NAME

gpart − guess PC-type hard disk partitions

SYNOPSIS

gpart [options] *device*

Options: [-b <backup MBR>][-C c,h,s][-c][-d][-E][-e][-f] [-g][-h][-i][-K <last-sector>][-k <# of sectors>] [-L] [-l <log file>][-n <increment>] [-q][-s <sector-size>] [-t <module-name>][-V][-v] [-W <device>][-w <module-name, weight>]

DESCRIPTION

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.

gpart ignores the primary partition table and scans the disk (or disk image, file) sector after sector for several filesystem/partition types. It does so by "asking" filesystem recognition modules if they think a given sequence of sectors resembles the beginning of a filesystem or partition type. Currently the following filesystem types are known to **gpart** (listed by module names) :

beos	BeOS filesystem type.
bsddl	FreeBSD/NetBSD/386BSD disklabel sub-partitioning scheme used on Intel platforms.
ext2	Linux second extended filesystem.
fat	MS-DOS FAT12/16/32 "filesystems".
hpfs	IBM OS/2 High Performance filesystem.
hmlvm	Linux LVM physical volumes (LVM by Heinz Mauelshagen).
lswap	Linux swap partitions (versions 0 and 1).
minix	The Minix operating system filesystem type.
ntfs	MS Windows NT/2000 filesystem.
qnx4	QNX 4.x filesystem.
reiserfs	The Reiser filesystem (version 3.5.X, X > 11, 3.6.X).
s86dl	Sun Solaris on Intel platforms uses a sub-partitioning scheme on PC hard disks similar to the BSD disklabels.
xfs	Silicon Graphic’s journalling filesystem for Linux.

More filesystem guessing modules can be added at runtime (see the *-t* option). Please consult the **gpart** README file for detailed explanations on how to create guessing modules. All modules are accompanied by a guessing weight factor which denotes how "educated" their guesses are compared to other modules. This weight can be changed if a certain module keeps on mis-identifying a partition.

Naturally only partitions which have been formatted in some way can be recognized. If the type of a partition entry in the primary partition table is changed from x to y while the filesystem is still of type x, **gpart** will also still guess a type x.

No checks are performed whether a found filesystem is clean or even consistent/mountable, so it is quite possible that **gpart** may identify partitions which existed prior to the current partitioning scheme of the disk. Especially on large disks old file system headers/superblocks may be present a long time until they are finally overwritten with user data.

It should be stressed that **gpart** does a very heuristic job, never believe its output without any plausability checks. It can be easily right in its guesswork but it can also be terribly wrong. You have been warned.

After having found a list of possible partition types, the list is checked for consistency. For example, a partition which overlaps with the previous one will be discarded. All remaining partitions are labelled with one of the following attributes: "primary", "logical", "orphaned" or "invalid".

A partition labelled "orphaned" is a logical partition which seems ok but is missed in the chain of logical partitions. This may occur if a logical partition is deleted from the extended partition table without overwriting the actual disk space.

An "invalid" partition is one that cannot be accepted because of various reasons. If a consistent primary partition table was created in this process it is printed and can be written to a file or device.

EXTENDED PARTITIONS

If the disk/file to be examined consists of primary partitions only, **gpart** has quite a good chance to identify them. Extended partitions on the other hand can result in a lot of problems.

Extended partitions are realized as a linked list of extended partition tables, each of which include an entry pointing to a logical partition. The size of an extended partition depends on where the last logical partition ends. This means that extended partitions may include "holes", unallocated disk space which should only be assigned to logical, not primary partitions.

gpart tries to do its best to check a found chain of logical partitions but there are very many possible points of failure. If "good" fdisk programs are used to create extended partitions, the resulting tables consist of a zeroed boot record and the four partition entries of which at least two should be marked unused. Unfortunately e.g. the fdisk program shipped with Windows NT does not seem to zero out the boot record area so **gpart** has to be overly tolerant in recognizing extended partition tables. This tolerance may result in quite stupid guesses.

DISK TRANSFERS

If you want to investigate hard disks from other systems you should note down the geometry (number of cylinders, heads per cylinder and sectors per head) used for that disk, and tell **gpart** about this geometry.

Investigating disks from machines with a different endianness than the scanning one has not been tested at all, and is currently not recommended.

LARGE DISKS

gpart relies on the OS reporting the correct disk geometry. Unfortunately sometimes the OS may report a geometry smaller the the actual one (e.g. disks with more than 1024 or 16384 cylinder).

gpart checks if guessed partitions extend beyond the disk size and marks those "invalid", but may be mistaken in case the disk size is calculated from an incorrect geometry. For instance if a disk with the geometry 1028/255/63 should be scanned, and the OS reports 1024/255/63 **gpart** should be called like

```
gpart -C 1028,255,63 <other options> <device>
```

PRECAUTIONS

gpart may be of some help when the primary partition table was lost or destroyed but it can under **no** circumstances replace proper disk/partition table backups. To save the master boot record (MBR) including the primary partition table to a file type

```
dd if=/dev/hda of=mbr bs=512 count=1
```

exchanging /dev/hda with the block device name of the disk in question. This should be done for all disks in the system. To restore the primary partition table without overwriting the MBR type

dd if=mbr of=/dev/hda bs=1 count=64 skip=446 seek=446

Warning: make sure that all parameters are typed as shown and that the disk device is correct. Failing to do so may result in severe filesystem corruption. The saved file should be stored in a safe place like a floppy disk.

OPTIONS

-b backupfile
: If the guessed primary partition table seems consistent and should be written (see the *-W* option) backup the current MBR into the specified file.

-C c,h,s
: Set the disk geometry (cylinders, heads, sectors) for the scan. This is useful if a disk should be scanned which was partitioned using a different geometry, if the *device* is a disk-image or if the disk geometry cannot be retrieved through the PCs BIOS. No spaces are allowed between the numbers, unless all three are enclosed in quotes.

-c
: Check/compare mode (implies the *-q* quiet option). After the scan is done, the resulting primary partition table is compared to the existing one. The return code of **gpart** then contains the number of differences (0 if they are identical except for the boot/active flag which cannot be guessed). This option has no effect if *-d* is given on the command line.

-d
: Do not start the guessing loop. Useful if the partition table should be printed (in combination with the *-v* option) without actually scanning for partitions.

-E
: Do not try to identify extended partition tables. If there are extended partitions on the given device **gpart** will most certainly complain about too many primary partitions because there can be only four primary partitions. Existing logical partitions will be listed as primary ones.

-e
: Do not skip disk read errors. If this option is given, and short disk reads or general disk read errors (EIO) are encountered, **gpart** will exit. If not given, the program tries to continue.

-f
: Full scan. When a possible partition is found, **gpart** normally skips all sectors this entry seems to occupy and continues the scan from the end of the last possible partition. The disk scan can take quite a while if this option is given, be patient.

-g
: Do not try to get the disk geometry from the OS. If the *device* is no block or character device but a plain file this option should be supplied. If the file to be scanned is an image of a disk, the geometry can be given by the *-C* option.

-h
: Show some help.

-i
: Run interactively. Each time a possible partition is identified the user is asked for confirmation.

-K last sector
: Scan only up to the given sector or the end of the file or device whichever comes first.

-k sectors
: Skip given number of sectors before the scan. Potentially useful if a partition is looked for at the end of a large disk.

-L
: List available filesystem/partition type modules and their weights, then exit.

-l logfile
: Log output to the given file (even if *-q* was supplied).

-n increment
: Scan increment: number of sectors or "s" for single sector increment, "h" for an increment of sectors per head (depends on geometry) or "c" for cylinder increment.

  The increment also influences the condition where extended partition tables are searched: if the scan increment is "s" (i.e. 1) extended partition tables are required to be on a head boundary, otherwise they must be on a cylinder boundary.

If the disk geometry could not be retrieved and no geometry was given on the command line, the default increment is one sector.

-q Quiet/no output mode. However if a logfile was specified (see *-l* option) all output is written to that file. This option overrides the *-i* interactive mode.

-s sector size
Preset medium sector size. **gpart** tries to find out the sector size but may fail in doing so. Probed sector sizes are 2^i, i=9..14 (512 to 16384 bytes). The default medium sector size is 512 bytes.

-t module name
Plug in another guessing module. The module to be dynamically linked in must be a shared object file named "gm_<modname>.so".

-V Show version number.

-v Be verbose. This option can be given more than once resulting in quite a lot of information.

-W device
Write partition table. If a consistent primary partition table has been guessed it can be written to the specified file or device. The supplied device can be the same as the scanned one.

Additionally the guessed partition entries can be edited. No checks are performed on the entered values, thus the resulting table is allowed to be highly inconsistent. Please beware.

Warning: The guessed partition table should be checked very carefully before writing it back. You can always write the guessed partition table into a plain file and write this into sector 0 using **dd**(1) (see section PRECAUTIONS above).

-w module name,weight
Put the given module at the head of the module chain and assign a new weight to that module. All modules are given an initial weight of 1.0. Again no spaces are allowed.

Default settings are "-n h".

EXAMPLES

− To scan the first IDE hard disk under Linux using default settings type

```
gpart /dev/hda
```

− To print the primary partition table of the third IDE drive without starting the scan loop in FreeBSD type

```
gpart -vvd /dev/wd2
```

− If **lilo(8)** was installed in the master boot record (MBR) of a hard disk it saves the contents of the first sector in a file called /boot/boot.<major/minor>. To list the partitions contained in such a file type e.g.

```
gpart -vdg /boot/boot.0300
```

If the partition table contains an extended partition, **gpart** will complain about invalid extended partition tables because the extended entry points to sectors not within that file.

− Usually the first primary partition starts on the second head. If **gpart** cannot identify the first partition properly this may not be the case. **gpart** can be told to start the scan directly from sector one of the disk, using the sector-wise scan mode:

gpart -k 1 -n s /dev/hdb

− Suppose **gpart** identifies an NTFS partition as FAT on a certain disk. In this situation the "ntfs" module should be made the first module to be probed and given a weight higher than the usual weight of 1.0:

gpart -w ntfs,1.5 /dev/hdb

To list the available modules and their weights use the *-L* option.

− After having checked the output of **gpart** at least thrice, the primary partition table can be written back to the device this way:

gpart -W /dev/sdc /dev/sdc

This of course may be extremely dangerous to your health and social security, so beware.

− A hard disk with 63 sectors per head is scanned in steps of 63 sectors. To perform the scan on every second head while skipping the first 1008 sectors type

gpart -k 1008 -n 126 /dev/sda

− If you want to see how easily **gpart** can be mislead, and how many probable partition starts are on a disk, search the whole disk really sector by sector, writing all output to a logfile:

gpart -vvfn s -ql /tmp/gpart.log /dev/sd2 &

Usually **gpart** will not be able to produce an educated guess of the primary partition table in this mode. The logfile however may contain enough hints to manually reconstruct the partition table.

FILES

*/dev/**
: Hard disk block devices. The naming scheme of hard disk block devices is OS dependent, consult your system manuals for more information.

DIAGNOSTICS

There are many error message types, all of them should be self-explanatory. Complain if they are not.

BUGS

gpart is beta software, so expect buggy behaviour.

− **gpart** only accepts extended partition links with one logical partition. There may be **fdisk** variants out there creating links with up to three logical partition entries but these are not accepted.

TO DO

− Support big-endian architectures.
− Test on 64-bit architectures.
− Look for boot manager partitions (e.g. OS/2 BM).
− Think about reconstructing logical partition chains.

AUTHOR

Please send bug reports, suggestions, comments etc. to

Michail Brzitwa <michail@brzitwa.de>

SEE ALSO

fdisk(8).