For example purposes here's a `goldenfile` that was created with some arbitrary timestamp.

```
$ touch -d 20120101 goldenfile
$ ls -l goldenfile
-rw-rw-r--. 1 saml saml 0 Jan  1  2012 goldenfile
```

Now I make some new file:

```
$ touch newfile
$ ls -l newfile
-rw-rw-r--. 1 saml saml 0 Mar  7 09:06 newfile
```

Now apply `goldenfile`'s attributes to `newfile`.

```
$ touch -r goldenfile newfile
$ ls -l goldenfile newfile
-rw-rw-r--. 1 saml saml 0 Jan  1  2012 newfile
-rw-rw-r--. 1 saml saml 0 Jan  1  2012 goldenfile
```

Now `newfile` has the same attributes.

You can change the modification time of a file using the `touch` command:

```
touch filename
```

By default this will set the file's modification time to the current time, but there are a number of flags, such as the `-d` flag to pick a particular date. So for example, to set a file as being modified two hours before the present, you could use the following:

```
touch -d "2 hours ago" filename
```

If you want to modify the file relative to its existing modification time instead, the following should do the trick:

```
touch -d "$(date -R -r filename) - 2 hours" filename
```

If you want to modify a large number of files, you could use the following:

```
find DIRECTORY -print | while read filename; do
    # do whatever you want with the file
    touch -d "$(date -R -r "$filename") - 2 hours" "$filename"
done
```

You can change the arguments to `find` to select only the files you are interested in. If you only want to update the file modification times relative to the present time, you can simplify this to:

```
find DIRECTORY -exec touch -d "2 hours ago" {} +
```

This form isn't possible with the file time relative version because it uses the shell to form the arguments to `touch`.

As far as the creation time goes, most Linux file systems do not keep track of this value. There is a `ctime` associated with files, but it tracks when the file metadata was last changed. If the file never has its permissions changed, it might happen to hold the creation time, but this is a coincidence. Explicitly changing the file modification time counts as a metadata change, so will also have the side effect of updating the `ctime`.