity of a single backup medium. A daily backup can be a complete backup or an incremental backup.

You can decide to keep your partition sizes small enough that they fit completely onto one backup medium (choose daily full backups). In any case a partition should be small enough that its daily delta (all modified files) fits onto one backup medium (choose incremental backup and expect to change backup media for the weekly/monthly full dump − no unattended operation possible).

Your backup strategy depends on that decision.

When planning and buying disk space, remember to set aside a sufficient amount of money for backup! Unbackuped data is worthless! Data reproduction costs are much higher than backup costs for virtually everyone!

For performance it is useful to keep files of different lifetimes on different partitions. This way the short lived files on the news partition may be fragmented very heavily. This has no impact on the performance of the `/` or `/home` partition.

4.4. Swap Partitions

4.4.1. How large should my swap space be?

If you have decided to use a dedicated swap partition, which is generally a Good Idea [tm], follow these guidelines for estimating its size:

- In Linux RAM and swap space add up (This is not true for all Unices). For example, if you have 8 MB of RAM and 12 MB swap space, you have a total of about 20 MB virtual memory.
- When sizing your swap space, you should have at least 16 MB of total virtual memory. So for 4 MB of RAM consider at least 12 MB of swap, for 8 MB of RAM consider at least 8 MB of swap.
- Currently, the maximum size of a swap partition is architecture−dependent. For i386 and PowerPC, it is approximately 2Gb. It is 128Gb on alpha, 1Gb on sparc, and 3Tb on sparc64. For linux kernels 2.1 and earlier, the limit is 128Mb. The partition may be larger than 128 MB, but excess space is never used. If you want more than 128 MB of swap for a 2.1 and earlier kernel, you have to create multiple swap partitions. See the man page for mkswap for details.
- When sizing swap space, keep in mind that too much swap space may not be useful at all.

A very old rule of thumb in the days of the PDP and the Vax was that the size of the working set of a program is about 25% of its virtual size. Thus it is probably useless to provide more swap than three times your RAM.

But keep in mind that this is just a rule of thumb. It is easily possible to create scenarios where programs have extremely large or extremely small working sets. For example, a simulation program with a large data set that is accessed in a very random fashion would have almost no noticeable locality of reference in its data segment, so its working set would be quite large.

On the other hand, an xv with many simultaneously opened JPEGs, all but one iconified, would have a very large data segment. But image transformations are all done on one single image, most of the memory occupied by xv is never touched. The same is true for an editor with many editor windows where only one window is being modified at a time. These programs have − if they are designed properly − a very high locality of reference and large parts of them can be kept swapped out without too severe performance impact.

One could suspect that the 25% number from the age of the command line is no longer true for modern GUI programs editing multiple documents, but I know of no newer papers that try to verify these numbers.

So for a configuration with 16 MB RAM, no swap is needed for a minimal configuration and more than 48 MB of swap are probably useless. The exact amount of memory needed depends on the application mix on the machine (what did you expect?).

4.4.2. Where should I put my swap space?

- Mechanics are slow, electronics are fast.

  Modern hard disks have many heads. Switching between heads of the same track is fast, since it is purely electronic. Switching between tracks is slow, since it involves moving real world matter.

  So if you have a disk with many heads and one with less heads and both are identical in other parameters, the disk with many heads will be faster.

  Splitting swap and putting it on both disks will be even faster, though.
- Older disks have the same number of sectors on all tracks. With these disks it will be fastest to put your swap in the middle of the disks, assuming that your disk head will move from a random track towards the swap area.
- Newer disks use ZBR (zone bit recording). They have more sectors on the outer tracks. With a constant number of rpms, this yields a far greater performance on the outer tracks than on the inner ones. Put your swap on the fast tracks.
- Of course your disk head will not move randomly. If you have swap space in the middle of a disk between a constantly busy home partition and an almost unused archive partition, you would be better of if your swap were in the middle of the home partition for even shorter head movements. You would be even better off, if you had your swap on another otherwise unused disk, though.

Summary: Put your swap on a fast disk with many heads that is not busy doing other things. If you have multiple disks: Split swap and scatter it over all your disks or even different controllers.

Even better: Buy more RAM.

---

5. Partitioning with fdisk

5.1. Partitioning with fdisk

This section shows you how to actually partition your hard drive with the **fdisk** utility. Linux allows only 4 primary partitions. You can have a much larger number of logical partitions by sub−dividing one of the primary partitions. Only one of the primary partitions can be sub−divided.

Examples:

1. Four primary partitions (Section 5.1.2)
2. Mixed primary and logical partitions (Section 5.1.3)

5.1.1. Notes about fdisk:

fdisk is started by typing (as root) **`fdisk`** ***`device`*** at the command prompt. "device" (Section 2.1.1) might be something like `/dev/hda` or `/dev/sda`. The basic fdisk commands you need are:

p

print the partition table

n

create a new partition

d

delete a partition

q

quit without saving changes

w

write the new partition table and exit

Changes you make to the partition table do not take effect until you issue the write (w) command. Here is a sample partition table:

```
Disk /dev/hdb: 64 heads, 63 sectors, 621 cylinders
Units = cylinders of 4032 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdb1   *         1       184    370912+  83  Linux
/dev/hdb2           185       368    370944   83  Linux
/dev/hdb3           369       552    370944   83  Linux
/dev/hdb4           553       621    139104   82  Linux swap
```

The first line shows the geometry of your hard drive. It may not be physically accurate, but you can accept it as though it were. The hard drive in this example is made of 32 double−sided platters with one head on each side (probably not true). Each platter has 621 concentric tracks. A 3−dimensional track (the same track on all disks) is called a cylinder. Each track is divided into 63 sectors. Each sector contains 512 bytes of data. Therefore the block size in the partition table is 64 heads * 63 sectors * 512 bytes er...divided by 1024. (See 4 for discussion on problems with this calculation.)

The start and end values are cylinders. The first cylinder (0) is reserved for information about the drive layout.

5.1.2. Four primary partitions

*The overview:*Decide on the size (Section 4.4.1) of your swap space and where (Section 4.4.2) it ought to go. Divide up the remaining space for the three other partitions.

Example:

I start fdisk from the shell prompt:

```
# fdisk /dev/hdb
```

which indicates that I am using the second drive on my IDE controller. (See Section 2.1.) Now I need to plan my layout. In this example, I want to use only primary partitions for my linux partitions and my swap space. When I print the (empty) partition table, I just get configuration information.

```
Command (m for help): p

Disk /dev/hdb: 64 heads, 63 sectors, 621 cylinders
Units = cylinders of 4032 * 512 bytes
```

I knew that I had a 1.2Gb drive, but now I really know: 64 * 63 * 512 * 621 = 1281982464 bytes. I decide to reserve 128Mb of that space for swap, leaving 1153982464. If I use one of my primary partitions for swap, that means I have three left for ext2 partitions. Divided equally, that makes for 384Mb per partition. Now I get to work.

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1−4)
P
Partition number (1−4): 1
First cylinder (1−621, default 1):<RETURN>
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1−621, default 621): +384M
```

Next, I set up the partition I want to use for swap:

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1−4)
p
Partition number (1−4): 2
First cylinder (197−621, default 197):<RETURN>
Using default value 197
Last cylinder or +size or +sizeM or +sizeK (197−621, default 621): +128M
```

Now the partition table looks like this:

```
   Device Boot    Start       End    Blocks   Id  System
/dev/hdb1             1       196    395104   83  Linux
/dev/hdb2           197       262    133056   83  Linux
```

I set up the remaining two partitions the same way I did the first. Finally, I make the first partition bootable:

```
Command (m for help): a
Partition number (1-4): 1
```

And I make the second partition of type swap:

```
Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): 82
Changed system type of partition 2 to 82 (Linux swap)
Command (m for help): p
```

The end result:

```
Disk /dev/hdb: 64 heads, 63 sectors, 621 cylinders
Units = cylinders of 4032 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdb1   *         1       196    395104+  83  Linux
/dev/hdb2           197       262    133056   82  Linux swap
/dev/hdb3           263       458    395136   83  Linux
/dev/hdb4           459       621    328608   83  Linux
```

Finally, I issue the write command (w) to write the table on the disk.

For more information, see:

- Section 7.1
- Section 7
- Section 7.2

5.1.3. Mixed primary and logical partitions

The overview: create one use one of the primary partitions to house all the extra partitions. Then create logical partitions within it. Create the other primary partitions before or after creating the logical partitions.

Example:

I start fdisk from the shell prompt:

```
# fdisk /dev/sda
```

which indicates that I am using the first drive on my SCSI chain. (See Section 2.1.)

First I figure out how many partitions I want. I know my drive has a 183Gb capacity and I want 26Gb partitions (because I happen to have back−up tapes that are about that size).

```
183Gb / 26Gb = ~7
```

so I will need 7 partitions. Even though fdisk accepts partition sizes expressed in Mb and Kb, I decide to calculate the number of cylinders that will end up in each partition because fdisk reports start and stop points in cylinders. I see when I enter fdisk that I have 22800 cylinders.

```
> The number of cylinders for this disk is set to 22800.  There is
> nothing wrong with that, but this is larger than 1024, and could in
> certain setups cause problems with: 1) software that runs at boot
> time (e.g., LILO) 2) booting and partitioning software from other
> OSs  (e.g., DOS FDISK, OS/2 FDISK)
```

So, 22800 total cylinders divided by seven partitions is 3258 cylinders. Each partition will be about 3258 cylinders long. I ignore the warning msg because this is not my boot drive (Section 4).

Since I have 4 primary partitions, 3 of them can be 3258 long. The extended partition will have to be (4 * 3258), or 13032, cylinders long in order to contain the 4 logical partitions.

I enter the following commands to set up the first of the 3 primary partitions (stuff I type is bold):

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
P
Partition number (1-4): 1
First cylinder (1-22800, default 1): <RETURN>
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-22800, default 22800): 3258
```

The last partition is the extended partition:

```
Partition number (1-4): 4
First cylinder (9775-22800, default 9775): <RETURN>
Using default value 9775
Last cylinder or +size or +sizeM or +sizeK (9775-22800, default 22800): <RETURN>
Using default value 22800
```

The result, when I issue the print table command is:

```
/dev/sda1             1      3258  26169853+  83  Linux
/dev/sda2          3259      6516  26169885   83  Linux
/dev/sda3          6517      9774  26169885   83  Linux
/dev/sda4          9775     22800 104631345    5  Extended
```

Next I segment the extended partition into 4 logical partitions, starting with the first logical partition, into 3258−cylinder segments. The logical partitions automatically start from /dev/sda5.

```
Command (m for help):  n
First cylinder (9775-22800, default 9775): <RETURN>
Using default value 9775
Last cylinder or +size or +sizeM or +sizeK (9775-22800, default 22800): 13032
```

The end result is:

```
   Device Boot    Start       End    Blocks   Id  System
/dev/sda1             1      3258  26169853+  83  Linux
/dev/sda2          3259      6516  26169885   83  Linux
/dev/sda3          6517      9774  26169885   83  Linux
/dev/sda4          9775     22800 104631345    5  Extended
/dev/sda5          9775     13032  26169853+  83  Linux
/dev/sda6         13033     16290  26169853+  83  Linux
/dev/sda7         16291     19584  26459023+  83  Linux
/dev/sda8         19585     22800  25832488+  83  Linux
```

Finally, I issue the write command (w) to write the table on the disk. To make the partitions usable, I will have to format (Section 7) each partition and then mount (Section 7.2) it.

5.1.4. Submitted Examples

I'd like to submit my partition layout, because it works well with any distribution of Linux (even big RPM based ones). I have one hard drive that ... is 10 gigs, exactly. Windows can't see above 9.3 gigs of it, but Linux can see it all, and use it all. It also has much more than 1024 cylenders.

Table 3. Partition layout example

Partition	Mount point	Size

/dev/hda1	/boot	(15 megs)
/dev/hda2	windows 98 partition	(2 gigs)
/dev/hda3	extended	(N/A)
/dev/hda5	swap space	(64 megs)
/dev/hda6	/tmp	(50 megs)
/dev/hda7	/	(150 megs)
/dev/hda8	/usr	(1.5 gigs)
/dev/hda9	/home	(rest of drive)

I test new kernels for the USB mass storage, so that explains the large /boot partition. I install LILO into the MBR, and by default I boot windows (I'm not the only one to use this computer).

I also noticed that you don't have any REAL examples of partition tables, and for newbies I HIGHLY suggest putting quite a few up. I'm freshly out of the newbie stage, and partitioning was what messed me up the most.

Valkor

6. Recovering a Deleted Partition Table

1. Make a partition that is at least as big as your first partition was. You can make it larger than the original partition by any amount. If you underestimate, there will be much wailing and gnashing of teeth.

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
P
Partition number (1-4): 1
First cylinder (1-23361, default 1): <RETURN>
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-22800, default 22800): 13032

Command (m for help): w
```

2. Run **dumpe2fs** on the first partition and grep out the block count.

   Example:

```
           % dumpe2fs /dev/sda1 | grep "Block count:"
           Block count:              41270953
```

   If you are uncertain about this value, repeat Step 1 with a bigger partition size. If the block count changes, then you underestimated the size of the original partition. Repeat Step 1 until you get a stable block count.

3. Remove the partition you just created

```
         Command (m for help): d
         Partition number (1-4): 1
```

4.

   Make a new partition with the exact size you got from the block count. Since you cannot enter block size in fdisk, you need to figure out how many cylinders to request. Here is the formula:

```
  (number of needed cylinders) = (number of blocks) / (block size)

  (block size) = (unit size) / 1024

  (unit size) = (number of cylinders) * (number of heads) * (number of sectors/cylinder) * (number of bytes/sector)
```

   Consider the following example, where a hard drive has been partitioned into four primary partitions of 1, 2, 4, and 8 cylinders.

```
disk /dev/sda: 16 heads, 63 sectors, 23361 cylinders
Units = cylinders of 1008 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/sda1             1         2       976+  83  Linux
/dev/sda2             3         5      1512   83  Linux
/dev/sda3             6        10      2520   83  Linux
/dev/sda4            11        19      4536   83  Linux
```

   fdisk provides the configuration information I need in the head of the output. The unit size is `516096` (`16` heads * `63` sectors/cyl * `512` bytes/sector). The block size is `504` (`516096 / 1024`). The number of needed cylinders for the second partition is therefore `3` (`1512` blocks `/ 504`). The partition table shows that this is indeed the case: the first cylinder is `3`, the second `4`, and the last is `5`, for a total of three cylinders. The number of needed cylinders for the third partition is calculated similarly: `2520` blocks `/ 504 = 5`, which corresponds to blocks `6,7,8,9,10` . Notice that this calculation does not work for the first partition because the block count is wrong (`976` instead of `1008`). The plus sign indicates that not all the blocks are included in the fdisk value. When you try the calculation (`976 / 504`) you get `1.937`. Knowing that the number of cylinders must be an integer, you can simply round up.

5. Run **e2fsck** on it to verify that you can read the new partition.
6. Repeat Steps 1−5 on remaining partitions.

Remount your partitions. Amazingly, all of your data will be there.

Credit goes to:

- Mike Vevea, jedi sys admin and MGH's finest, for giving me these tips.

7. Formating Partitions

At the shell prompt, I begin making the file systems on my partitions. Continuing with the example in Section 5.1.3, this is:

```
# mke2fs /dev/sda1
```

I need to do this for each of my partitions, but not for /dev/sda4 (my extended partition). Linux supports types of file systems other than ext2. You can find out what kinds your kernel supports by looking in: `/usr/src/linux/include/linux/fs.h`

The most common file systems can be made with programs in `/sbin` that start with "mk" like **mkfs.msdos** and **mke2fs**.

7.1. Activating Swap Space

To set up a swap partition:

```
# mkswap -f /dev/hda5
```

To activate the swap area:

```
# swapon  /dev/hda5
```

Normally, the swap area is activated by the initialization scripts at boot time.

7.2. Mounting Partitions

Mounting a partition means attaching it to the linux file system. To mount a linux partition:

```
# mount -t ext2 /dev/sda1 /opt
```

−t ext2

File system type. Other types you are likely to use are:

- msdos (DOS)
- hfs (mac)
- iso9660 (CDROM)
- nfs (network file system)

/dev/sda1

Device name. Other device names you are likely to use:

- `/dev/hdb2` (second partition in second IDE drive)
- `/dev/fd0` (floppy drive A)
- `/dev/cdrom` (CDROM)

/opt

mount point. This is where you want to "see" your partition. When you type **`ls /opt`**, you can see what is in /dev/sda1. If there are already some directories and/or files under /opt, they will be invisible after this mount command.

7.3. Some facts about file systems and fragmentation

Disk space is administered by the operating system in units of blocks and fragments of blocks. In ext2, fragments and blocks have to be of the same size, so we can limit our discussion to blocks.

Files come in any size. They don't end on block boundaries. So with every file a part of the last block of every file is wasted. Assuming that file sizes are random, there is approximately a half block of waste for each file on your disk. Tanenbaum calls this "internal fragmentation" in his book "Operating Systems".

You can guess the number of files on your disk by the number of allocated inodes on a disk. On my disk

```
# df -i
Filesystem           Inodes   IUsed   IFree  %IUsed Mounted on
/dev/hda3              64256   12234   52022    19%  /
/dev/hda5              96000   43058   52942    45%  /var
```

there are about 12000 files on `/` and about 44000 files on `/var`. At a block size of 1 KB, about 6+22 = 28 MB of disk space are lost in the tail blocks of files. Had I chosen a block size of 4 KB, I had lost 4 times this space.

Data transfer is faster for large contiguous chunks of data, though. That's why ext2 tries to preallocate space in units of 8 contigous blocks for growing files. Unused preallocation is released when the file is closed, so no space is wasted.

Noncontiguous placement of blocks in a file is bad for performance, since files are often accessed in a sequential manner. It forces the operating system to split a disk access and the disk to move the head. This is called "external fragmentation" or simply "fragmentation" and is a common problem with MS−DOS file systems. In conjunction with the abysmal buffer cache used by MS−DOS, the effects of file fragmentation on performance are very noticeable. DOS users are accustomed to defragging their disks every few weeks and some have even developed some ritualistic beliefs regarding defragmentation.

None of these habits should be carried over to Linux and ext2. Linux native file systems do not need defragmentation under normal use and this includes any condition with at least 5% of free space on a disk. There is a defragmentation tool for ext2 called defrag, but users are cautioned against casual use. A power outage during such an operation can trash your file system. Since you need to back up your data anyway, simply writing back from your copy will do the job.

The MS−DOS file system is also known to lose large amounts of disk space due to internal fragmentation. For partitions larger than 256 MB, DOS block sizes grow so large that they are no longer useful (This has been corrected to some extent with FAT32). Ext2 does not force you to choose large blocks for large file systems, except for very large file systems in the 0.5 TB range (that's terabytes with 1 TB equaling 1024 GB) and above, where small block sizes become inefficient. So unlike DOS there is no need to split up large disks into multiple partitions to keep block size down.

Use a 1Kb block size if you have many small files. For large partitions, 4Kb blocks are fine.