How to format an external storage device using parted

How to format an external storage device using parted

Why Parted?

Because Parted is an advance, feature-rich and well-maintained partition editor
that can create, destroy, resize, rescue partitions and more.

Also, Parted can handle devices with more than 2T of memory unlike tools like fdisk.

Why would you want to change the format of a storage device?

Nowadays, storage devices are set to a file system called fat32 and while it has the advantage of being readable and writable from almost all operating systems, it is an old and inefficient format.

What is needed?

You'll need the utility parted:

```
$ sudo apt-get install parted
```

Let's get started

First, get root access:

```
$ sudo su
```

As an example I will use a completely erased USB with no partition table, so as not to skip any steps.

Now, let's detect where the device is:

```
# fdisk -l
```

The output, in my case was:

```
Disk /dev/sda: 320.1 GB, 320072933376 bytes
255 heads, 63 sectors/track, 38913 cylinders, total 625142448 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0000f43e

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048   608374783   304186368   83  Linux
/dev/sda2       608376830   625141759     8382465    5  Extended
/dev/sda5       608376832   625141759     8382464   82  Linux swap / Solaris

Disk /dev/sdb: 4008 MB, 4008706048 bytes
124 heads, 62 sectors/track, 1018 cylinders, total 7829504 sectors
Units = sectors of 1 * 512 = 512 bytes
```

```
Disk /dev/sdb doesn't contain a valid partition table
```

The last line shows that the device corresponds to

```
 /dev/sdb
```

.

Before continuing, you may unmount the device partitions, because Trisquel automatically mounts them. To see all file systems mounted on your system, use this command:

```
# mount
```

To see the file system type used by your mounted volumes:

```
# df -T
```

To unmount the device partition you want, enter:

```
umount /dev/DEVICE
```

Don't forget that you'll need all the partitions of the device unmounted in order to manipulate it.

Now, let's invoke Parted specifying the device, in my case is sdb:

```
# parted /dev/sdb
```

```
GNU Parted 3.2
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

with the help command you will see a list of commands for parted.

```
(parted) help
  align-check TYPE N                        check partition N for TYPE(min|opt)
        alignment
  help [COMMAND]                            print general help, or help on
        COMMAND
  mklabel,mktable LABEL-TYPE                create a new disklabel (partition
        table)
  mkpart PART-TYPE [FS-TYPE] START END      make a partition
  name NUMBER NAME                          name partition NUMBER as NAME
  print [devices|free|list,all|NUMBER]      display the partition table,
        available devices, free space, all  found partitions, or a particular
        partition
  quit                                      exit program
  rescue START END                          rescue a lost partition near START
        and END
  resizepart NUMBER END                     resize partition NUMBER
  rm NUMBER                                 delete partition NUMBER
  select DEVICE                             choose the device to edit
  disk_set FLAG STATE                       change the FLAG on selected device
  disk_toggle [FLAG]                        toggle the state of FLAG on selected
        device
  set NUMBER FLAG STATE                     change the FLAG on partition NUMBER
  toggle [NUMBER [FLAG]]                    toggle the state of FLAG on partition
        NUMBER
  unit UNIT                                 set the default unit to UNIT
  version                                   display the version number and
        copyright information of GNU Parted
```

Now execute the

```
print
```

command to look at the partiton table:

```
(parted) print
```

```
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:
```

As you can see, Parted detected no disk label in the device, if you already have a partition table -and you probably do- you can skip the next part if you are happy with your existing partition table, if you don't have one or you want to replace the existing one, please read on.

Create a new partition table

The command to make a new partition table is

```
mklabel
```

, but first, type the following to list all label-types avaliable:

```
(parted) help mklabel
```

this should be the output:

```
(parted) help mklabel
  mklabel,mktable LABEL-TYPE              create a new disklabel (partition
        table)

        LABEL-TYPE is one of: aix, amiga, bsd, dvh, gpt, mac, msdos, pc98, sun,
        loop
```

If you don't know what to choose, I suggest msdos, it's the most widely used and it should work on the majority of systems:

```
(parted) mklabel msdos
```

If you don't want to use msdos, gpt is a good alternative and it could also let you name partitions as you like.

Now type print in order to see the partition table:

```
(parted) print
```

```
Model: USB2.0 Flash Disk (scsi)
Disk /dev/sdb: 4009MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:
```

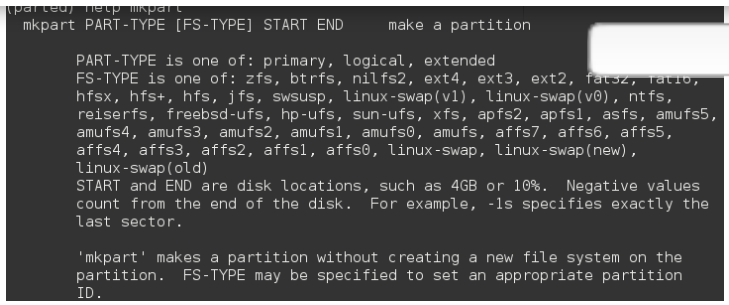
```

```

```
Number  Start  End  Size  Type  File system  Flags
```

Partitioning

The command for partitioning is called

```
mkpart
```



This is a delicate step, that is why the interactive function of parted will be used:

```
(parted) mkpart
```

First, the program will ask you about the partition type. Since we are going to set a single partition for the storage device we are going to choose

```
primary
```

:

```
Partition type?  primary/extended? primary
```

Then it will ask you about your file system type. the file system you choose depends on what you want, if you have a small USB like in the example and the only thing you want is that it has to be compatible with all systems, perhaps FAT32 is ok but if you want a file system that handles the data in a more elegant way, I suggest ext3 because:

1. ext3 can be manipulated also from other OS through, e.g nt2fsd.
2. You can easily convert ext3 to ext2 or to ext4.
3. ext3 has journalism active by default preventing file damage in cases like sudden shutdown.
4. ext3 techology can be used in 32T partitions on modern drives in front of 16T in FAT32.

```
File system type?  [ext2]? ext3
```

Now, the program will ask you where you want to start; which depends on what you want: if you start on the first sector, you will have more space in your partition, but it won't be aligned, which means that you will reduce performance in writing operations. Otherwise, you can have the partition aligned which means better performance.
I suggest having the partition aligned so if you specify a percentage, Parted will choose where to start in order to have it aligned:

```
Start? 0%
```

Now Parted will ask you where to end, if you put a negative number, parted will interpret it as the last sector on the drive:

```
End? -1s
```

where "s" means that we are talking about sectors.

```
Start? 2048s
```


Now type

```
print
```

to see your partition table:

```
(parted) print
Model: USB2.0 Flash Disk (scsi)
Disk /dev/sdb: 7829504s
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start  End       Size      Type     File system  Flags
 1      2048s  7829503s  7827456s  primary  ext3         lba
```

All should be fine now so type

```
quit
```

to exit from Parted.

Formating

Now is time to format the partition, when we specified de partition in Parted, we were setting the ID so to properly format in ext3, you should:

```
# mkfs.ext3 /dev/sdb1
```