In Bash 3.x and higher we have a new operator <<< that lets us use a string output as an input file. Using this we can write the `done` line of the loop as follows:

```
done <<< "`find $path -type f -print`"
```

`${!statarray[@]}` is used to return the list of array indexes.

Using loopback files

Loopback filesystems are very interesting components of Linux-like systems. We usually create filesystems on devices (for example, disk drive partitions). These storage devices are available as device files such as `/dev/device_name`. In order to use the storage device filesystem, we mount it at a directory called a **mount point**. On the other hand, loopback filesystems are those that we create in files rather than a physical device. We can then mount those files as filesystems at a mount point. This essentially lets you create logical "disks" inside a file on your physical disk!

How to do it...

Let us see how to create an ext4 filesystem on a file of size 1 GB:

1. The following command will create a file that is 1 GB in size:

   ```
   $ dd if=/dev/zero of=loobackfile.img bs=1G count=1
   1024+0 records in
   1024+0 records out
   1073741824 bytes (1.1 GB) copied, 37.3155 s, 28.8 MB/s
   ```

   You can see that the size of the created file exceeds 1 GB. This is because the hard disk is a block device and, hence, storage must be allocated by integral multiples of blocks size.

2. Now format the 1 GB file to ext4 using the `mkfs` command as follows:

   ```
   $ mkfs.ext4 loopbackfile.img
   ```

3. Check the file type using the following command:

   ```
   $ file loobackfile.img
   loobackfile.img: Linux rev 1.0 ext4 filesystem data,
   UUID=c9d56c42-f8e6-4cbd-aeab-369d5056660a (extents) (large files)
   (huge files)
   ```

4. Now you can mount the loopback file as follows:

   ```
   # mkdir /mnt/loopback
   # mount -o loop loopbackfile.img /mnt/loopback
   ```

   The `-o loop` additional option is used to mount loopback filesystems.

   This is actually a short method where we don't have to manually attach it to any devices. But, internally it attaches to a device called `/dev/loop1` or `loop2`.

5. We can do it manually as follows:

   ```
   # losetup /dev/loop1 loopbackfile.img
   # mount /dev/loop1 /mnt/loopback
   ```

6. To umount (`unmount`), use the following syntax:

   ```
   # umount mount_point
   ```

   For example:

   ```
   # umount /mnt/loopback
   ```

7. Or, alternately, we can use the device file path as an argument to the `umount` command as:

   ```
   # umount /dev/loop1
   ```

Note that the `mount` and `umount` commands should be executed as a root user since it is a privileged command.

How it works...

First we had to create a file that will act as a loopback file. For this we used `dd`, which is a generic command for copying raw data. It starts copying data from the file specified in its `if` parameter to the file specified in its `of` parameter. Additionally, we instruct `dd` to copy data in blocks of size 1 GB and copy one such block, hence creating a file of size 1 GB. The `/dev/zero` file is a special file, which will always contain 0 if you read from it.

We then used the `mkfts.ext4` command to create an ext4 filesystem in the file. A filesystem is needed because it provides a way of storing files on a disk/loopback file.

Finally, we used the `mount` command to mount the loopback file to a **mountpoint** (`/mnt/loopback` in this case). A mountpoint makes it possible for users to access the files stored on a filesystem. Before executing the `mount` command, the mountpoint should be created using the `mkdir` command. We pass the option `-o loop` to mount to tell it that what we are passing to it is a loopback file.