How it works...

`pbzip2` internally uses the same compression algorithms as `bzip2`, but it compresses separate chunks of data simultaneously using pthreads - a threading library. However, this is all transparent to the user and all that happens is a much faster compression.

Just like `gzip` or `bzip2`, `pbzip2` does not create archives itself, it can only work on a single file. Hence, to compress multiple files and directories, we use it in conjunction with tar.

There's more...

There are other useful options we can use with `pbzip2`:

Manually specifying the number of CPUs

Use the `-p` option to `pbzip2` to specify the number of CPU cores manually. This is useful if the automatic detection fails or you want some CPU cores to be free for some other job.

```
pbzip2 -p4 myfile.tar
```

This will tell `pbzip2` to use 4 CPUs.

Specifying the compression ratio

Just like other compression tools we saw up to now, we can use the options from 1 to 9 to specify the fastest and best compression ratios respectively.

Creating filesystems with compression

`squashfs` is a heavy-compression based read-only filesystem that is capable of compressing 2 to 3 GB of data onto a 700 MB file. If you have ever used a Linux LiveCD (or LiveUSB), they are built using `squashfs` . These CDs make use of a read-only compressed filesystem which keeps the root filesystem on a compressed file. It can be loopback mounted and loads a complete Linux environment. Thus, when some files are required by processes, they are decompressed and loaded onto the RAM and used.

`squashfs` can be useful when it is required to keep files heavily compressed and to access a few of them without extracting all the files. This is because completely extracting a large compressed archive takes a long time. However, if an archive is loopback mounted, it will be very fast since only the required portion of the compressed archive is decompressed when requested. Let's see how we can use `squashfs`.

Getting ready

`squashfs` internally uses compression algorithms such as `gzip` and `lzma` and is supported in all modern Linux distros. However, in order to create `squashfs` files we need to install **squashfs-tools** using the package manager.

How to do it...

Let's see how to create and mount `squashfs` files:

1. In order to create a `squashfs` file by adding source directories and files, use:

   ```
   $ mksquashfs SOURCES compressedfs.squashfs
   ```

   Sources can be wildcards, or file, or folder paths.

   For example:

   ```
   $ sudo mksquashfs /etc test.squashfs
   Parallel mksquashfs: Using 2 processors
   Creating 4.0 filesystem on test.squashfs, block size 131072.
   [=========================================] 1867/1867 100%
   ```

   

   More details will be printed on the terminal. The output is stripped to save space

2. To mount the `squashfs` file to a mount point, use loopback mounting as follows:

   ```
   # mkdir /mnt/squash
   # mount -o loop compressedfs.squashfs /mnt/squash
   ```

You can access the contents at `/mnt/squashfs`.

There's more...

The `squashfs` file system can be customized by specifying additional parameters. Let's go through the additional options.

Excluding files while creating a squashfs file

While creating a `squashfs` file, we can exclude a list of files or a file pattern specified using wildcards.