NAME

gdisk – Interactive GUID partition table (GPT) manipulator

SYNOPSIS

gdisk [–l] *device*

DESCRIPTION

GPT fdisk (aka **gdisk**) is a text–mode menu–driven program for creation and manipulation of partition tables. It will automatically convert an old–style Master Boot Record (MBR) partition table or BSD disklabel stored without an MBR carrier partition to the newer Globally Unique Identifier (GUID) Partition Table (GPT) format, or will load a GUID partition table. When used with the *–l* command–line option, the program displays the current partition table and then exits.

GPT fdisk operates mainly on the GPT headers and partition tables; however, it can and will generate a fresh protective MBR, when required. (Any boot loader code in the protective MBR will not be disturbed.) If you've created an unusual protective MBR, such as a hybrid MBR created by **gptsync** or **gdisk**'s own hybrid MBR creation feature, this should not be disturbed by most ordinary actions. Some advanced data recovery options require you to understand the distinctions between the main and backup data, as well as between the GPT headers and the partition tables. For information on MBR vs. GPT, as well as GPT terminology and structure, see the extended **gdisk** documentation at *http://www.rodsbooks.com/gdisk/* or consult Wikipedia.

The **gdisk** program employs a user interface similar to that of Linux's **fdisk**, but **gdisk** modifies GPT partitions. It also has the capability of transforming MBR partitions or BSD disklabels into GPT partitions. Like the original **fdisk** program, **gdisk** does not modify disk structures until you explicitly write them to disk, so if you make a mistake, you can exit from the program with the 'q' option to leave your partitions unmodified.

Ordinarily, **gdisk** operates on disk device files, such as */dev/sda* or */dev/hda* under Linux, */dev/disk0* under Mac OS X, or */dev/ad0* or */dev/da0* under FreeBSD. The program can also operate on disk image files, which can be either copies of whole disks (made with **dd**, for instance) or raw disk images used by emulators such as QEMU or VMWare. Note that only *raw* disk images are supported; **gdisk** cannot work on compressed or other advanced disk image formats.

The MBR partitioning system uses a combination of cylinder/head/sector (CHS) addressing and logical block addressing (LBA). The former is klunky and limiting. GPT drops CHS addressing and uses 64–bit LBA mode exclusively. Thus, GPT data structures, and therefore **gdisk**, do not need to deal with CHS geometries and all the problems they create. Users of **fdisk** will note that **gdisk** lacks the options and limitations associated with CHS geometries.

For best results, you should use an OS–specific partition table program whenever possible. For example, you should make Mac OS X partitions with the Mac OS X Disk Utility program and Linux partitions with the Linux **gdisk** or GNU Parted program.

Upon start, **gdisk** attempts to identify the partition type in use on the disk. If it finds valid GPT data, **gdisk** will use it. If **gdisk** finds a valid MBR or BSD disklabel but no GPT data, it will attempt to convert the MBR or disklabel into GPT form. (BSD disklabels are likely to have unusable first and/or final partitions because they overlap with the GPT data structures, though.) GPT fdisk can identify, but not use data in, Apple Partition Map (APM) disks, which are used on 680x0– and PowerPC–based Macintoshes. Upon exiting with the 'w' option, **gdisk** replaces the MBR or disklabel with a GPT. *This action is potentially dangerous!* Your system may become unbootable, and partition type codes may become corrupted if the disk uses unrecognized type codes. Boot problems are particularly likely if you're multi–booting with any GPT–unaware OS. If you mistakenly launch **gdisk** on an MBR disk, you can safely exit the program without making any changes by using the 'q' option.

The MBR−to−GPT conversion will leave at least one gap in the partition numbering if the original MBR used logical partitions. These gaps are harmless, but you can eliminate them by using the 's' option, if you like. (Doing this may require you to update your */etc/fstab* file.)

When creating a fresh partition table, certain considerations may be in order:

* For data (non−boot) disks, and for boot disks used on BIOS−based computers with GRUB as the boot loader, partitions may be created in whatever order and in whatever sizes are desired.

* Boot disks for EFI−based systems require an *EFI System Partition* (**gdisk** internal code 0xEF00) formatted as FAT−32. I recommended making this partition 550 MiB. (Smaller ESPs are common, but some EFIs have flaky FAT drivers that necessitate a larger partition for reliable operation.) Boot−related files are stored here. (Note that GNU Parted identifies such partitions as having the "boot flag" set.)

* Some boot loaders for BIOS−based systems make use of a *BIOS Boot Partition* (**gdisk** internal code 0xEF02), in which the secondary boot loader is stored, possibly without the benefit of a filesystem. (GRUB2 may optionally use such a partition.) This partition can typically be quite small (roughly 32 to 200 KiB, although 1 MiB is more common in practice), but you should consult your boot loader documentation for details.

* If Windows is to boot from a GPT disk, a partition of type *Microsoft Reserved* (**gdisk** internal code 0x0C01) is recommended. This partition should be about 128 MiB in size. It ordinarily follows the EFI System Partition and immediately precedes the Windows data partitions. (Note that old versions of GNU Parted create all FAT partitions as this type, which actually makes the partition unusable for normal file storage in both Windows and Mac OS X.)

* Some OSes' GPT utilities create some blank space (typically 128 MiB) after each partition. The intent is to enable future disk utilities to use this space. Such free space is not required of GPT disks, but creating it may help in future disk maintenance. You can use GPT fdisk's relative partition positioning option (specifying the starting sector as '+128M', for instance) to simplify creating such gaps.

OPTIONS

−l List the partition table for the specified device and then exits.

Most interactions with **gdisk** occur with its interactive text−mode menus. Three menus exist: the main menu, the recovery & transformation menu, and the experts' menu. The main menu provides the functions that are most likely to be useful for typical partitioning tasks, such as creating and deleting partitions, changing partition type codes, and so on. Specific functions are:

b Save partition data to a backup file. You can back up your current in−memory partition table to a disk file using this option. The resulting file is a binary file consisting of the protective MBR, the main GPT header, the backup GPT header, and one copy of the partition table, in that order. Note that the backup is of the current in−memory data structures, so if you launch the program, make changes, and then use this option, the backup will reflect your changes. Note also that the restore option is on the recovery & transformation menu; the backup option is on the main menu to encourage its use.

c Change the GPT name of a partition. This name is encoded as a UTF−16 string, but proper entry and display of anything beyond basic ASCII values requires suitable locale and font support. For

the most part, Linux ignores the partition name, but it may be important in some OSes. GPT fdisk sets a default name based on the partition type code. Note that the GPT partition name is different from the filesystem name, which is encoded in the filesystem's data structures.

d Delete a partition. This action deletes the entry from the partition table but does not disturb the data within the sectors originally allocated to the partition on the disk. If a corresponding hybrid MBR partition exists, **gdisk** deletes it, as well, and expands any adjacent 0xEE (EFI GPT) MBR protective partition to fill the new free space.

i Show detailed partition information. The summary information produced by the 'p' command necessarily omits many details, such as the partition's unique GUID and the translation of **gdisk**'s internal partition type code to a plain type name. The 'i' option displays this information for a single partition.

l Display a summary of partition types. GPT uses a GUID to identify partition types for particular OSes and purposes. For ease of data entry, **gdisk** compresses these into two−byte (four−digit hexadecimal) values that are related to their equivalent MBR codes. Specifically, the MBR code is multiplied by hexadecimal 0x0100. For instance, the code for Linux swap space in MBR is 0x82, and it's 0x8200 in **gdisk**. A one−to−one correspondence is impossible, though. Most notably, the codes for all varieties of FAT and NTFS partition correspond to a single GPT code (entered as 0x0700 in **gdisk**). Some OSes use a single MBR code but employ many more codes in GPT. For these, **gdisk** adds code numbers sequentially, such as 0xa500 for a FreeBSD disklabel, 0xa501 for FreeBSD boot, 0xa502 for FreeBSD swap, and so on. Note that these two−byte codes are unique to **gdisk**. The type code list may optionally be filtered by a search string; for instance, entering **linux** shows only partition type codes with descriptions that include the string *Linux*. This search is performed case−insensitively.

n Create a new partition. This command is modeled after the equivalent **fdisk** option, although some differences exist. You enter a partition number, starting sector, and an ending sector. Both start and end sectors can be specified in absolute terms as sector numbers or as positions measured in kibibytes (K), mebibytes (M), gibibytes (G), tebibytes (T), or pebibytes (P); for instance, **40M** specifies a position 40MiB from the start of the disk. You can specify locations relative to the start or end of the specified default range by preceding the number by a '+' or '−' symbol, as in **+2G** to specify a point 2GiB after the default start sector, or **−200M** to specify a point 200MiB before the last available sector. Pressing the Enter key with no input specifies the default value, which is the start of the largest available block for the start sector and the end of the same block for the end sector.

o Clear out all partition data. This includes GPT header data, all partition definitions, and the protective MBR. The sector alignment is reset to the default (1 MiB, or 2048 sectors on a disk with 512-byte sectors).

p Display basic partition summary data. This includes partition numbers, starting and ending sector numbers, partition sizes, **gdisk**'s partition types codes, and partition names. For additional information, use the 'i' command.

q Quit from the program *without saving your changes*. Use this option if you just wanted to view information or if you make a mistake and want to back out of all your changes.

r Enter the recovery & transformation menu. This menu includes emergency recovery options (to fix damaged GPT data structures) and options to transform to or from other partitioning systems,

including creating hybrid MBRs.

s Sort partition entries. GPT partition numbers need not match the order of partitions on the disk. If you want them to match, you can use this option. Note that some partitioning utilities sort partitions whenever they make changes. Such changes will be reflected in your device filenames, so you may need to edit */etc/fstab* if you use this option.

t Change a single partition's type code. You enter the type code using a two−byte hexadecimal number, as described earlier. You may also enter a GUID directly, if you have one and **gdisk** doesn't know it.

v Verify disk. This option checks for a variety of problems, such as incorrect CRCs and mismatched main and backup data. This option does not automatically correct most problems, though; for that, you must use options on the recovery & transformation menu. If no problems are found, this command displays a summary of unallocated disk space.

w Write data. Use this command to save your changes.

x Enter the experts' menu. Using this option provides access to features you can use to get into even more trouble than the main menu allows.

? Print the menu. Type this command (or any other unrecognized command) to see a summary of available options.

The second **gdisk** menu is the recovery & transformation menu, which provides access to data recovery options and features related to the transformation of partitions between partitioning schemes (converting BSD disklabels into GPT partitions or creating hybrid MBRs, for instance). A few options on this menu duplicate functionality on the main menu, for the sake of convenience. The options on this menu are:

b Rebuild GPT header from backup. You can use the backup GPT header to rebuild the main GPT header with this option. It's likely to be useful if your main GPT header was damaged or destroyed (say, by sloppy use of **dd**).

c Load backup partition table. Ordinarily, **gdisk** uses only the main partition table (although the backup's integrity is checked when you launch the program). If the main partition table has been damaged, you can use this option to load the backup from disk and use it instead. Note that this will almost certainly produce no or strange partition entries if you've just converted an MBR disk to GPT format, since there will be no backup partition table on disk.

d Use main GPT header and rebuild the backup. This option is likely to be useful if the backup GPT header has been damaged or destroyed.

e Load main partition table. This option reloads the main partition table from disk. It's only likely to be useful if you've tried to use the backup partition table (via 'c') but it's in worse shape then the main partition table.

f Load MBR and build fresh GPT from it. Use this option if your GPT is corrupt or conflicts with the MBR and you want to use the MBR as the basis for a new set of GPT partitions.

g Convert GPT into MBR and exit. This option converts as many partitions as possible into MBR form, destroys the GPT data structures, saves the new MBR, and exits. Use this option if you've tried GPT and find that MBR works better for you. Note that this function generates up to four primary MBR partitions or three primary partitions and as many logical partitions as can be generated. Each logical partition requires at least one unallocated block immediately before its first block. Therefore, it may be possible to convert a maximum of four partitions on disks with tightly−packed partitions; however, if free space was inserted between partitions when they were created, and if the disk is under 2 TiB in size, it should be possible to convert all the partitions to MBR form. See also the 'h' option.

h Create a hybrid MBR. This is an ugly workaround that enables GPT−unaware OSes, or those that can't boot from a GPT disk, to access up to three of the partitions on the disk by creating MBR entries for them. Note that these hybrid MBR entries can easily go out of sync with the GPT entries, particularly when hybrid−unaware GPT utilities are used to edit the disk. Thus, you may need to re−create the hybrid MBR if you use such tools. Unlike the 'g' option, this option does not support converting any partitions into MBR logical partitions.

i Show detailed partition information. This option is identical to the 'i' option on the main menu.

l Load partition data from a backup file. This option is the reverse of the 'b' option on the main menu. Note that restoring partition data from anything but the original disk is not recommended.

m Return to the main menu. This option enables you to enter main−menu commands.

o Print protective MBR data. You can see a summary of the protective MBR's partitions with this option. This may enable you to spot glaring problems or help identify the partitions in a hybrid MBR.

p Print the partition table. This option is identical to the 'p' option in the main menu.

q Quit without saving changes. This option is identical to the 'q' option in the main menu.

t Transform BSD partitions into GPT partitions. This option works on BSD disklabels held within GPT (or converted MBR) partitions. Converted partitions' type codes are likely to need manual adjustment. **gdisk** will attempt to convert BSD disklabels stored on the main disk when launched, but this conversion is likely to produce first and/or last partitions that are unusable. The many BSD variants means that the probability of **gdisk** being unable to convert a BSD disklabel is high compared to the likelihood of problems with an MBR conversion.

v Verify disk. This option is identical to the 'v' option in the main menu.

w Write table to disk and exit. This option is identical to the 'w' option in the main menu.

x Enter the experts' menu. This option is identical to the 'x' option in the main menu.

? Print the menu. This option (or any unrecognized entry) displays a summary of the menu options.

The third **gdisk** menu is the experts' menu. This menu provides advanced options that aren't closely related to recovery or transformation between partitioning systems. Its options are:

a Set attributes. GPT provides a 64−bit attributes field that can be used to set features for each partition. **gdisk** supports four attributes: *system partition*, *read−only*, *hidden*, and *do not automount*. You can set other attributes, but their numbers aren't translated into anything useful. In practice, most OSes seem to ignore these attributes.

c Change partition GUID. You can enter a custom unique GUID for a partition using this option. (Note this refers to the GUID that uniquely identifies a partition, not to its type code, which you can change with the 't' main−menu option.) Ordinarily, **gdisk** assigns this number randomly; however, you might want to adjust the number manually if you've wound up with the same GUID on two partitions because of buggy GUID assignments (hopefully not in **gdisk**) or sheer incredible coincidence.

d Display the sector alignment value. See the description of the 'l' option for more details.

e Move backup GPT data structures to the end of the disk. Use this command if you've added disks to a RAID array, thus creating a virtual disk with space that follows the backup GPT data structures. This command moves the backup GPT data structures to the end of the disk, where they belong.

f Randomize the disk's GUID and all partitions' unique GUIDs (but not their partition type code GUIDs). This function may be used after cloning a disk with another utility in order to render all GUIDs once again unique.

g Change disk GUID. Each disk has a unique GUID code, which **gdisk** assigns randomly upon creation of the GPT data structures. You can generate a fresh random GUID or enter one manually with this option.

h Recompute CHS values in protective or hybrid MBR. This option can sometimes help if a disk utility, OS, or BIOS doesn't like the CHS values used by the partitions in the protective or hybrid MBR. In particular, the GPT specification requires a CHS value of 0xFFFFFF for over-8GiB partitions, but this value is technically illegal by the usual standards. Some BIOSes hang if they encounter this value. This option will recompute a more normal CHS value -- 0xFEFFFF for over-8GiB partitions, enabling these BIOSes to boot.

i Show detailed partition information. This option is identical to the 'i' option on the main menu.

j Adjust the location of the main partition table. This value is normally 2, but it may need to be increased in some cases, such as when a system−on−chip (SoC) is hard−coded to read boot code from sector 2. I recommend against adjusting this value unless doing so is absolutely necessary.

l Change the sector alignment value. Disks with more logical sectors per physical sectors (such as modern Advanced Format drives), some RAID configurations, and many SSD devices, can suffer performance problems if partitions are not aligned properly for their internal data structures. On new disks, GPT fdisk attempts to align partitions on 1 MiB boundaries (2048−sectors on disks with 512-byte sectors) by default, which optimizes performance for all of these disk types. On pre−partitioned disks, GPT fdisk attempts to identify the alignment value used on that disk, but will set 8-sector alignment on disks larger than 300 GB even if lesser alignment values are detected. In either case, it can be changed by using this option.

m Return to the main menu. This option enables you to enter main−menu commands.

n Create a new protective MBR. Use this option if the current protective MBR is damaged in a way that **gdisk** doesn't automatically detect and correct, or if you want to convert a hybrid MBR into a "pure" GPT with a conventional protective MBR.

o Print protective MBR data. You can see a summary of the protective MBR's partitions with this option. This may enable you to spot glaring problems or help identify the partitions in a hybrid MBR.

p Print the partition table. This option is identical to the 'p' option in the main menu.

q Quit without saving changes. This option is identical to the 'q' option in the main menu.

r Enter the recovery & transformations menu. This option is identical to the 'r' option on the main menu.

s Resize partition table. The default partition table size is 128 entries. Officially, sizes of less than 16KB (128 entries, given the normal entry size) are unsupported by the GPT specification; however, in practice they seem to work, and can sometimes be useful in converting MBR disks. Larger sizes also work fine. OSes may impose their own limits on the number of partitions, though.

t Swap two partitions' entries in the partition table. One partition may be empty. For instance, if partitions 1−4 are defined, transposing 1 and 5 results in a table with partitions numbered from 2−5. Transposing partitions in this way has no effect on their disk space allocation; it only alters their order in the partition table.

u Replicate the current device's partition table on another device. You will be prompted to type the new device's filename. After the write operation completes, you can continue editing the original device's partition table. Note that the replicated partition table is an exact copy, including all GUIDs; if the device should have its own unique GUIDs, you should use the **f** option on the new disk.

v Verify disk. This option is identical to the 'v' option in the main menu.

z Zap (destroy) the GPT data structures and exit. Use this option if you want to repartition a GPT disk using **fdisk** or some other GPT−unaware program. You'll be given the choice of preserving the existing MBR, in case it's a hybrid MBR with salvageable partitions or if you've already created new MBR partitions and want to erase the remnants of your GPT partitions. *If you've already created new MBR partitions, it's conceivable that this option will damage the first and/or last MBR partitions!* Such an event is unlikely, but could occur if your new MBR partitions overlap the old GPT data structures.

? Print the menu. This option (or any unrecognized entry) displays a summary of the menu options.

In many cases, you can press the Enter key to select a default option when entering data. When only one option is possible, **gdisk** usually bypasses the prompt entirely.

BUGS

Known bugs and limitations include:

* The program compiles correctly only on Linux, FreeBSD, Mac OS X, and Windows. Linux versions for x86−64 (64−bit), x86 (32−bit), and PowerPC (32−bit) have been tested, with the x86−64 version having seen the most testing. Under FreeBSD, 32−bit (x86) and 64−bit (x86−64) versions have been tested. Only 32−bit versions for Mac OS X and Windows have been tested by the author, although I've heard of 64-bit versions being successfully compiled.

* The FreeBSD version of the program can't write changes to the partition table to a disk when existing partitions on that disk are mounted. (The same problem exists with many other FreeBSD utilities, such as **gpt**, **fdisk**, and **dd**.) This limitation can be overcome by typing **sysctl kern.geom.debugflags=16** at a shell prompt.

* The fields used to display the start and end sector numbers for partitions in the 'p' command are 14 characters wide. This translates to a limitation of about 45 PiB. On larger disks, the displayed columns will go out of alignment.

* In the Windows version, only ASCII characters are supported in the partition name field. If an existing partition uses non−ASCII UTF−16 characters, they're likely to be corrupted in the 'i' and 'p' menu options' displays; however, they should be preserved when loading and saving partitions. Binaries for Linux, FreeBSD, and OS X support full UTF-16 partition names.

* The program can load only up to 128 partitions (4 primary partitions and 124 logical partitions) when converting from MBR format. This limit can be raised by changing the *#define MAX_MBR_PARTS* line in the *basicmbr.h* source code file and recompiling; however, such a change will require using a larger−than−normal partition table. (The limit of 128 partitions was chosen because that number equals the 128 partitions supported by the most common partition table size.)

* Converting from MBR format sometimes fails because of insufficient space at the start or (more commonly) the end of the disk. Resizing the partition table (using the 's' option in the experts' menu) can sometimes overcome this problem; however, in extreme cases it may be necessary to resize a partition using GNU Parted or a similar tool prior to conversion with **gdisk**.

* MBR conversions work only if the disk has correct LBA partition descriptors. These descriptors should be present on any disk over 8 GiB in size or on smaller disks partitioned with any but very ancient software.

* BSD disklabel support can create first and/or last partitions that overlap with the GPT data structures. This can sometimes be compensated by adjusting the partition table size, but in extreme cases the affected partition(s) may need to be deleted.

* Because of the highly variable nature of BSD disklabel structures, conversions from this form may be unreliable −− partitions may be dropped, converted in a way that creates overlaps with other partitions, or converted with incorrect start or end values. Use this feature with caution!

* Booting after converting an MBR or BSD disklabel disk is likely to be disrupted. Sometimes re−installing a boot loader will fix the problem, but other times you may need to switch boot loaders. Except on EFI−based platforms, Windows through at least Windows 7 doesn't support booting

from GPT disks. Creating a hybrid MBR (using the 'h' option on the recovery & transformation menu) or abandoning GPT in favor of MBR may be your only options in this case.

AUTHORS

Primary author: Roderick W. Smith (rodsmith@rodsbooks.com)

Contributors:

* Yves Blusseau (1otnwmz02@sneakemail.com)

* David Hubbard (david.c.hubbard@gmail.com)

* Justin Maggard (justin.maggard@netgear.com)

* Dwight Schauer (dschauer@gmail.com)

* Florian Zumbiehl (florz@florz.de)

SEE ALSO

cfdisk(8), **cgdisk**(8), **fdisk**(8), **mkfs**(8), **parted**(8), **sfdisk**(8), **sgdisk**(8), **fixparts**(8).

http://en.wikipedia.org/wiki/GUID_Partition_Table

http://developer.apple.com/technotes/tn2006/tn2166.html

http://www.rodsbooks.com/gdisk/

AVAILABILITY

The **gdisk** command is part of the *GPT fdisk* package and is available from Rod Smith.