The CTIE Processor

Julian Gilbey
jdg@debian.org

Based on the original TIE documentation by:
Klaus Guntermann
TH Darmstadt
Fachbereich Informatik
Institut für Theoretische Informatik

January 2002, updated April 2003

1 Introduction

When installing a program on a particular computer system, certain system dependent changes are often needed. The original WEB programs (WEAVE and TANGLE) handled this problem by using *change files*, which work very much like a patch file and are read by WEAVE and TANGLE along with the main WEB file. This allows the original WEB file to remain intact and the particular system dependencies to be isolated.

Sometimes it is good practice to develop a set of multiple change files for a given WEB file to distinguish between different changes such as program enhancements, bug fixes, system dependent changes and output device dependent modifications. Additionally this allows combinations of changes that can be used with a set of programs that share some common features, such as WEAVE and TANGLE themselves.

Unfortunately, the processors TANGLE and WEAVE can handle only one change file. Rather than modifying these processors to handle multiple change files, it is easier to combine them in a preprocessing step. The TIE processor was written to perform this task. It is able to create either a new master file or a single change file that comprises the effect of all change files.

The CWEB system was later developed in order to open up the WEB system of literate programming to C programmers. It functions in an essentially identical way to the original Pascal WEB system, and the syntax of change files is the same. So it should be possible to use the TIE processor for CWEB source files. However, one significant enhancement was made to CWEB which makes this sometimes impossible. CWEB supports *include files*, which are introduced by a line beginning '`@i` *filename*', which includes the CWEB file specified, just like the C preprocessor command `#include`. As these include commands are effectively expanded before the change file is applied, both in the master file and the change file, it is clear that TIE needs to interpret `@i` commands in order to work for all CWEB files. Thus CTIE was born: it essentially combines the traditional behaviour of TIE with the ability to handle CWEB include files.

2 Application

The current version of CTIE accepts a master file and up to 32 change files. In general it is important to use the change files in the correct sequence, as later change files may modify the results of applying earlier changes. Thus the order of the change files on the command line is taken to specify the order in which the change files should be applied. Conceptually, each change file is applied to the master file in turn to create a new master file, to which the next change file is applied; in practice, though, a more space-efficient algorithm is used which only requires storage space for one line per file, so the lengths of the individual files or changes are irrelevant for storage requirements.

Note that CTIE will expand all @i include lines, including the specified files whether or not they are modified by any change files. So another application of CTIE is to simply expand all of these include files, producing an expanded master file. (Thanks to Hartmut Henkel <hartmut_henkel@gmx.de> for this observation.)

CTIE must be called with at least 3 parameters as follows:

```
ctie -[cm] outfile master changefile(s)
```

where the parameters are (in order):

1. Either the option -c or -m. This option determines whether an amalgamated change file or a new master file is created.
2. The name of the output file.
3. The name of the master file.
4. The name(s) of the change file(s), if any.

Unlike the CWEB programs CWEAVE and CTANGLE, no attempt to add default suffixes such as .w or .ch is made; the filenames must be the full filenames. If CTIE has been compiled with kpathsea support, then the kpathsea library will be used to search for the files using the CWEBINPUTS variable, otherwise just the current directory and the directory specified by the CWEBINPUTS environment variable will be searched, in that order.

Also, CTIE can be called with the single argument --help or --version to get basic help or version information.

3 Example

To illustrate the actions `CTIE` performs you may inspect the following example that exercises some of the borderline cases.

ctie.tie

```
line 1
line 2
line 3
line 4
line 5
@i ctie.inc
line 6
line 7
line 8
line 9
line 10
⟨EOF⟩
```

ctie.cf1

```
@x
line 2
line 3
@y
changed line 2
changed line 3
inserted line
@z

@x
line 7
@y
changed line 7
@z
⟨EOF⟩
```

ctie.cf2

```
@x
line 1
changed line 2
changed line 3
inserted line
line 4
@y
final line 2
final line 3
changed line 4
@z

@x
inc line 1
@y
changed inc line 1
inserted inc line
@z

@x
changed line 7
line 8
@y
final line 7
final line 8
@z
⟨EOF⟩
```

ctie.cf3

```
@x
changed line 4
@y
final line 4
@z

@x
changed inc line 1
inserted inc line
@y
final inc line 1
@i "ctie.inc2"
@z
⟨EOF⟩
```

The included files are as follows:

ctie.inc

```
inc line 1
inc line 2
@i ctie.inc1
inc line 3
⟨EOF⟩
```

ctie.inc1

```
included inc line
⟨EOF⟩
```

ctie.inc2

```
final inserted inc line
⟨EOF⟩
```

Using these input files and running `CTIE` with the following commands to create a new master file and a new change file will result in the following output files.

For the master file: `ctie -m ctie.outm ctie.cf1 ctie.cf2 ctie.cf3`
For the change file: `ctie -c ctie.outc ctie.cf1 ctie.cf2 ctie.cf3`

ctie.outm

```
final line 2
final line 3
final line 4
line 5
final inc line 1
final inserted inc line
inc line 2
included inc line
inc line 3
line 6
final line 7
final line 8
line 9
line 10
```

ctie.outc

```
@x
line 1
line 2
line 3
line 4
@y
final line 2
final line 3
final line 4
@z

@x
inc line 1
@y
final inc line 1
final inserted inc line
@z

@x
line 7
line 8
@y
final line 7
final line 8
@z
```