Go to the first, previous, next, last section, table of contents.

---

2. Using Parted

2.1 Partitioning Concepts

Unfortunately, partitioning your disk is rather complicated. This is because there are interactions between many different systems that need to be taken into consideration:

- The BIOS or firmware - the program that is built into a ROM chip inside your computer, that does memory checks, etc. You can not (easily) change programs in this system. Examples of BIOS or firmware programs: AmiBIOS, Award, Phoenix, OpenFirmware. You will only have one of these programs.
- The boot loader - the program that allows you to select which operating system you want to use, and loads that operating system. Examples: LILO, GRUB, Yaboot, Quik. You may have more than one boot loader installed, especially if you have more than one type of operating system installed.
- The operating system (at the moment, this must be GNU/Linux) that runs Parted, and the other operating systems that you use.
- The file system types - the way the data stored on partitions. Examples of these are: ext2, fat32, hfs, reiserfs. You will often have partitions of of different file system types.

Parted suppports many combinations of BIOS, boot loader, operating system, and file systems, and will support more in the future. To further understand the roles of each, please see section 3. BIOSes and Firmware, section 4. Boot Loaders, section 5. Operating Systems, and section 6. File Systems supported by Parted.

This chapter describes how to use Parted, which is largely the same, no matter what systems you are using. You should read this chapter, then each of chapters on BIOSes, boot loaders, operating systems, and file systems. However, you only need to read the sections that are relevant to you. For example, if you are only using LILO as your boot loader, then you only need to read the introduction, and section 4.1 LILO: a bootloader for the Linux kernel.

2.2 Using GNU Parted

Parted has two modes: command line and interactive. Parted should always be started with:

```
# parted device
```

where *device* is the hard disk device to edit. (If you're lazy, Parted will attempt to guess which device you want.)

In command line mode, this is followed by one or more commands. For example:

```
# parted /dev/sda resize 1 52 104 mkfs 2 fat16
```

Options (like `--help`) can only be specified on the command line.

In interactive mode, commands are entered one at a time at a prompt, and modify the disk immediately. For example:

```
(parted) resize 1 52.0005 104.5
(parted) mkfs 2 fat16
```

Unambiguous abbreviations are allowed. For example, you can type "p" instead of "print", and "re" instead of "resize". Commands can be typed in, either in English, or your native language (if your language has been

translated). This may create ambiguities.

Also note that you can specify decimal places in the numbers corresponding to partition locations (in megabytes). Negative numbers count from the end of the disk, with "-0" being the end of the disk.

If you don't give a parameter to a command, Parted will ask you for it. For example:

```
(parted) resize 1
Start? 0
End? 400
```

Parted will always warn you before doing something that is potentially dangerous, unless it's something that's obviosuly dangerous (i.e. rm, mklabel, mkfs) For example, if you attempt to shrink a partition "too much" (i.e. by more than the free space available), Parted will automatically resize to the minimum it can without losing data. If this minimum is significantly different, it will warn you that it's doing something significanlty different to what you asked. Since many partitioning systems have complicated constraints, Parted will usually do something slightly different to what you asked. (For example, create a partitiong starting at 10.352, not 10.4)

2.3 Command Line Options

When invoked from the command line, parted supports the following syntax:

```
# parted [option] device [command [argument]]
```

Available options and commands follow. For detailed explanations of the use of parted commands, see section 2.4 Parted Session Commands. Options begin with a hyphen, commands do not:

Options:

`-h'
`--help'
: display a help message

`-i'
`--interactive'
: where necessary, prompt for user intervention

`-s'
`--script'
: never prompt for user intervention

`-v'
`--version'
: display the version

2.4 Parted Session Commands

GNU Parted provides the following commands:

2.4.1 check

Command: **check** *minor*

Checks if the file system on partition *minor* has any errors.

Example:

```
(parted) check 1
```

Check the file system on partition 1.

2.4.2 cp

Command: **cp** *[from-device] from-minor to-minor*

Copies the file system on the partition *from-minor* to partition *to-minor*, deleting the original contents of the destination partition.

An optional device parameter, *from-device* can be given, which specifies which device the source partition is on.

Supported file systems:

- ext2, ext3 (provided the destination partition is larger than the source partition)
- fat16, fat32
- linux-swap (equivalent to mkswap on destination partition)
- reiserfs (if libreiserfs is installed)

Example:

```
(parted) cp /dev/hdb 2 3
```

Copy partition 2 of `/dev/hdb' (i.e. `/dev/hdb2') to partition on 3, on the device Parted was loaded with, destroying the original contents of partition 3.

2.4.3 help

Command: **help** *[command]*

Prints general help, or help on *command*.

Example:

```
(parted) help resize
```

Print help for the resize command.

2.4.4 mklabel

Command: **mklabel** *label-type*

Creates a new disk label, of type *label-type*. The new disk label will have no partitions. This command (normally) won't technically destroy your data, but it will make it will make it basically unusable, and you will need to use the rescue command (see section 9. Related Software and Info) to recover any partitions. Gpart only works for msdos disk labels (AFAIK), but is much better than parted at recovering partitions. Parted works on all partition tables. (1)

label-type must be one of these supported disk labels:

- bsd
- loop (raw disk access)
- gpt
- mac
- msdos
- pc98
- sun

Example:

```
(parted) mklabel msdos
```

Create an msdos style disklabel.

2.4.5 mkfs

Command: **mkfs** *minor fs-type*

Makes a file system *fs-type* on partition *minor*, destroying all data that resides on that partition.

Supported file systems:

- ext2
- mips
- fat16
- fat32
- linux-swap
- reiserfs (if libreiserfs is installed)

Example:

```
(parted) mkfs 2 fat32
```

Make a *fat32* file system on partition 2.

2.4.6 mkpart

Command: **mkpart** *part-type [fs-type] start end*

Creates a new partition, *without* creating a new file system on that partition. This is useful for creating partitions for file systems (or LVM, etc.) that Parted doesn't support. You may specify a file system type, to set the appropriate partition code in the partition table for the new partition. *fs-type* is required for data partitions (i.e., non-extended partitions). *start* and *end* are the offset from the beginning of the disk, that is, the "distance" from the start of the disk.

part-type is one of: primary, extended, logical. Extended and logical are only used for msdos and mips disk labels.

fs-type must be on of these supported file systems:

- ext2
- fat32
- fat16
- HFS
- linux-swap
- NTFS
- reiserfs
- ufs

Example:

```
(parted) mkpart logical 0.0 692.1
```

Create a logical partition that will contain an ext2 filesystem. The partition will start at the beginning of the disk, and end 692.1 megabytes into the disk.

2.4.7 mkpartfs

Command: **mkpartfs** *part-type fs-type start end*

Creates a new partition of type *part-type* with a new file system of type *fs-type* on it. The new partition will start *start* megabytes, and end *end* megabytes from the beginning of the disk. Do not use this command to recover a deleted partition (use mkpart instead).

part-type is one of: primary, extended, logical. Extended and logical are only used for msdos and mips disk labels.

fs-type must be one of these supported file systems:

- ext2
- fat32
- fat16
- linux-swap
- reiserfs (if libreiserfs is installed)

Example:

```
(parted) mkpartfs logical ext2 440 670
```

Make a logical partition and write an ext2 file system, starting 440 megabytes and ending 670 megabytes from the beginning of the disk.

2.4.8 move

Command: **move** *minor start [end]*

Moves partition on the disk, by moving its beginning to *start*. Note: move never changes the minor number.

If no *end* is given, the partition's size remains the same.

Supported file systems:

- ext2, ext3 (provided the destination partition is larger than the source partition)
- fat32
- fat16
- linux-swap
- reiserfs (if libreiserfs is installed)

Example:

```
(parted) move 2 150
```

Move partition with minor number 2 so that it begins 150 megabytes from the start of the disk.

2.4.9 name

Command: **name** *minor name*

Sets the name for the partition *minor* (Mac and PC98 only). The name can be placed in quotes.

Example:

```
(parted) name 2 'Secret Documents'
```

Set the name of partition 2 to `Secret Documents'.

2.4.10 print

Command: **print**

Displays the partition table on the device parted is editing.

Example:

```
(parted) print
Disk geometry for /dev/hda: 0.000-2445.679 megabytes
Disk label type: msdos
Minor    Start       End     Type      Filesystem  Flags
1          0.031    945.000  primary   FAT         boot, lba
2        945.000   2358.562  primary   ext2
3       2358.562   2445.187  primary   linux-swap
```

2.4.11 quit

Command: **quit**

Quits Parted.

It is only after Parted exits that the Linux kernel knows about the changes Parted has made to the disks. However, the changes caused by typing your commands will *probably* be made to the disk immediately after typing a command. However, Linux's cache, and the disk's hardware cache may delay this.

2.4.12 rescue

Command: **rescue** *start end*
rescue a lost partition that used to be about *start* and *end*

Looks for file system signatures around *start* and *end*. If one is found, it will ask you if you want to create a partition for it. This is useful if you accidently deleted a partition with parted's rm command, for example.

Example:

```
(parted) print
Disk geometry for /dev/hdc: 0.000-8063.507 megabytes
Disk label type: msdos
Minor    Start       End     Type      Filesystem  Flags
1          0.031   8056.032  primary   ext3
(parted) rm
Partition number? 1
(parted) print
Disk geometry for /dev/hdc: 0.000-8063.507 megabytes
Disk label type: msdos
Minor    Start       End     Type      Filesystem  Flags
```

OUCH! We deleted our ext3 partition!!! Parted comes to the rescue...

```
(parted) rescue
Start? 0
End? 8056
Information: A ext3 primary partition was found at 0.031Mb ->
8056.030Mb.  Do you want to add it to the partition table?
Yes/No/Cancel? y
(parted) print
Disk geometry for /dev/hdc: 0.000-8063.507 megabytes
Disk label type: msdos
Minor    Start       End     Type      Filesystem  Flags
1          0.031   8056.032  primary   ext3
```

It's back! :)

2.4.13 resize

Command: **resize** *minor start end*

Resizes the partition with number *minor*. The partition will start *start* from the beginning of the disk, and end *end* from the beginning of the disk. resize never changes the minor number. Extended partitions can be resized, so long as the new extended partition completely contains all logical partitions.

Note that Parted does not require a file system to be "defragged" (Parted can safely move data around if necessary). It's a waste of time defragging. Don't bother!

Supported file systems:

- ext2, ext3 - restriction: the new *start* must be the same as the old *start*.
- fat16, fat32
- linux-swap
- reiserfs (if libreiserfs is installed)

Example:

```
(parted) resize 3 200 850
```

Resize partition 3, so that it begins 200 megabytes and ends 850 megabytes from the beginning of the disk.

2.4.14 rm

Command: **rm** *minor*

Removes the partition with number *minor*. If you accidently delete a partition with this command, use mkpart (*not* mkpartfs) to recover it. Also, you can use the gpart program (see section 9. Related Software and Info) to recover damaged disk labels.

Note for msdos disk labels: if you delete a logical partition, all logical partitions with a larger minor number will be renumbered. For example, if you delete a logical partition with a minor number of 6, then logical partitions that were number 7, 8 and 9 would be renumbered to 6, 7 and 8 respectively. This means, for example, that you have to update `/etc/fstab' on GNU/Linux systems.

Example:

```
(parted) rm 3
```

Remove partition 3.

2.4.15 select

Command: **select** *device*

Selects the device, *device*, for Parted to edit. The device will usually be a Linux hard disk device, or, if direct access to a file system is required -- a partition, software RAID device, or LVM logical volume.

Example:

```
(parted) select /dev/hdb
```

Select `/dev/hdb' (the slave device on the first ide controller on Linux) as the device to edit.

2.4.16 set

Command: **set** *minor flag state*

Changes a flag on the partition with number *minor*. A flag can be either "on" or "off". Some or all of these flags will be available, depending on what disk label you are using:

`boot'
(Mac, MSDOS, PC98) - should be enabled if you want to boot off the partition. The semantics vary between disk labels. For MSDOS disk labels, only one partition can be bootable. If you are installing LILO on a partition (see section 4.1 LILO: a bootloader for the Linux kernel), then that partition must be bootable. For PC98 disk labels, all ext2 partitions must be bootable (this is enforced by Parted).

`lba'
(MSDOS) - this flag can be enabled, to tell MS DOS, MS Windows 9x and MS Windows ME based operating systems to use Linear (LBA) mode.

`root'
(Mac) - this flag should be enabled if the partition is the root device to be used by Linux.

`swap'
(Mac) - this flag should be enabled if the partition is the swap device to be used by Linux.

`hidden'
(MSDOS, PC98) - this flag can be enabled to hide partitions from Microsoft operating systems.

`raid'
(MSDOS) - this flag can be enabled to tell linux the partition is a software RAID partition See section 7. LVM and RAID.

`LVM'
(MSDOS) - this flag can be enabled to tell linux the partition is a physical volume.

The print command displays all enabled flags for each partition.

Example:

```
(parted) set 1 boot on
```

Set the `boot' flag on partition 1.

2.5 Example Parted Sessions

These examples attempt to cover the most common circumstances, with the exception of disk imaging, which is covered in section 8. Disk Imaging.

2.5.1 Example: Growing a partition into unused space

Suppose your disk layout looks like this:

```
(parted) print
Disk geometry for /dev/hda: 0.000-1000.000 megabytes
Disk label type: msdos
Minor    Start       End     Type      Filesystem  Flags
1          0.063   500.000  primary   ext2
2        500.000   625.000  primary   linux-swap
```

There is 375 Mb of free space at the end of the disk (after partition 2). Partition 1 has an ext2 file system, which is the root device. Partition 2 is a swap device.

Suppose you wanted to use the free space at the end of the disk for the file system on partition 1. You could do the following:

1. These steps will modify both the root file system on partition 1, and the swap device on partition 2. Therefore, you shouldn't be using either partitions. You should probably use a Parted boot disk. See section 1.6 Using a Parted Boot Disk. From the boot disk, run Parted:

   ```
   # parted /dev/hda
   ```

2. Remove partition 2 (the swap partition). Normally, you wouldn't want to delete a partition with data on it. However, a swap partition doesn't contain data when it isn't "swapped on" (mounted), so you can remove it, and create a replacement swap partition later.

```
(parted) rm 2
```

3. Create the new swap partition at the end of the disk:

```
(parted) mkpartfs primary linux-swap 875 999.9
(parted) print
Disk geometry for /dev/hda: 0.000-1000.000 megabytes
Disk label type: msdos
Minor    Start       End     Type      Filesystem  Flags
1          0.063    500.000  primary   ext2
2        875.000   1000.000  primary   linux-swap
```

4. Grow partition 1, into the adjacent free space:

```
(parted) resize 1 0.063 874.9
```

All done!

```
(parted) print
Disk geometry for /dev/hda: 0.000-1000.000 megabytes
Disk label type: msdos
Minor    Start       End     Type      Filesystem  Flags
1          0.063    874.999  primary   ext2
2        875.000   1000.000  primary   linux-swap
```

2.5.2 Example: Resizing an ext2 partition on a crowded disk.

Suppose your disk layout looks like this:

```
(parted) print
Disk geometry for /dev/hda: 0-8063.5 megabytes
Disk label type: msdos
Minor   Start     End    Type            Filesystem     Flags
1           0.0    23.5  primary         ext2           boot
2          23.5  8056.0  extended
5          23.6  3545.6  logical         ext2
6        3545.6  7067.7  logical         ext2
7        7067.7  7326.5  logical         ext2
8        7326.5  7585.4  logical         ext2
9        7585.4  7844.2  logical         linux-swap

$ df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda8             251M   31M  207M  13% /
/dev/hda1              23M  2.4M   19M  11% /boot
/dev/hda5             3.4G  577M  2.7G  18% /usr
/dev/hda6             3.4G  289M  2.9G   9% /home
/dev/hda7             251M   12M  226M   5% /var
```

Suppose you wanted to increase the `/var' partition (`/dev/hda7') to 1GB, using some space from `/home' (`/dev/hda6').

To resize a partition with Parted, you use the resize command:

```
(parted) resize partition_number new start new end
```

new start must be the same as the old start for ext2 partitions (unfortunately). So this process is going to be rather complicated. It *is* possible, though. (2)

1. Shrink the `/home' partition (`/dev/hda6') by 500MB:

```
# parted /dev/hda
(parted) resize 6 3545.6 6200
```

2. Make a new partition in its place. This is where `/var' will be, eventually. This new partition will be numbered 10.

   ```
   (parted) mkpartfs logical ext2 6200 7067.7
   ```

3. Copy the old `/var' partition (`/dev/hda7') to the new one (`/dev/hda10').

   ```
   (parted) cp 7 10
   ```

4. Delete the old `/var'.

   ```
   (parted) rm 7
   ```

   At this point: all logical partitions greater than 7 just changed number. So 8, 9 and 10 become 7, 8 and 9 respectively. This renumbering won't take place while any partitions are mounted on that disk (this will happen when you reboot). That's what that warning message is talking about. So you should *never* attempt to mount a file system touched by Parted (resized or created by Parted), before rebooting, if you get this message.

5. Resize the new `/var' partition (now numbered 9), adding the space from the old `/var' partition:

   ```
   (parted) resize 9 6200 7326.5
   (parted) quit
   Warning: The kernel was unable to re-read the partition table on
   /dev/hda (Device or resource busy).  This means Linux knows nothing
   about any modifications you made.  You should reboot your computer
   before doing anything with /dev/hda.
   ```

6. Since the partition numbers have changed, `/etc/fstab' must be updated. This can be done before rebooting, because the root device wasn't touched by Parted. (If you want to use Parted to do something to the root device, you need to use the boot disk). If the old `/etc/fstab' looks like this:

   ```
   /dev/hda8       /             ext2    defaults         1 1
   /dev/hda1       /boot         ext2    defaults         1 2
   /dev/hda6       /home         ext2    grpquota,usrquota  0  2
   /dev/cdrom      /mnt/cdrom    iso9660 noauto,owner,ro 0 0
   /dev/hda5       /usr          ext2    defaults         1 2
   /dev/hda7       /var          ext2    grpquota,usrquota  0  2
   /dev/fd0        /mnt/floppy   auto    noauto,owner     0 0
   none            /proc         proc    defaults         0 0
   none            /dev/pts      devpts  gid=5,mode=620   0 0
   /dev/hda9       swap          swap    defaults         0 0
   ```

   A few lines need to be changed:
   - `/var' is now /dev/hda9 (because we copied it to a new partition)
   - `/dev/hda8' (the root device) has been renumbered to `/dev/hda7'
   - `/dev/hda9' (the swap device) has been renumbered to `/dev/hda8'

   The new `/etc/fstab' looks like this:

   ```
   /dev/hda7       /             ext2    defaults         1 1
   /dev/hda1       /boot         ext2    defaults         1 2
   /dev/hda6       /home         ext2    grpquota,usrquota  0  2
   /dev/cdrom      /mnt/cdrom    iso9660 noauto,owner,ro 0 0
   /dev/hda5       /usr          ext2    defaults         1 2
   /dev/hda9       /var          ext2    grpquota,usrquota  0  2
   /dev/fd0        /mnt/floppy   auto    noauto,owner     0 0
   none            /proc         proc    defaults         0 0
   none            /dev/pts      devpts  gid=5,mode=620   0 0
   /dev/hda8       swap          swap    defaults         0 0
   ```

7. Reboot. That's it!

Go to the first, previous, next, last section, table of contents.