Dealing with line endings

MAC | WINDOWS | LINUX | ALL

If you're using Git to collaborate with others on GitHub, ensure that Git is properly configured to handle line endings.

Every time you press return on your keyboard you're actually inserting an invisible character called a *line ending*. Historically, different operating systems have handled line endings differently.

When you view changes in a file, Git handles line endings in its own way. Since you're collaborating on projects with Git and GitHub, Git might produce unexpected results if, for example, you're working on a Windows machine, and your collaborator has made a change in OS X.

Global settings for line endings

The `git config core.autocrlf` command is used to change how Git handles line endings. It takes a single argument.

On Windows, you simply pass `true` to the configuration. For example:

```
$ git config --global core.autocrlf true
# Configure Git on Windows to properly handle line endings
```

Per-repository settings

Optionally, you can configure the way Git manages line endings on a per-repository basis by configuring a special *.gitattributes* file. This file is committed into the repository and overrides an individual's `core.autocrlf` setting, ensuring consistent behavior for all users, regardless of their Git settings. The advantage of a *.gitattributes* file is that your line configurations are associated with your repository. You don't need to worry about whether or not collaborators have the same line ending settings that you do.

The *.gitattributes* file must be created in the root of the repository and committed like any other file.

A *.gitattributes* file looks like a table with two columns:

- On the left is the file name for Git to match.
- On the right is the line ending configuration that Git should use for those files.

Example

Here's an example *.gitattributes* file. You can use it as a template for your repositories:

```
# Set the default behavior, in case people don't have core.autocrlf set.
* text=auto

# Explicitly declare text files you want to always be normalized and converted
# to native line endings on checkout.
*.c text
*.h text
```

Article versions

GitHub.com
GitHub Enterprise 2.10
GitHub Enterprise 2.9
GitHub Enterprise 2.8
GitHub Enterprise 2.7

```
# Declare files that will always have CRLF line endings on checkout.
*.sln text eol=crlf

# Denote all files that are truly binary and should not be modified.
*.png binary
*.jpg binary
```

You'll notice that files are matched--`*.c`, `*.sln`, `*.png`--, separated by a space, then given a setting--`text`, `text eol=crlf`, `binary`. We'll go over some possible settings below.

`text=auto`
Git will handle the files in whatever way it thinks is best. This is a good default option.

`text eol=crlf`
Git will always convert line endings to `CRLF` on checkout. You should use this for files that must keep `CRLF` endings, even on OSX or Linux.

`text eol=lf`
Git will always convert line endings to `LF` on checkout. You should use this for files that must keep LF endings, even on Windows.

`binary`
Git will understand that the files specified are not text, and it should not try to change them. The `binary` setting is also an alias for `-text -diff`.

Refreshing a repository after changing line endings

After you've set the `core.autocrlf` option and committed a *.gitattributes* file, you may find that Git wants to commit files that you have not modified. At this point, Git is eager to change the line endings of *every file* for you.

The best way to automatically configure your repository's line endings is to first backup your files with Git, delete every file in your repository (**except the *.git* directory**), and then restore the files all at once.

1. Save your current files in Git, so that none of your work is lost.

```
$ git add . -u
$ git commit -m "Saving files before refreshing line endings"
```

2. Remove the index and force Git to rescan the working directory.

```
$ rm .git/index
```

3. Rewrite the Git index to pick up all the new line endings.

```
$ git reset
```

4. Show the rewritten, normalized files.

```
$ git status
```

5. Add all your changed files back, and prepare them for a commit. This is your chance to inspect which files, if any, were unchanged.

```
$ git add -u
# It is perfectly safe to see a lot of messages here that read
```

```
# "warning: CRLF will be replaced by LF in file."
```

6 Rewrite the `.gitattributes` file.

```
$ git add .gitattributes
```

7 Commit the changes to your repository.

```
$ git commit -m "Normalize all the line endings"
```

Further reading

- "Customizing Git - Git Attributes" from the Pro Git book
- "git-config(1) Manual Page"
- "Getting Started - First-Time Git Setup" from the Pro Git book
- "Mind the End of Your Line" - The full story of line endings in Git by Tim Clem

Contact a human