Partitioning a disk prior to Linux Installation

by Andy Pepperdine

This paper will describe how to partition a disk how you want it before you install Linux. The partitioning process may be initiated either directly from a live CD and before the installation is started. Or it may be done as part of the installation process when you select “manual” (or some similar option) when the time to comes to select how and where the system should be placed on a hard disk.

I will be using screenshots from the application GParted, which is often the one used in live CD's for investigation of existing disk drive configurations, and during the installation process itself to manipulate the partitions. Note that if you are working in an existing Linux installation, it may be necessary to install GParted.

If you do not have access to this, there is also a “live CD” version of GParted alone. See the references for where to get it from. By booting your PC from this CD you will get just GParted and can define your partitions appropriately before doing anything else.

Setting up

When you first start GParted, it will search for all the hard disks it can find. **Make sure that it is displaying the contents of the correct disk drive.** You can change the drive it is looking at from the drop-down menu at the top right of the window.

Also, **make sure you have taken a copy of all essential data from the drive**. Changing partitions is never to be done lightly, and you do not want to lose any of the data you have accumulated.

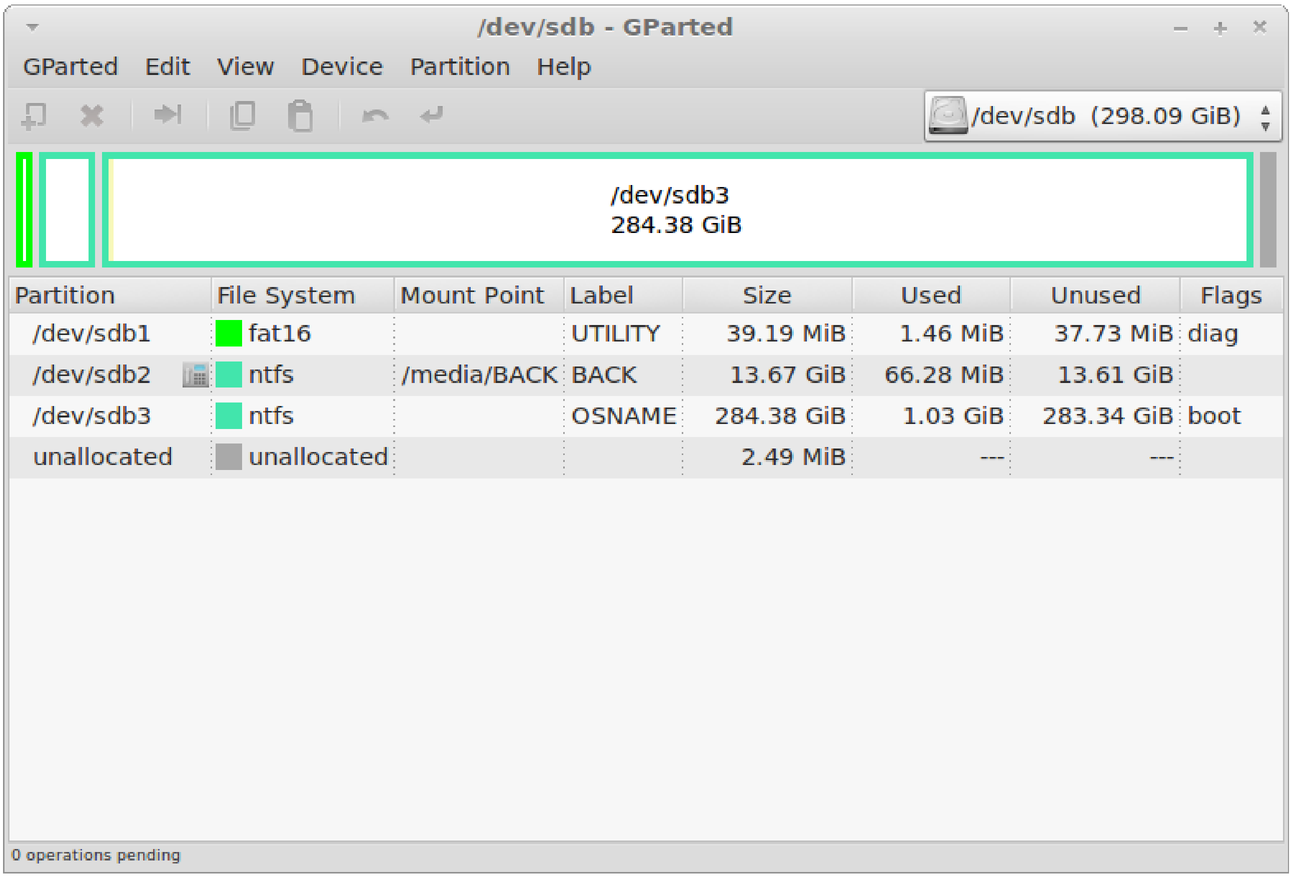
It is strongly advised that you clean up your file system and do a “defrag” before you start as well, as that will make recovering space in the main partition more effective.

The starting position may look something like Figure 1 below, but different machines vary. You will notice a very small amount of unallocated space at the end. This is usually due to the preceding partition being aligned on cylinder boundaries, often to make copying and recovery easier, or to fit with other special requirements. It has no significant effect.

A note about the numbers for the sizes

You will see the numbers quoted as MiB and GiB, and not MB and GB. This is because the standard these days is to distinguish between megabytes (MB) calculated in powers of ten, and mebibytes (MiB) in powers of 2. Since $2^{10}=1024$, is a little more than $10^{3}=1000$, the difference can cause confusion. When different numbers are used from what was expected, the sums will not add up right.

MiB is pronounced mebibyte, where the “bi” implies binary. GiB is gibibyte, TiB is tebibyte, etc. going up in multiples of 1024.



Partition	File System	Mount Point	Label	Size	Used	Unused	Flags
/dev/sdb1	🟩 fat16		UTILITY	39.19 MiB	1.46 MiB	37.73 MiB	diag
/dev/sdb2	🟩 ntfs	/media/BACK	BACK	13.67 GiB	66.28 MiB	13.61 GiB	
/dev/sdb3	🟩 ntfs		OSNAME	284.38 GiB	1.03 GiB	283.34 GiB	boot
unallocated	▨ unallocated			2.49 MiB	---	---	

0 operations pending

Figure 1: Initial state

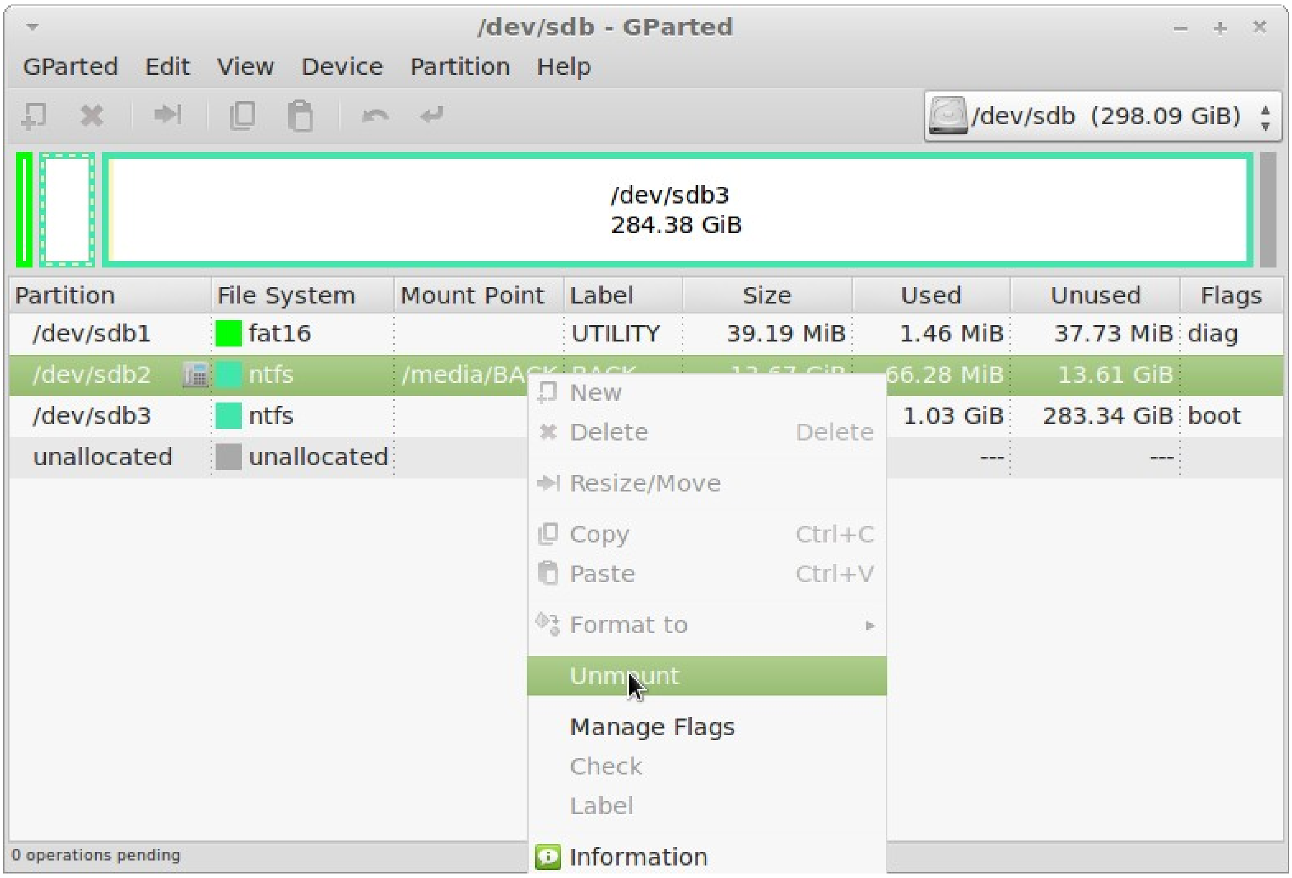
GParted will show each partition on the disk and give some basic information about it. In this particular example, the disk has a small partition at the beginning to hold a utility from the manufacturer of the PC. **Do not change this**, since it may affect any diagnostic ability that they have incorporated.

In the case above, there is also a recovery partition. **Do not change this one either**, as it may be included by the system maker as a backup for re-installing a system that has been corrupted.

The largest partition is almost certainly the one containing the working system, and all the data, both for the system, and your data. This will be the one that will be changed to create space for a Linux system to be included on the disk in parallel with its current contents. Make sure that there is enough free space (Unused column) before continuing.

Clearing the way to prepare the disk

One or more of the partitions may be “mounted”. This is shown by an entry in the column labeled Mount Point, and also by an icon (either the block here, or a key symbol) against the name of the partition in the left hand column. Before you can modify any partition, it must be unmounted.



Partition	File System	Mount Point	Label	Size	Used	Unused	Flags
/dev/sdb1	🟩 fat16		UTILITY	39.19 MiB	1.46 MiB	37.73 MiB	diag
/dev/sdb2	🟩 ntfs	/media/BACK	BACK		66.28 MiB	13.61 GiB	
/dev/sdb3	🟩 ntfs				1.03 GiB	283.34 GiB	boot
unallocated	▨ unallocated				---	---	

Figure 2: Unmounting

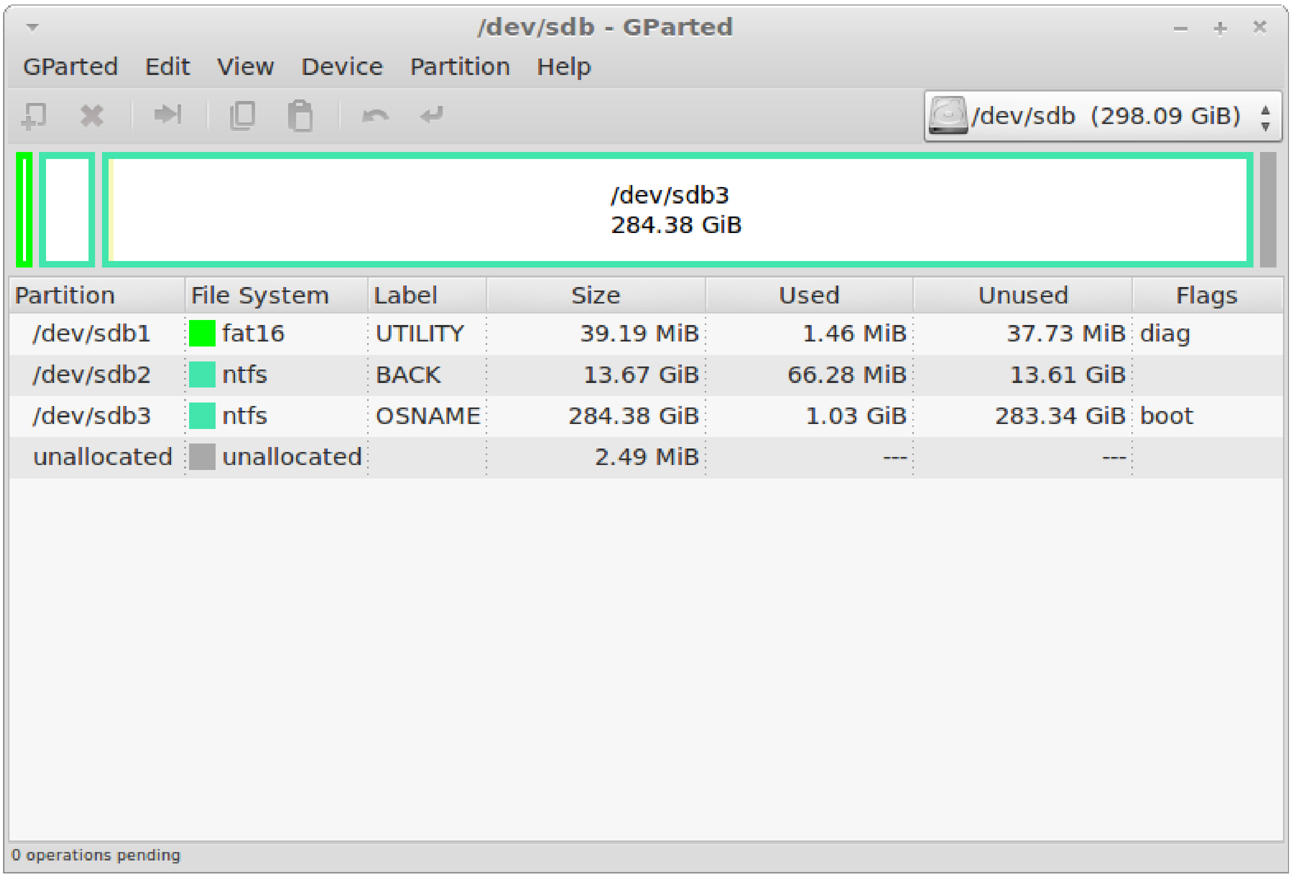
To unmount a partition, select the partition by clicking on the line. The right click on it and select the Unmount command. Wait for it to take effect. If there are no mounted partitions, the column Mount will not be shown.

Start position and appearance

The state you should now see is the actual allocation of space among the various partitions. Some of them may be marked with flags as seen in the right hand column. The only flags you should need to change will be the boot flag, but that will occur naturally later in the installation process. Otherwise, it is recommend you leave these alone.

Note also that there is a column called File System. A computer program cannot understand the contents of a partition on a disk unless it knows how it is laid out – what its format is. The structures of each may differ as there are several different ways in which they can be structured. These different structures are known as file systems. Different file systems use different structures. Windows systems these days will be using a file system called NTFS. USB sticks for transferring data between systems will probably be using a FAT16, or possibly FAT32 (also called VFAT).

Linux can use many different types, and some comments will appear later when we get to defining them in the demonstration.



Partition	File System	Label	Size	Used	Unused	Flags
/dev/sdb1	🟩 fat16	UTILITY	39.19 MiB	1.46 MiB	37.73 MiB	diag
/dev/sdb2	🟩 ntfs	BACK	13.67 GiB	66.28 MiB	13.61 GiB	
/dev/sdb3	🟩 ntfs	OSNAME	284.38 GiB	1.03 GiB	283.34 GiB	boot
unallocated	▨ unallocated		2.49 MiB	---	---	

0 operations pending

Figure 3: Ready for work

Before going any further, you should prepare the amounts you want for each partition that you will create and the one you will take the space from. It will be important that you reduce the main partition just once if at all possible as this is the most difficult part of the operation to do and the most likely to go wrong.

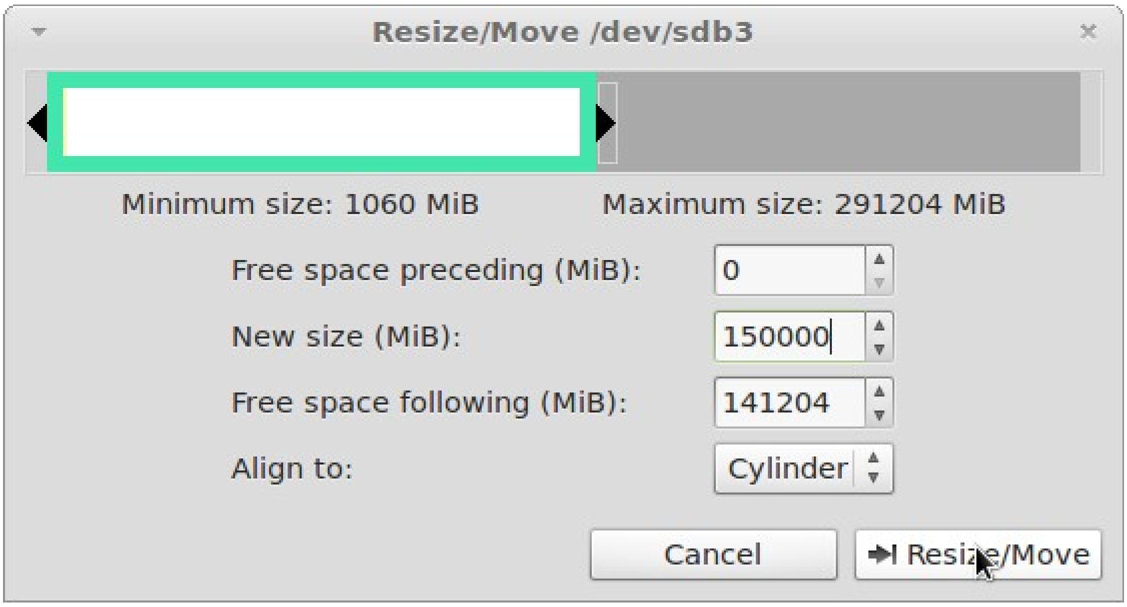
Once you have this list ready, you will first shrink the main partition, then create the others one by one. GParted does not perform the commands immediately, but places them in a “to do” list. You then ask it to apply all pending requests. It will also ask whether you wish to go ahead to make sure you really do want to go ahead. Nothing will change until you reply that you do.

Resize the existing system partition and file system

This is where we make space for the new system(s). The partition to be reduced will almost certainly contain an NTFS file system. GParted themselves recommend that this action should be done as a single operation. Do not combine with with other actions. They also suggest after resizing this partition, you should reboot into Windows twice immediately to allow Windows to do its file system checking and ensure that all is on order.

Do not move the partition, only resize it, unless you know how to fix the problem with the booting process because it will be looking in the wrong place afterwards, and it may decide that the drive is no longer C:, which will throw the registry off.

Using the menu item Partition → Resize/Move, you will get the dialog window in Figure 4.



Resize/Move /dev/sdb3

Minimum size: 1060 MiB

Maximum size: 291204 MiB

Free space preceding (MiB): 0

New size (MiB): 150000

Free space following (MiB): 141204

Align to: Cylinder

Figure 4: Resizing an ntfs partition

When changing the numbers in this dialog, leave the Free space preceding at zero MiB. Change the New size to the size you want it to be, and leave everything else alone. The Align to: in the above case has been set to Cylinder, but that is not necessary. Leave it as MiB so that there is little danger of it being moved surreptitiously.

You will also see the diagram above the figures adjust to show how much space is being freed relative to the actual current size.

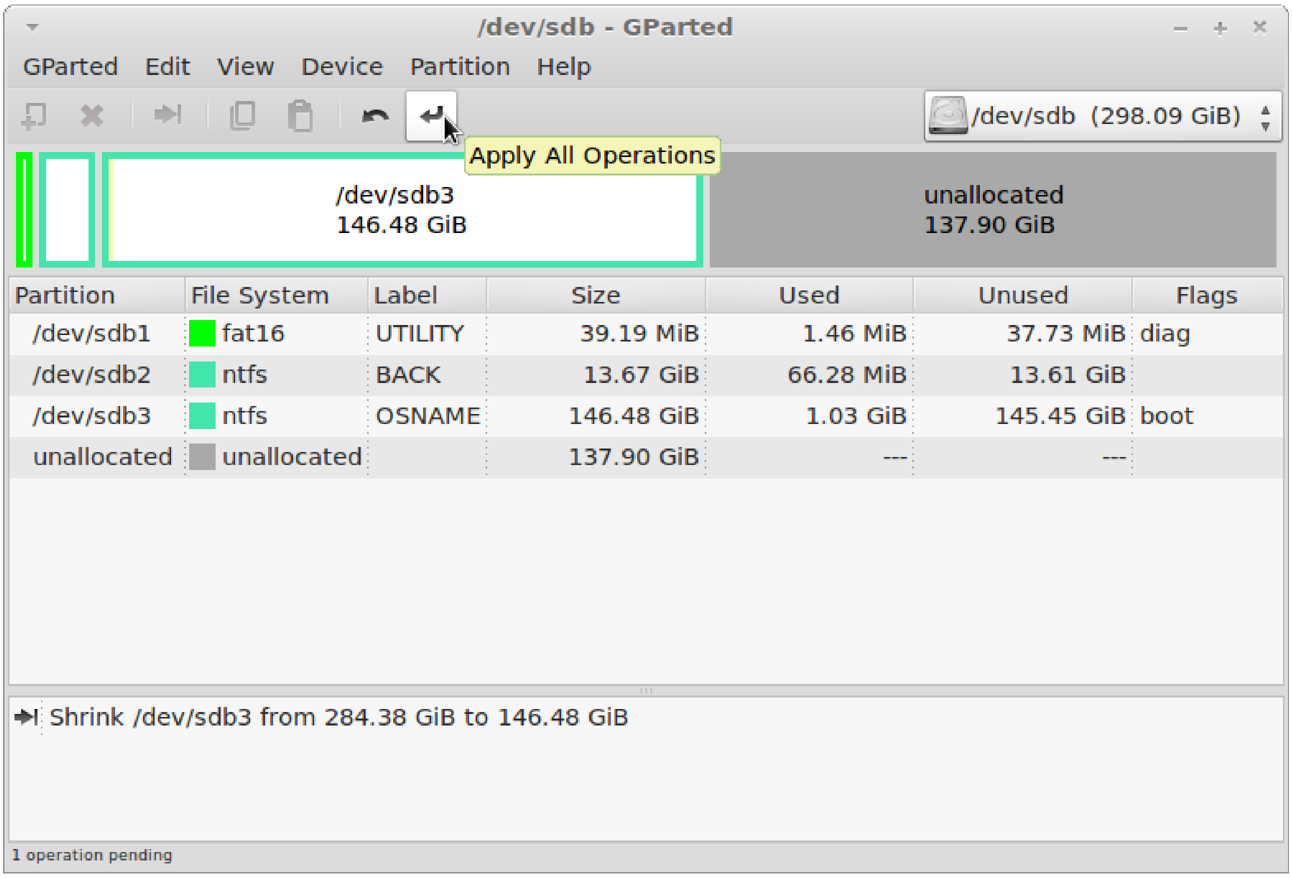
Clicking on Resize/Move will place the task in a new pane that will appear in the main window.

About to shrink

The new pane will contain all the commands that have not yet been done, as in Figure 5. It is at this point you can check it says it will shrink the partition, and not move it. remember that the job has not been carried out yet, you still have time to review the state by looking at the spaces presented in the table, and also the picture above the table, all of which will have been adjusted to reflect what will be the case after the work is done.

If you hover the mouse over the icon that looks like the inscription on a Return key (as shown in the figure), you will see it says “Apply All Operations”. Hitting that icon will give you one last chance to cancel before it starts the work of shrinking.

It appears from searching documentation that GParted uses a program called ntsfclone to do the resizing job. So far as I can see, this program does no recovery of space, nor does it “defrag” the file system. You can only make space free that is currently not used. This means that it is strongly advisable to do any moving around of files, and “defragging”, **before** starting any of this work.



Partition	File System	Label	Size	Used	Unused	Flags
/dev/sdb1	🟩 fat16	UTILITY	39.19 MiB	1.46 MiB	37.73 MiB	diag
/dev/sdb2	🟩 ntfs	BACK	13.67 GiB	66.28 MiB	13.61 GiB	
/dev/sdb3	🟩 ntfs	OSNAME	146.48 GiB	1.03 GiB	145.45 GiB	boot
unallocated	▨ unallocated		137.90 GiB	---	---	

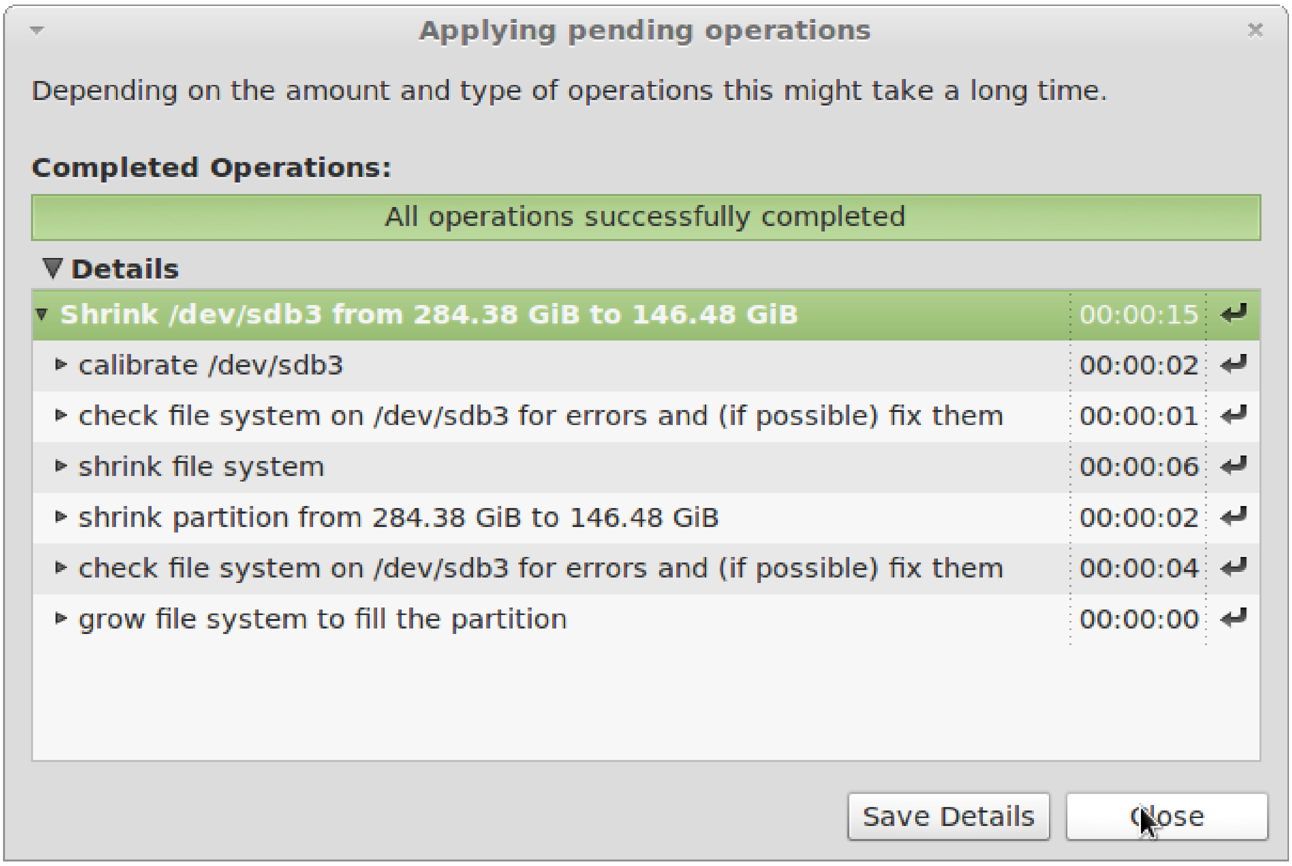
Shrink /dev/sdb3 from 284.38 GiB to 146.48 GiB

1 operation pending

Figure 5: About to shrink a partition

Result of shrinking

After the command has been performed, the result is as shown in Figure 6, where the details box has been expanded by clicking on the little triangle to the left of the word Details. This is useful if there have been any untoward events and messages generated, when they will appear in the detail box. As you can see, it lists all the activities that had to be done to complete the single request.



Depending on the amount and type of operations this might take a long time.

Completed Operations:

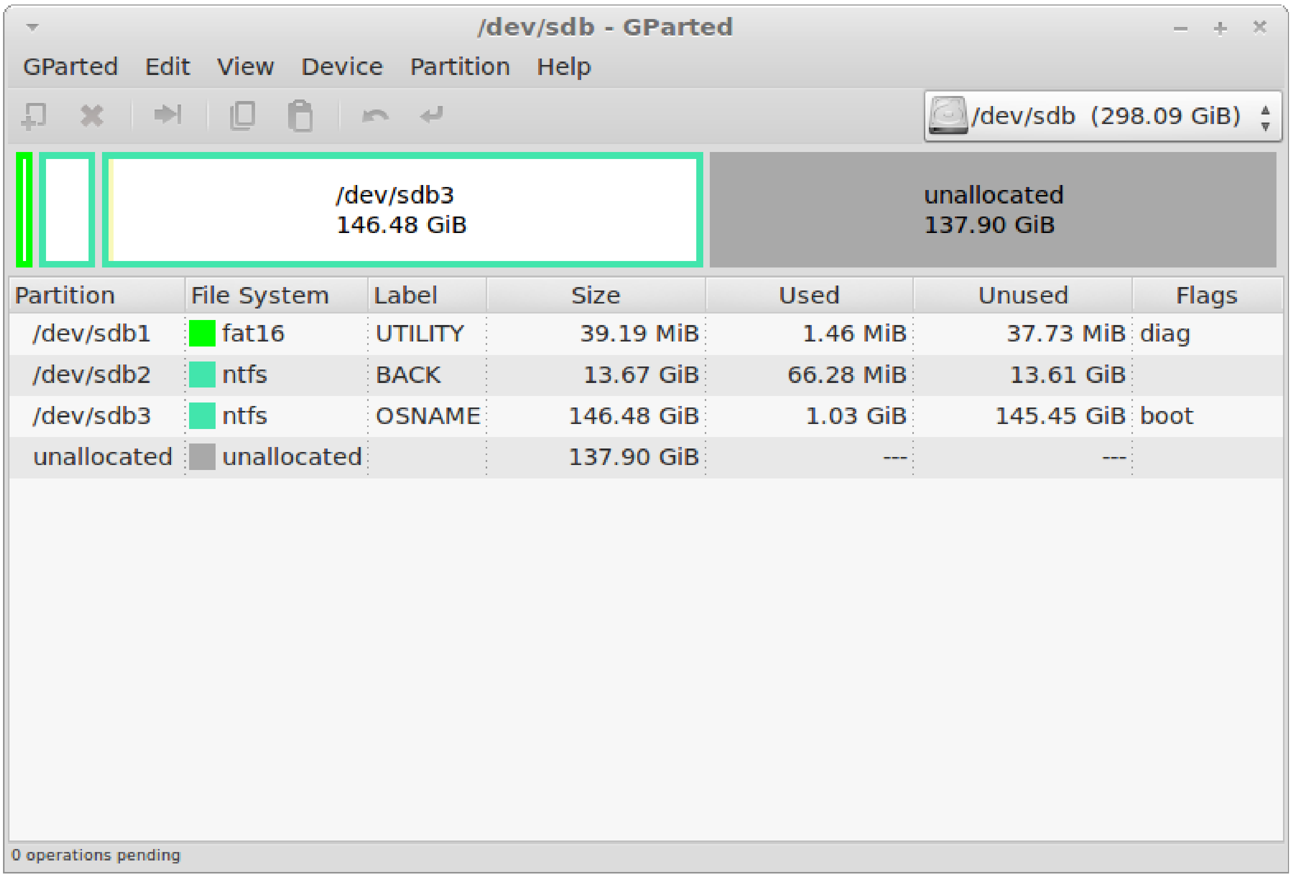
All operations successfully completed

▼ Shrink /dev/sdb3 from 284.38 GiB to 146.48 GiB	00:00:15
▸ calibrate /dev/sdb3	00:00:02
▸ check file system on /dev/sdb3 for errors and (if possible) fix them	00:00:01
▸ shrink file system	00:00:06
▸ shrink partition from 284.38 GiB to 146.48 GiB	00:00:02
▸ check file system on /dev/sdb3 for errors and (if possible) fix them	00:00:04
▸ grow file system to fill the partition	00:00:00

Figure 6: After a partition has been shrunk, with details

Clicking the Close button will return you to the main window after the partition table has been re-examined. It takes a little time to re-evaluate the situation, so be patient.

The new partition table will look something like Figure 7, when you can plan the next steps.



Partition	File System	Label	Size	Used	Unused	Flags
/dev/sdb1	🟩 fat16	UTILITY	39.19 MiB	1.46 MiB	37.73 MiB	diag
/dev/sdb2	🟩 ntfs	BACK	13.67 GiB	66.28 MiB	13.61 GiB	
/dev/sdb3	🟩 ntfs	OSNAME	146.48 GiB	1.03 GiB	145.45 GiB	boot
unallocated	▨ unallocated		137.90 GiB	---	---	

Figure 7: Ready to create new partitions

A word about labels

You will probably be looking at this partition table later on and will need to remember what each is for. Labels are very useful to remind you of their purpose. However, they can only be applied to partitions that have some form of file system (or swap) in them. Labels are an attribute of the file system, and consequently the number of characters available for a label varies between files systems, as does the type of characters. To be sure, it is best to stick always to upper case letters.

For example, FAT file systems have a limit of 11 characters, XFS a limit of 12, Linux systems ext2, 3, and 4, have a limit of 16, and ntfs has a limit of 128 Unicode characters.

Some systems (e.g. Redhat Linux) want all labels to be unique system-wide, so choose carefully when deciding how to name your file systems.

A word about partitions, primary, extended and logical

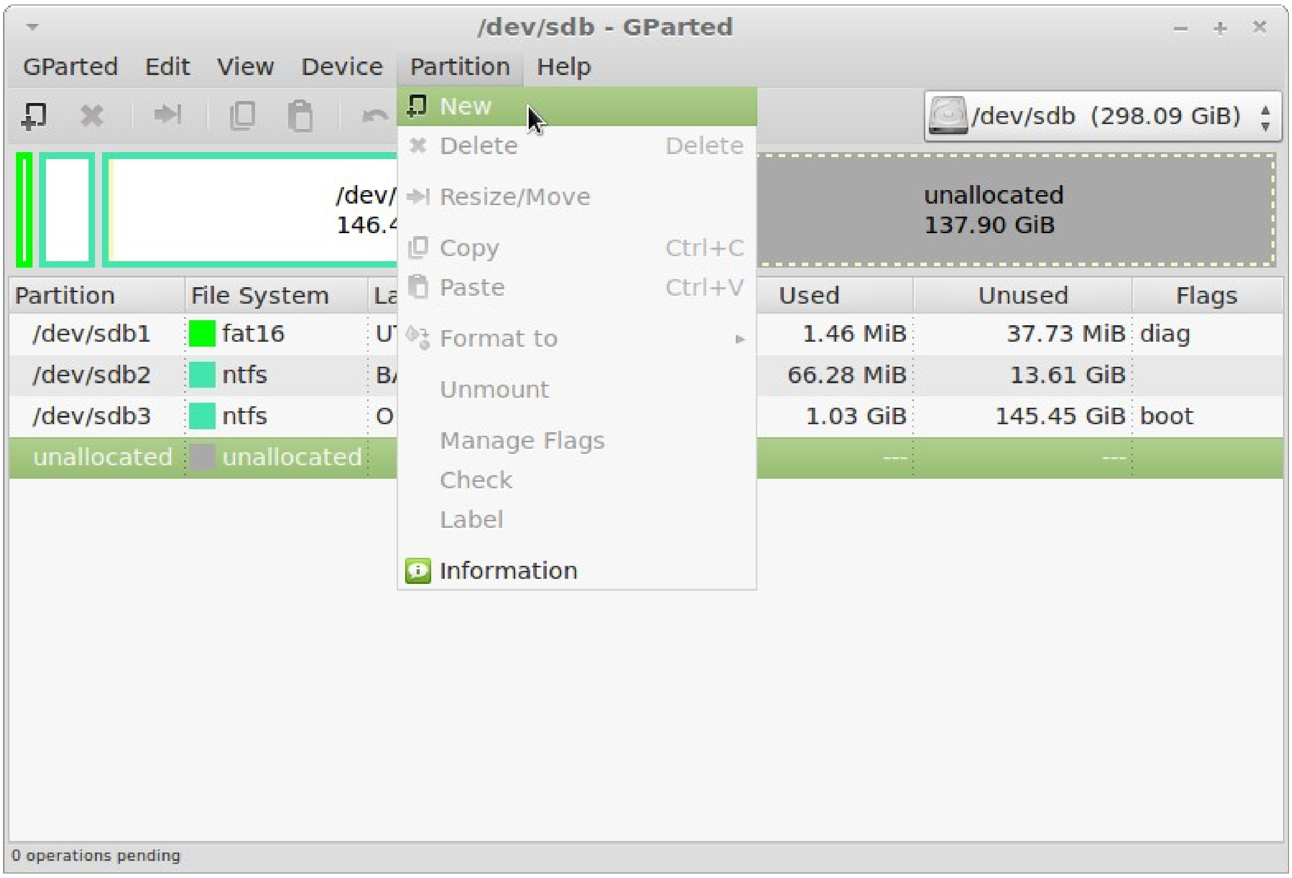
The standard DOS partition table on the vast majority of devices has space for only 4 (yes, four) **primary** partitions. As you can see in the above case, three are already in use. We are going to want more than four.

The way we do that is to define an **extended** partition, and then create **logical** partitions within the extended partition. In theory you can have as many logical partitions as you like, but in practice it is probably unwise to define more than is strictly necessary. Extended partitions contain only logical partitions, not file systems in themselves.

Note that Windows generally can be booted only from a primary partition, but Linux does not mind whether it is primary or logical, assuming the initial boot loader is sensible. Technically, Windows can run from a logical partition, but still needs a primary boot partition to get started. Best to avoid the problem.

Prepare a new extended partition

To create a new partition, first select the unallocated space where it is to be put, and go to Partition → New, as in Figure 8.



Partition	File System	La	Used	Unused	Flags
/dev/sdb1	🟩 fat16	U	1.46 MiB	37.73 MiB	diag
/dev/sdb2	🟩 ntfs	B	66.28 MiB	13.61 GiB	
/dev/sdb3	🟩 ntfs	O	1.03 GiB	145.45 GiB	boot
unallocated	▨ unallocated		---	---	

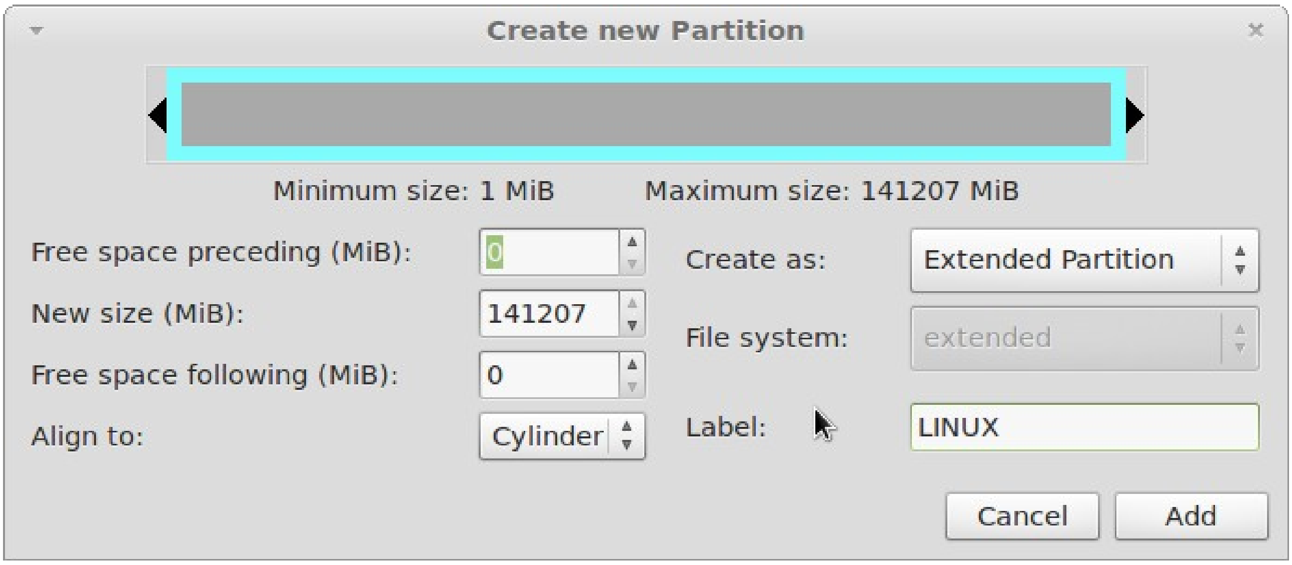
0 operations pending

Figure 8: To create a new partition

This will put you into the window in Figure 9, where you define the partition size etc. In our case, here, we want the type to be Extended, and to fill the whole of the unallocated space. Just as an example, I've changed the alignment to Cylinder, which is not strictly necessary – it can have minor effects when speed is important, but for normal cases it is not important.

In this case, although I set the Label field, an extended partition cannot contain a label, and it was ignored as will be seen from the result.

Closing the window, and then applying the command as before, will then show an extended partition whose content is all unallocated space.



Minimum size: 1 MiB Maximum size: 141207 MiB

Free space preceding (MiB): 0

New size (MiB): 141207

Free space following (MiB): 0

Align to: Cylinder

Create as: Extended Partition

File system: extended

Label: LINUX

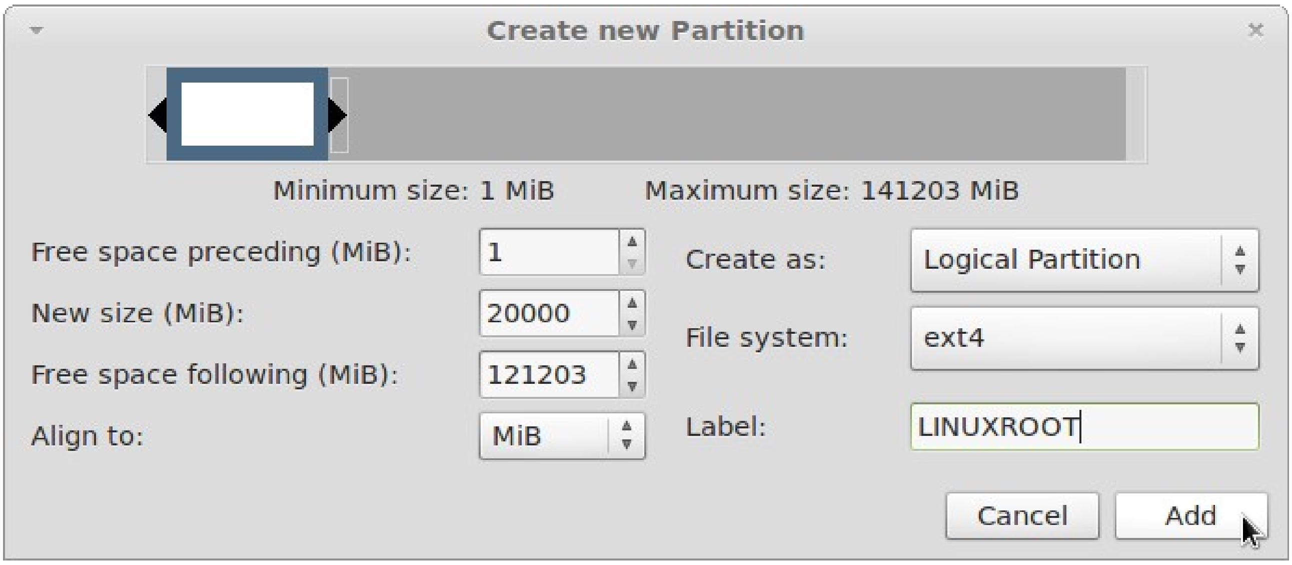
Figure 9: Creating extended partition

Make a root partition

The next task is to make a logical partition for the Linux system. So we need to create a logical partition, inside the extended one, which will be used to contain the root directory when Linux is run. In this case, I've set the details as in Figure 10, where the size is set to 20GiB – easily enough for all applications that you would expect to install. If you are short of space, 7 GiB should be adequate except in very unusual circumstances.

In this case, I set the New size, and let it compute the free space left.

This is not the right place to discuss which file system to use, but ext4 is a well established journaling file system, which means it is safe against power failures at unexpected times.



Minimum size: 1 MiB Maximum size: 141203 MiB

Free space preceding (MiB): 1

New size (MiB): 20000

Free space following (MiB): 121203

Align to: MiB

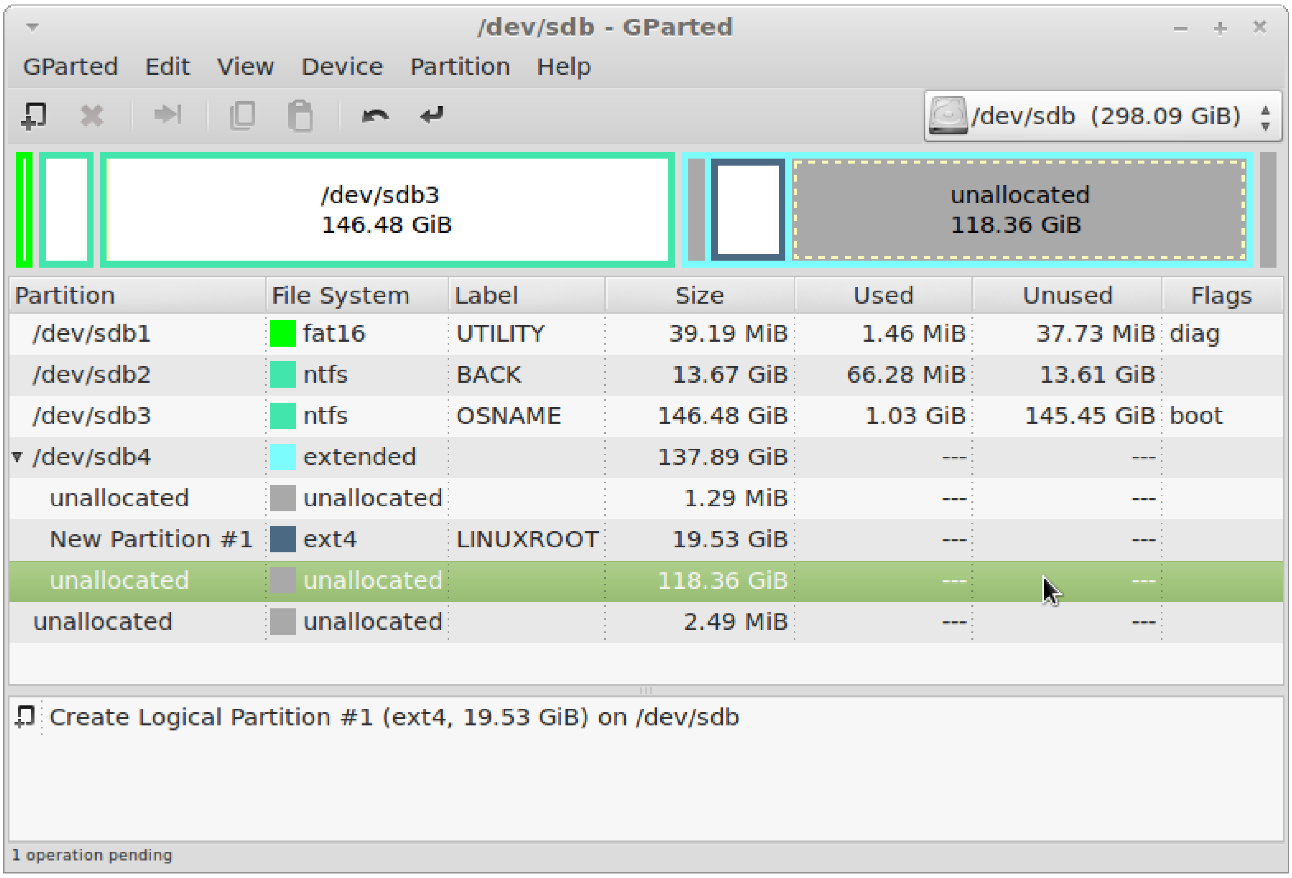
Create as: Logical Partition

File system: ext4

Label: LINUXROOT

Figure 10: Defining a Linux root partition

Before the partition is actually created, the window looks like Figure 11.



Partition	File System	Label	Size	Used	Unused	Flags
/dev/sdb1	🟩 fat16	UTILITY	39.19 MiB	1.46 MiB	37.73 MiB	diag
/dev/sdb2	🟩 ntfs	BACK	13.67 GiB	66.28 MiB	13.61 GiB	
/dev/sdb3	🟩 ntfs	OSNAME	146.48 GiB	1.03 GiB	145.45 GiB	boot
▾ /dev/sdb4	🟦 extended		137.89 GiB	---	---	
unallocated	⬜ unallocated		1.29 MiB	---	---	
New Partition #1	🟦 ext4	LINUXROOT	19.53 GiB	---	---	
unallocated	⬜ unallocated		118.36 GiB	---	---	
unallocated	⬜ unallocated		2.49 MiB	---	---	

Create Logical Partition #1 (ext4, 19.53 GiB) on /dev/sdb

1 operation pending

Figure 11: Ready to create root logical partition

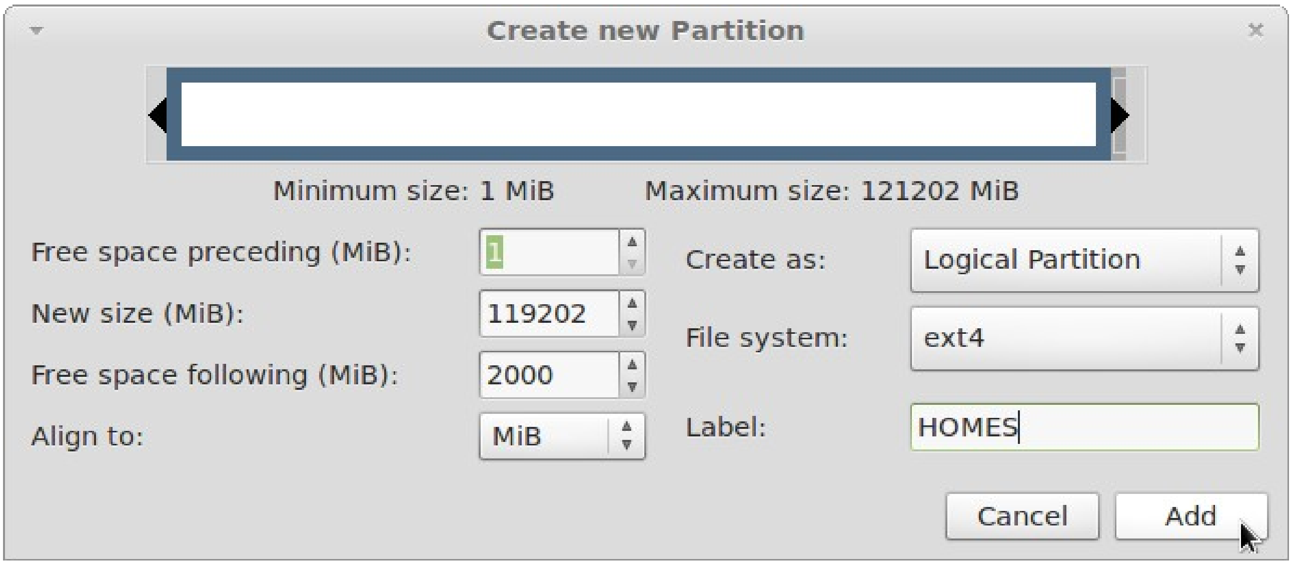
Applying the operations will complete the job when the logical partition will turn up as /dev/sdb5.

Adding home partition

It is not essential to create another partition for all the user's data, but is recommended. It is easier to format the system partition and install a new system, while leaving all the user's settings and data alone in another. You may not think that is important now, but later you may well be glad of it.

The other partition required will be the swap space. So to define the home partition as large as possible, I set the size of the unallocated space I was to leave for the swap area (2 GiB), and let it compute the actual size of the partition.

The sizes and other characteristics are shown in 12. Again, I used an ext4 file system.



Create new Partition

Minimum size: 1 MiB Maximum size: 121202 MiB

Free space preceding (MiB): 1

New size (MiB): 119202

Free space following (MiB): 2000

Align to: MiB

Create as: Logical Partition

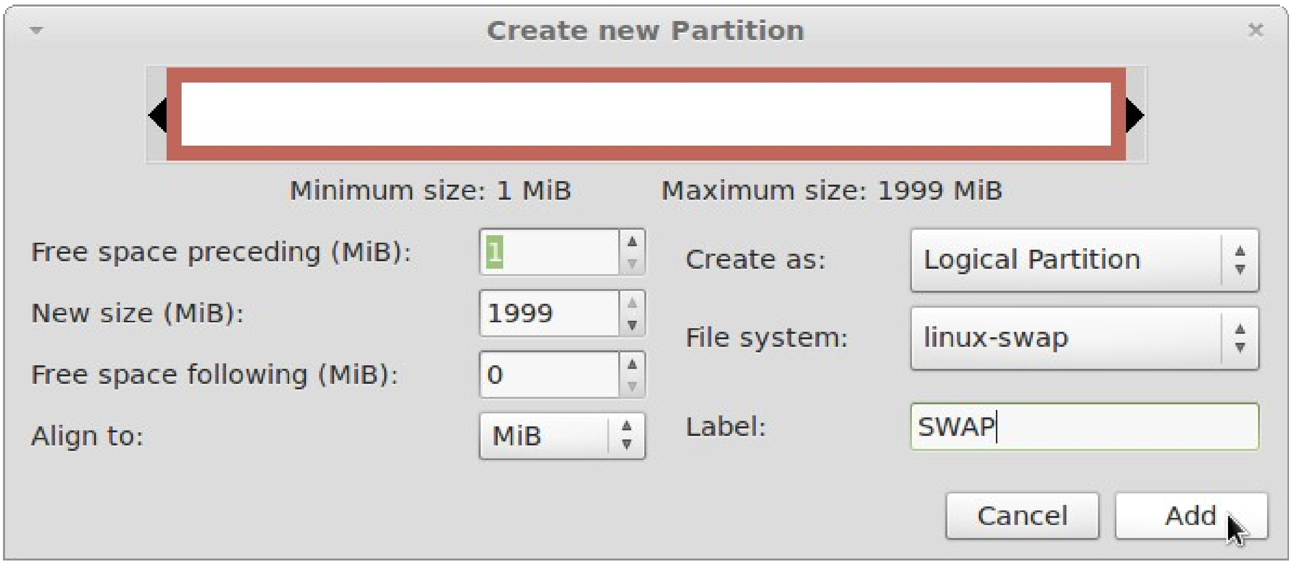
File system: ext4

Label: HOMES

Figure 12: Defining the home partition

Make swap area

Finally, the rest of the unallocated space at the end is assigned as swap, as in Figure 13.



Create new Partition

Minimum size: 1 MiB Maximum size: 1999 MiB

Free space preceding (MiB): 1

New size (MiB): 1999

Free space following (MiB): 0

Align to: MiB

Create as: Logical Partition

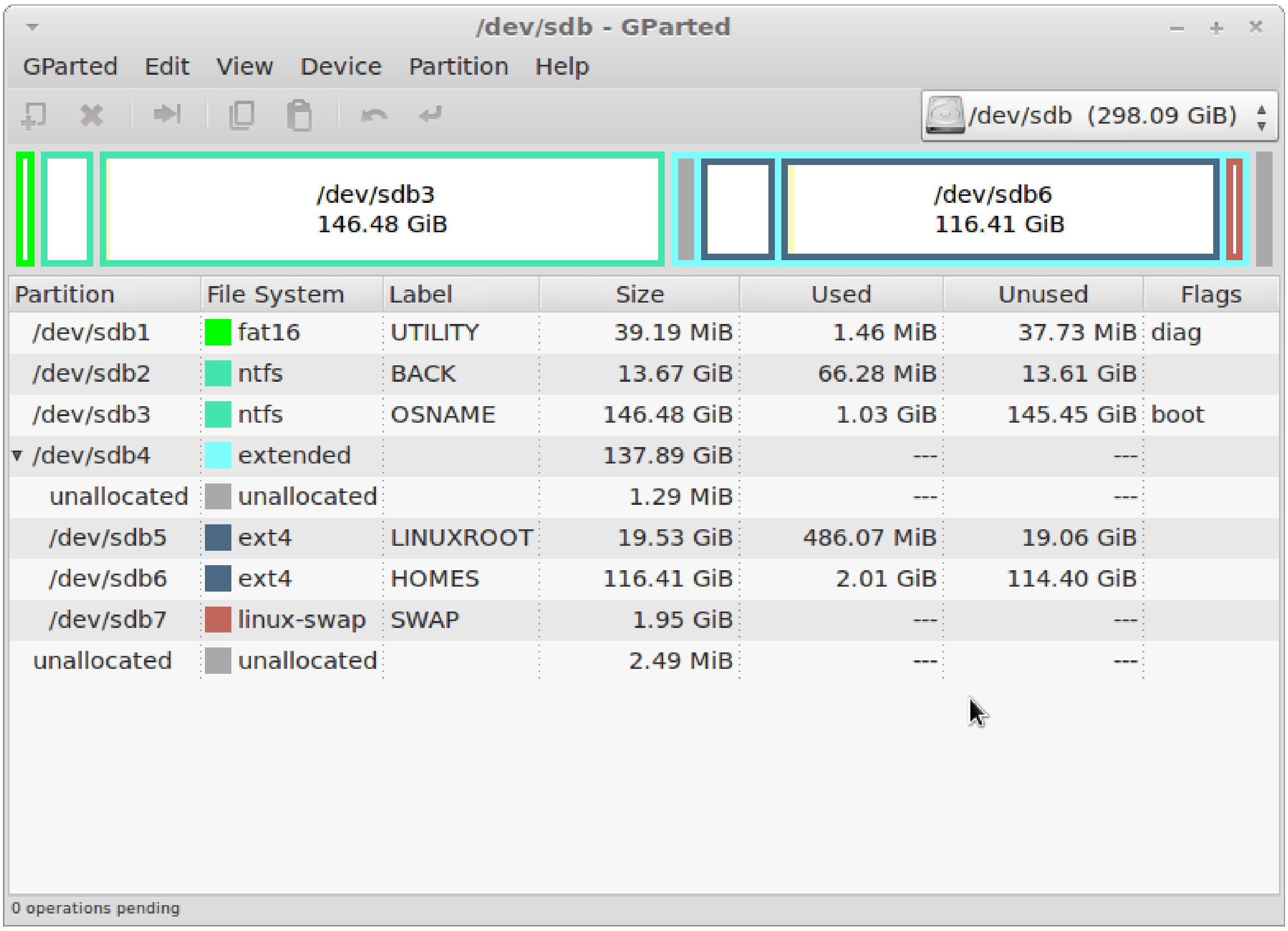
File system: linux-swap

Label: SWAP

Figure 13: Defining swap space

Final state

When everything has been done, the table looks like Figure 14.



/dev/sdb (298.09 GiB)

Partition	File System	Label	Size	Used	Unused	Flags
/dev/sdb1	🟩 fat16	UTILITY	39.19 MiB	1.46 MiB	37.73 MiB	diag
/dev/sdb2	🟩 ntfs	BACK	13.67 GiB	66.28 MiB	13.61 GiB	
/dev/sdb3	🟩 ntfs	OSNAME	146.48 GiB	1.03 GiB	145.45 GiB	boot
▾ /dev/sdb4	🟦 extended		137.89 GiB	---	---	
unallocated	⬜ unallocated		1.29 MiB	---	---	
/dev/sdb5	🟦 ext4	LINUXROOT	19.53 GiB	486.07 MiB	19.06 GiB	
/dev/sdb6	🟦 ext4	HOMES	116.41 GiB	2.01 GiB	114.40 GiB	
/dev/sdb7	🟥 linux-swap	SWAP	1.95 GiB	---	---	
unallocated	⬜ unallocated		2.49 MiB	---	---	

0 operations pending

Figure 14: Final layout

Next steps

The next steps will be to install Linux, which is beyond this paper. However, before you do, you will have to set the boot flag onto the Linux root partition, but at that point Windows will no longer boot until you've either completed the installation, or reset the flag. Only one partition can be marked as boot, and GParted will remove it from the old one when setting the new.

Further information

To know more on what partitions are and what they are used for in Linux especially, then visit http://tldp.org/HOWTO/html_single/Partition/

More on the subject is here: http://xpt.sourceforge.net/techdocs/nix/disk/general/disk03-PartitionPractices/single/

and some technical detail, like what restrictions exist for names of partitions, see https://help.ubuntu.com/community/RenameUSBDrive

More on GParted can be found at: http://gparted.sourceforge.net/faq.php

GParted can also be run from its own cut-down live CD obtainable from: http://gparted.sourceforge.net/livecd.php