

e-listing: Pretty Listing Maker, version 1.04

Karel Kubat, e-tunity
e-tunity

2003 ff.

Contents

1	Introduction	1
2	Using e-listing	2
3	Installation	3

1 Introduction

e-listing is a program that parses an input file, and produces a pretty listing on the standard output stream *stdout*. The program is mainly used to include listings of source programs in HTML or in printable output generated by \LaTeX .

To produce such output, *e-listing* knows what comment is, what strings are, and what 'words' (identifiers) are. *e-listing* is generic enough to be used on a multitude of input formats, such as: shell scripts, Perl programs, C/C++ sources, PHP sources. Files that are not suitable for *e-listing* are sources with 'unbalanced' quotes or double quotes (such as *m4* scripts).

When parsing a source file, *e-listing* follows these rules:

- All lines are preceded by their line number. Tabs in the source are handled as space-fills up to the next 8-column stop.
- Comment is anything that starts with # or // and ends on the same line, or anything that is enclosed in /* and */. (Note that therefore *#include* statements in C/C++ will be shown as comments.)
- Strings are enclosed by double quotes or by single quotes, and may contain any characters, including newlines and escaped quotes (such as \" and \').
- Words are sequences that begin with a letter-character (a..z) or an underscore, and optionally contain more letters, number-characters or underscores. Such a word may be an identifier in the given programming language, though *e-listing* doesn't know about identifiers.
- All input that doesn't fall into the above categories is treated as 'miscellaneous'.

Though this approach may look simplistic, it will correctly interpret most sources.

Depending on an 'output mode', selected during the invocation of *e-listing*, the generated listing is formatted for a given destination (be it the web, or a printed document). The most often used mode is probably HTML, which is also the default mode. To see an overview of available output modes, invoke *e-listing* without arguments. The usage information shows what's supported.

2 Using e-listing

This section shows some examples of the usage of *e-listing*.

HTML Generation: The invocation *e-listing inputfile* generates an output listing on *stdout*, formatted for HTML (which is the default output mode). Such output is ready for inclusion in an HTML document, as illustrated in the hypothetical PHP page below, which might reside on some webserver and show the listing of a source *myfile.c* when invoked:

```
<html>
<head>
  <title>Listing of myfile.c</title>
</head>
<body>
  <h1>Listing of myfile.c</h1>
  <hr>

  <script language="php">
    system ("e-listing myfile.c");
  </script>

  <hr>
</body>
</html>
```

The invocation *e-listing myfile.c* implies the flag *-m html*; the selection of HTML output mode.

Using Stylesheets: The HTML elements in the output are enclosed in `` tags, specifying *classes*:

- *comment* for comments,
- *lineno* for line numbers,
- *string* for "strings",
- *charstr* for 'strings',
- *ident* for words (identifiers).

All code is furthermore enclosed in `<code>` tags. This allows the creation of a custom CSS stylesheet that modifies the look of the listing when viewed through a browser.

e-listing can output a sample stylesheet (which can further be edited) when the flag *-s* is given.

Generating stand-alone HTML documents: When *e-listing* is invoked with the flag *-h*, a full HTML document is written (i.e., including the `<head>`, `<body>` and so on).

As an example, consider the following PHP source, in which *e-listing* is responsible for the generation of all HTML document tags and the default stylesheet:

```
<script language="php">
system ("e-listing -hs myfile.c");
</script>
```

Documents for L^AT_EX: When *e-listing* is invoked with the flag *-m latex*, then L^AT_EX output is generated. (Internally, the listing is typeset in a `tabbing` environment.) Additionally, flag *-h* instructs *e-listing* to enclose the listing in its own document header and footer. Hence,

- *e-listing -mlatex -h myfile.c > myfile.latex* generates a file *myfile.latex* that can be processed using LaTeX(),
- *e-listing -mlatex myfile.c > myfile.latex* generates a file *myfile.latex* that can be included in an existing L^AT_EX document.

Using e-listing with Yodl: *Yodl* is a document preprocessor, which can generate several output formats from one document specification in the Yodl format. To use *e-listing* in Yodl, prepare your listings before calling Yodl, e.g., using

```
e-listing myfile.c > tmp.html
e-listing -mlatex myfile.c > tmp.latex
```

Then place the following directives in a Yodl document:

```
whenlatex(includeverbatim(tmp.latex))
whenhtml(includeverbatim(tmp.html))
```

A sample is included below, generated using exactly this syntax. It is b.t.w. the *main()* function of the program *e-listing*.

```
INCLUDENOEXPAND(test.latex)
```

3 Installation

The installation of *e-listing* is very straight-forward, the following steps are required.

- Obtain the distribution archive *e-listing.tar.gz* from e-tunity's public site <http://public.e-tunity.com>.
- Extract the archive in one of your source directories. The contents are spilled into a created directory *e-listing/*.
- Make sure that you have the e-tunity package *misc* installed. Small tools from this package, e.g., a version ID checker, are used.
- Make sure that you have the e-tunity package *e-lib* correctly installed. *e-listing* uses functions of this library. Also make sure that the 'standard' e-tunity environment is present, i.e., with the environment variables *ELIBDIR*, *EBINDIR* and *EINCDIR*.
- Change-dir to that directory. Do a `make install`.