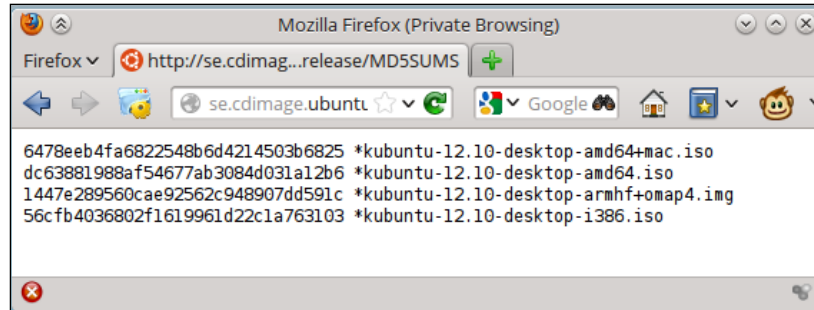


This is the md5sum checksum that is created:



There's more...

Checksums are also useful when used with a number of files. Let us see how to apply checksums to a collection of files and verify correctness.

Checksum for directories

Checksums are calculated for files. Calculating the checksum for a directory would mean that we would need to calculate the checksums for all the files in the directory, recursively.

It can be achieved by the `md5deep` or `sha1deep` command. Install the `md5deep` package to make these commands available. An example of this command is as follows:

```
$ md5deep -rl directory_path > directory.md5
# -r to enable recursive traversal
# -l for using relative path. By default it writes absolute file path in
output
```

Alternately, use a combination of `find` to calculate checksums recursively:

```
$ find directory_path -type f -print0 | xargs -0 md5sum >> directory.md5
```

To verify, use the following command:

```
$ md5sum -c directory.md5
```

Cryptographic tools and hashes

Encryption techniques are used mainly to protect data from unauthorized access. There are many algorithms available and we have discussed the most commonly used ones. There are a few tools available in a Linux environment for performing encryption and decryption. Sometimes we use encryption algorithm hashes for verifying data integrity. This section will introduce a few commonly used cryptographic tools and a general set of algorithms that these tools can handle.

How to do it...

Let us see how to use tools such as `crypt`, `gpg`, `base64`, `md5sum`, `sha1sum`, and `openssl`:

- ▶ The `crypt` command is a simple and relatively insecure cryptographic utility that takes a file from `stdin` and a passphrase as input and output encrypted data into `stdout` (and, hence, we use redirection for the input and output files):

```
$ crypt <input_file >output_file
```

Enter passphrase:


It will interactively ask for a passphrase. We can also provide a passphrase through command-line arguments:

```
$ crypt PASSPHRASE <input_file >encrypted_file
```

In order to decrypt the file, use:

```
$ crypt PASSPHRASE -d <encrypted_file >output_file
```

- ▶ `gpg` (GNU privacy guard) is a widely used tool for protecting files with encryption that ensures that data is not read until it reaches its intended destination. Here we discuss how to encrypt and decrypt a file.

 `gpg` signatures are also widely used in e-mail communications to "sign" e-mail messages, proving the authenticity of the sender.


In order to encrypt a file with `gpg` use:

```
$ gpg -c filename
```

This command reads the passphrase interactively and generates `filename.gpg`. In order to decrypt a `gpg` file use:

```
$ gpg filename.gpg
```

This command reads a passphrase and decrypts the file.

 We don't cover `gpg` in much detail in this book. If you're interested in more information, please see http://en.wikipedia.org/wiki/GNU_Privacy_Guard.