Partition−Rescue HOWTO

Jean−Daniel Dodin

Revision History

Revision v3.3	2001−11−17	Revised by: jdd
Minor update − docbook & revision history − emacs use.		
Revision v3.2	2001−09−25	Revised by: jdd
Major update.		

Whow ! my disk is empty ! My Linux is gone ! If you have or fear to have one day or an other such a problem, read this...

Table of Contents

1. Beginning

1.1. What's in

This mini−HOWTO addresses only the lost partition table problem. This can be when :

- you have no more access to your computer, with the no operating system message ;
- you have installed a new system (i.e. Windows) and you see no more Linux, and Windows take up all the capacity of the disk ;
- you have just partitioned the wrong drive with fdisk (for example in the way of changing your hard drive).

Here you will learn that if you know the right thing and do it, Linux comes usually safe from such things. Windows can, but it's luckier.

We will first see what you can do *before* the problem to ease the future recovery and what you must do *after* to recover. There is little to do to prevent from erasing a disk, usually this is done by automatic Windows or Linux−install ill behaved programs or users mistakes − nothing can be done to prevent this except care, but you are already careful, isn't it ?

It can also be done by the use of MSDOS/Windows fdisk. Avoid it as most as you can, but you probably can't.

I have done this many times, on my computer and on others' guy computers and restored Linux most of the time and Windows sometimes. I wish you luck !

1.2. What to do right now ?

If you don't have a problem yet, if you read this by curiosity or just seeking information and you are on a running Linux system, do immediately the following :

- open a root terminal or xterm ;
- key in /sbin/fdisk −l (that last character being l for Lima). Then do fdisk −u −l ;

You will be gratified by a list of all current partitions on all disks presents on your computer. The second one gives the listing in sectors units in place of cylinder and this is sometime necessary.

- Write this back on paper (or do /sbin/fdisk −l | lpr and /sbin/fdisk −u −l | lpr to print it) and save it in a safe place for future use.

If you are not the system administrator, you should not be concerned by the problem and can stop reading this.

1.3. Legal stuff

 You may do (almost) whatever you want with it. I don't care. Just be sure to keep my name intact. I just hope it becomes useful for any person in the Linux community.

In fact I would like to use a more standard LGPL or like licence, but I'm bound by the initial author license and I can't join him. If you know him or any way to contact him, let me know.

I am not responsible of any damage on any computer as a result of anyone reading this HOWTO. If you do any damage, _it is YOUR fault, NOT MINE_! Be careful when partitioning disks, and don't make any mistakes, because it can be fatal! Backup all your important data and check that everything you do is correct! What is described here worked on my computer, but it may or may not work on your computer. Although it should work for everyone, I can't guarantee anything. This is the last warning you get: BACKUP IMPORTANT DATA! Or, to put it short: Use at your own risk!

1.4. What do I must know right now?

You *must* know that in case of any major problem with your hard disk, you *must* stop using it at all in *write* mode, at least the time necessary to understand what happens. Information there is very volatile...

If ever, one morning, awaking, you computer say can't load, no system installed , you *must not begin reinstalling all the stuff*.

If you have Windows installed, I can't promise you can recover your data, but it's likely you will recover all your Linux stuff, provided it's not located too low in the disk structure. This is because some Windows viruses erases the very first disk cylinder, whatever is on. However I didn't ever experiment such virus and can't say for sure. Try recovering anyway.

You must also know that I give you all this information only for this −information purpose. Neither I nor any other people but you can be held responsible for any problem your data can have using this info. There are too many different systems on the world for anybody being able to promise anything. I can only wish you luck and hope you, like me, will be happy recovering data.

2. Technical info

2.1. Disks

A hard disk is made of sectors numbered from 0 to the max.

dmesg gives, for example:

```
hdb: ST34321A, 4103MB w/128kB Cache, CHS=523/255/63
```

CHS means Cylinders, Heads, Sectors.

523*255*63=8401995 sectors of 512 bytes, thus the 4103 Mbytes. This is only a logical map, it's not necessarily what is written on the disk cover (except for the total size).

The true size of the sectors is of no interest for us given we don't want to modify anything but restore a previous state. For us, the default size given by fdisk is alright.

The size seen by the system is directly dependent of the work of the BIOS (Basic Input/Output System – the PC's ROM). The mode of the hard disk indicated in the BIOS is essential. On a new disk, it's better to use BIOS automatic hard disk recognition and say yes . Anyway, any modification at this level may destroy all the data of the disk, so don't play with this without essential reason.

This is probably what your disk use, so don't be afraid.

2.2. Partitions

Disk are now huge, 40 Gb are not rare, so that it's not really handy to have all this stuff packed in only one part. Only Windows do so, and if you use Linux, may be it's because you are aware of how inefficient the other is.

So a hard disk is usually cut in some pieces called partitions (see the partition HOWTO for details, also read /usr/share/doc/package/util/README.fdisk)

Let's get a look at (part of) my own print of fdisk –l :

```
Disk /dev/hdb: 255 heads, 63 sectors, 523 cylinders
Units = cylinders of 16065 * 512 bytes
Device Boot Start End Blocks Id System
/dev/hdb1 1 153 1228941 83 Linux
/dev/hdb2 154 166 104422+ 82 Linux swap
/dev/hdb3 * 167 291 1004062+ 83 Linux
/dev/hdb4 295 523 1839442+ 5 Extended
/dev/hdb5 295 422 1028128+ 83 Linux
/dev/hdb6 423 523 811251 6 FAT16
```

This is my second hard disk, tied to guesses and tries (the first is too simple to be interesting).

/dev/hdb is my second ide disk (slave on the primary interface),

/dev/hdb1 is the first primary partition, running from the first (1) block to the block 153.

There can be four of such primary partitions. If one wants more than 4, one of them must be an extended one (not necessarily the fourth) and all other partitions are named logical and are located *inside* the extended one. Notice that partition number 5 and partition number 4 have the same beginning. Number five is logical, number 4 extended. Logicals begin always at 5, even if there are not 4 primary ones.

Here the fdisk −u −l listing of an other disk:

```
Disque /dev/hda : 240 têtes, 63 secteurs, 2584
cylindres Unités = secteurs sur 1 * 512 octets
Périphérique Amorce Début Fin Blocs Id Système
/dev/hda1 * 63 10357199 5178568+ c Win95 FAT32 (LBA)
/dev/hda2 15452640 39070079 11808720 83 Linux
/dev/hda3 10357200 15150239 2396520 f Win95 Etdue (LBA)
/dev/hda4 15150240 15452639 151200 84 Lecteur C: caché OS/2
/dev/hda5 10357263 10463039 52888+ 83 Linux
/dev/hda6 10463103 10780559 158728+ 82 Echange Linux
/dev/hda7 10780623 15150239 2184808+ 6 FAT16
Les entrées de la table de partitions ne suivent pas l'ordre du disque.
```

Don't worry for the french part, I'm french... look at your own disk listing. Of course numbers are bigger.

2.3. Why is there a problem ?

Problem is all the installed Operating Systems must share the disks and as at start the BIOS only scan the first one, there must be a so called partition table in the very beginning of this disk. This partition table is located in the Master Boot Record (MBR) side by side with the boot loader.

Any misuse of the MBR by any of the OS's leads to problems. When trying to install any system, yes answer at a question like automatic partitioning? is likely to give problems... This is specially true with Windows, especially with custom windows installation made by special makes PC's (when no true Windows CD is included, as with many laptops). But it's also true with new smart (not so smart!) Linux installation programs given with some recent distributions.

3. Solving the problem

Please, beware! following the explanations given here will lead you to turn back to a previous system, loosing all your recent installed one, if any! You must choose...

3.1. The simpler case

All is simple if you have at hand :

- a disk (floppy or CD) able to start Linux by itself with fdisk available − most rescue disks of any distribution can do that,
- a paper with the fdisk −l and fdisk −u −l content written down.

It's enough to

1. start Linux ;
2. start fdisk /dev/hda (or whatever is the disk to rescue) ;
3. use fdisk to delete (d option) all the existing partitions on the damaged disk ;
4. use fdisk to create all the primary (1−4) partition mentioned on the paper ;
5. give them the appropriate tag (t option) : 82 is for Linux swap, 83 for Linux main (L gives you the list), 5 is extended and must be done before creating logical partitions, c is Windows Fat 32 and f Windows extended when 6 is Windows Fat 16.
6. create any logical partition.

On my SuSE installation and anytime I had to do this for other peoples, this gives a good result.

However I was said that some fdisk may cut partitions on a sector basis, not cylinder. So the fdisk −u −l version of the paper.

For using the fdisk −u −l listing one must start fdisk −u :−). On my opinion, using sector limit is a very bad idea, but it may have a real use I'm not aware of. Problem is that with cylinder limit it's easy to guess even if you don't have paper. With sector one, there a many more guesses...

fdisk is a small and very smart programs. There are many other makes of fdisk, but I always prefer the bare bone one (I speak of Linux ones, of course, not the others...).

Be aware that fdisk doesn't write anything to disk before you hit w and return. In case you fear a mistake, hit q (quit) or Ctrl C (^C) to quit safe.

When your new partition table is written, start your Linux. Chance is you can't do that as usual: lilo can have been damaged also and you will need a boot floppy or booting from a CD (choose the option booting the installed partition).

If you use to boot with lilo, as soon as you are logged in as root, key in lilo and hit return to reinstall you favorite boot loader.

Your Linux should be all here, test it. Try also to start Windows if applicable. If you can't, there is a (very little) chance you can read your data from Linux, may be with a raw sector by sector read. If you can identify the disk sectors you data is on, using dd can copy them on a file. This is wise for text only. This recovery is NOT in the scope of this mini−HOWTO.

3.2. A not so simple case

3.2.1. By hand

This is when the previous case can't be used, for lack of fdisk paper or if it won't run for use of an out of date one.

First, be aware that as soon as you don't write to the disk (except with fdisk), you can't erase your data, so that you can use a block by block try. That is you need to know the beginning of the partition to start it. If, say a 153 don't fit, try a 154, and so on.

This can be tiresome, but if you remember approximately the size of the Linux partition, there is a chance to win.

3.2.2. Linux own info and other hacks

3.2.2.1. Kernel

If you just destroyed your own partition table, but have not rebooted Linux: Don't reboot! You can still retrieve the partition information stored in the Kernel:

cat /proc/partitions gives

```
major minor  #blocks  name
    3     0   19535040 hda
    3     1    2096451 hda1
    3     2    4980150 hda2
    3     3          1 hda3      <----- this marks an extended partition
    3     5    4980118 hda5
    3     6    4972086 hda6
```

3.2.2.2. hdparm

```
hdparm −g /dev/hda1
/dev/hda1: geometry     = 2432/255/63, sectors = 4192902, start = 63
```

You'll need to do a few unit conversions. "#blocks" are usually 1K in length. "Sectors" are disk sectors, often 512 bytes. But usually the disk partitioning tools work in units of cylinders (here 255*63=16065 sectors). Using this information you can build a new partition table.

3.2.2.3. I know the start of the partition, but not the end.

If you know the start of a Linux partition, but not the end, you can still mount it, and learn about the structure. Set the partition table start correctly, and set the end to something very large. See if you guessed correctly with:

```
e2fsck −n /dev/hd??
```

You can even mount the partition and check the size:

```
mount −r /dev/hd?? /mnt
df −T
```

This won't directly tell you where the next partition starts, because of rounding issues. But it can help you get close. Be sure to use the "−n" and "−r" flags to treat the system as read−only!!!

3.2.2.4. Other places partition information is stored

Some distributions record partition information in a file. Of course you probably won't be able to get to this file when you need it. But just in case:

```
SuSE: /var/lib/YaST/install.inf
```

(if you are aware of others, please email the maintainer of this document)

3.2.3. gpart

But there is a better way if you can still access the net or have gpart at hand. gpart is available in most distribution, by freshmeat.net or directly at http://www.stud.uni−hannover.de/user/76201/gpart.

gpart − guess PC−type hard disk partitions is the first line of the man page of gpart (man gpart).

gpart tries to guess which partitions are on a hard disk. If the primary partition table has been lost, overwritten or destroyed the partitions still exist on the disk but the operating system cannot access them. This is exactly what we need.

gpart is a very good tool.

The problem is the following: the first block of any partition is marked. But it's never unmarked if not overwritten. So many first partition block are existing on an old disk and gpart tries to do it's best guessing what is the good one. In fact it's not too difficult to try, nothing is written on the disk by gpart.

Here is the result of gpart on the previously seen disk hdb:

```
root@charles:/home/jdd > gpart /dev/hdb
Begin scan...
Possible partition(Linux ext2), size(1200Mb), offset(0Mb)
```

```
Possible partition(Windows NTFS), size(1200Mb), offset(1200Mb)
Possible partition(Linux ext2), size(1004Mb), offset(2402Mb)
Possible partition(Windows NTFS), size(1600Mb), offset(4102Mb)
End scan.
Checking partitions...
* Warning: partition(OS/2 HPFS, NTFS, QNX or Advanced UNIX) ends beyond disk end .
Partition(Linux ext2 filesystem): primary
Partition(OS/2 HPFS, NTFS, QNX or Advanced UNIX): primary
Partition(Linux ext2 filesystem): primary
Partition(OS/2 HPFS, NTFS, QNX or Advanced UNIX): invalid primary
Ok.
Guessed primary partition table:
Primary partition(1)
type: 131(0x83)(Linux ext2 filesystem)
size: 1200mb #s(2457880) s(63−2457942)
chs: (0/1/1)−(152/254/61)d (0/1/1)−(152/254/61)r
Primary partition(2)
type: 007(0x07)(OS/2 HPFS, NTFS, QNX or Advanced UNIX)
size: 1200mb #s(2457880) s(2457944−4915823)
chs: (152/254/63)−(305/253/60)d (152/254/63)−(305/253/60)r
Primary partition(3)
type: 131(0x83)(Linux ext2 filesystem)
size: 1004mb #s(2056256) s(4919781−6976036)
chs: (306/61/49)−(434/60/47)d (306/61/49)−(434/60/47)r
Primary partition(4)
type: 000(0x00)(unused) size: 0mb #s(0) s(0−0) chs: (0/0/0)−(0/0/0)d (0/0/0)−(0/0/0)r
```

As you see, primary partition can be recovered, but for extended ones it's still to be done.

Dos partitions are labeled windows NTFS because they were created while trying to install Windows 2000 (a very awful experience!). The invalid one is, in fact the extended partition.

With this, one can use fdisk and try re−creating the partition table (remember, this is risk−free given the original one is already lost).

gpart is updated on a weekly basis :−) and so new makes may be more powerful than I know.

3.2.3.1. Recovering partitions inside an extended partition

Extended partition information is scattered on the disk, not stored with the primary partition. To recover these often requires more works. The process is:

1. Scan for the start of the first partition (using gpart's −k option);
2. Create a temporary primary partition entry with the true start position and a fake end position (this may drive you to delete an actual primary partition if no one is available − this is risk free if you don't reuse the sectors of the deleted partition);
3. Use e2fsk −n , mount −r , and df to determine the true end point. Write this value down (warning:read the man page for each program mentioned, and use the read−only options, you do not want to write to your disk until all partitions are in the correct place);
4. Repeat this process for each partition to be recovered;
5. Build a complete new correct partition table.

3.2.3.2. If your hard drive has errors

If your hard drive has errors, you may have real trouble mounting, checking or using data (the drive read errors get in the way). Gpart may not even find it. But if you know the start of the partition, you can easily copy the data to a temporary file stored on a different drive. Sectors with read errors will usually be set to zero by this process:

- Copy the partition data to a file. You must know the start block of the partition;

```
dd if=/dev/hd?? of=/tmp/recover_hd?? bs=512 skip=XXXX count=YYY
```

XXX is the sector start and YYY the sector count (can be guessed).

- Mount the file as a loop file system.

```
mount −r −t ext2 −o loop /tmp/recover_hd?? /mnt/recover
```

3.3. The rich man case

Partition Magic is a commercial product, not so cheap given the little use one can have (approx a hundred bucks in France) but with a very high reputation all around there. However I never use it and will not rate it. It's said to be able to do anything with partitions, including restoring them.

Original Ralf partition−rescue mini HOWTO was essentially based around the use of Partition Magic, so I presume it's a very good solution if you have valuable data on your Linux partition and little Linux capability. However there are now very recent makes of Partition Magic and I think it's better for you to read the manual.

4. References

4.1. Authors

The author of this HOWTO is Jean−Daniel Dodin. I can be joined at jdanield@dodin.net or at jdanield@linux−france.org.

My web site is at http://www.dodin.net. Linux pages, who are of most interest for you are found at www.linux−france.org/article/jdanield.

I want to thank Rolf Klausen (− not responding − E−mail: rolfk@romsdal.vgs.no http://www2.romsdal.vgs.no/Ürolfk/) who write the previous partition−rescue mini HOWTO even if I rewrite it almost entirely, he had first the good idea. In 2000 He is no more responding to his e−mail. If anybody knows how to contact him, please let me know.

Every other member of the Linux community and everybody who supports Linux and writes documentation and programs for Linux and all the authors of the LDP and virtually any person involved in anything which has to do with Linux. Particularly Linus B. Torvalds − he is _The King_ !!!

I want also to thank Michail Brzitwa <mb@ichabod.han.de> for writing gpart !

Bryce Nesbitt <bryce at obviously dot com> did a very good job, Linux own info is from him as are some minor enhancements.

4.2. Most recent version

The most recent version of this mini−HOWTO shall be found on my Linux web site, at www.linux−france.org/article/jdanield/