headache

Vincent Simonet

November, 2002

1 Overview

It is a common usage to put at the beginning of source code files a short header giving, for instance, some copyright informations. `headache` is a simple and lightweight tool for managing easily these headers. Among its functionalities, one may mention:

- Headers must generally be generated as *comments* in source code files. `headache` deals with different files types and generates for each of them headers in an appropriate format.
- Headers automatically detects existing headers and removes them. Thus, you can use it to update headers in a set of files.

`headache` is distributed under the terms of the *GNU Library General Public License*. See file `LICENSE` of the distribution for more information.

2 Compilation and installation

Building `headache` requires *Objective Caml* (version 3.06 or up, available at `http://caml.inria.fr/`) and *GNU Make*.

Instructions

1. Configure the system. From the source directory, do:

   ```
   ./configure
   ```

   This generates the `Makefile` in the source directory. The software is installed by default in `/usr/local/bin`. This path is customizable thanks to the `--bindir` option.

2. Build the executable. From the source directory, do:

   ```
   make
   ```

   This builds an executable named `headache`.

3. In order to install it in the directory specified during configuration, as a superuser, do:

   ```
   make install
   ```

3 Usage

Let us illustrate the use of this tool with a small example. Assume you have a small project mixing C and Caml code consisting in three files `foo.c`, `bar.ml` and `bar.mli'`, and you want to equip them with some header. First of all, write a *header file*, i.e. a plain text file including the information headers must mention. An example of such a file is given in figure 1. In the following, we assume this file is named `myheader` and is in the same directory as source files.
Then, in order to generate headers, just run the command:

```
                            Headache
              Automatic generation of files headers

         Vincent Simonet, Projet Cristal, INRIA Rocquencourt

Copyright 2002
Institut National de Recherche en Informatique et en Automatique.
All rights reserved.  This file is distributed under the terms of
the GNU Library General Public License.

Vincent.Simonet@inria.fr          http://cristal.inria.fr/~simonet/
```

Figure 1: An example of header file

```
headache -h myheader foo.c bar.ml bar.mli
```

Each file is equipped with an header including the text given in the header file `myheader`, surrounded by some extra characters depending on its format making it a comment (e.g. `(*` and `*)` in `.ml` files). If you update informations in the header file `myheader`, you simply need to re-run the above command to update headers in source code files: existing ones are automatically removed.
Similarly, running:

```
headache -r foo.c bar.ml bar.mli
```

removes any existing in files `foo.c`, `bar.ml` and `bar.mli`. Files which do not have a header are kept unchanged.

4 Configuration file

File types and format of header may be specified by a *configuration file*. By default, the default builtin configuration file given in figure 2 is used. You can also use your own configuration file thanks to the `-c` option:

```
headache -c myconfig -h myheader foo.c bar.ml bar.mli
```

In order to write your own configuration, you can follow the example given in figure 2. A configuration file consists in a list of *entries* separated by the character `|`. Each of them is made of two parts separated by an `->`:

- The first one is a *regular expression*. (Regular expression are enclosed within double quotes and have the same syntax as in Gnu Emacs.) `headache` determines file types according to file basenames; thus, each file is dealt with using the first line its name matches.
- The second one describes the format of headers for files of this type. It consists of the name of a *model* (e.g. `frame`), possibly followed by a list of arguments. Arguments are named: `open:"(*"` means that the value of the argument `open` is `(*`.

`headache` currently supports three *models*:

- `frame`. With this model, headers are generated in a frame. This model requires three arguments: `open` and `close` (the opening and closing sequences for comments) and `line` (the character used to make the horizontal lines of the frame). Two optional arguments may be used `margin` (a string printed between the left and right side of the frame and the border, by default two spaces) and `width` (the width of the inside of the frame, default is 68).
- `lines`. Headers are typeset between two lines. Three arguments must be provided: `open` and `close` (the opening and closing sequences for comments), `line` (the character used to make the horizontal lines). Three optional arguments are allowd: `begin` (a string typeset at the beginning of each line, by default two spaces), `last` (a string typeset at the beginning of the last line) and `width` (the width of the lines, default is 70).

```
# Objective Caml source
  ".*\\.ml[il]?" -> frame open:"(*" line:"*" close:"*)"
| ".*\\.mly"     -> frame open:"/*" line:"*" close:"*/"
# C source
| ".*\\.[ch]"    -> frame open:"/*" line:"*" close:"*/"
# Misc
| ".*Makefile.*" -> frame open:"#"  line:"#" close:"#"
| ".*README.*"   -> frame open:"*"  line:"*" close:"*"
| ".*LICENSE.*"  -> frame open:"*"  line:"*" close:"*"
```

Figure 2: The default builtin configuration file

- `no`. This model generates no header and has no argument.

It is possible to change the default builtin configuration file at compile time. For this, just edit the file `config_builtin` present in the source distribution before building the software.