NAME

espretty − Eiffel source pretty printer

SYNOPSIS

espretty <filename> [**−lnsux**] [**−t | −2 | −3 | −4**] [**−ascii | −ansi | −html | −latex | −mif | −mime | −rtf**]

DESCRIPTION

espretty can be used to produce ‘pretty-printed’ versions of Eiffel class text files and thus make these easier to read.

The *espretty* program understands a substantial amount about the syntax of Eiffel, but it doesn’t make any attempts to cope with incomplete and misformed syntax. However, whenever possible it tries to format the class text according to the layout standards in ETL Appendix A.

espretty parses the Eiffel class file given as an argument and prints the formated result on the standard output. This version for the *SmallEiffel* compiler of *espretty* can only process one file argument per run and has not the ability to read from STDIN (this means you can’t pipe in a file).

If you want to handle multiple input files at once, you have to use the supplied *espp* shell script, which is a sort of preprocessor or frontend for *espretty.*

espp has the same options as *espretty* but allows the handling of multiple input files at once.

The program’s execution can be interrupted at any time by pressing the *BREAK key* (Ctrl-C) under most Unix systems.

OPTIONS

−t | −2 | −3 | −4

Indent levels by *NUM* blank characters, the *default* is *3.* This conforms to the code examples shown in ETL. The *NUM* argument should be one of those listed above. Use *−t* (8 blanks) to emulate a tabstop. However, *espretty* doesn’t check for other values, so it is the responsibility of the user to provide one of those *NUM* arguments shown above.

−ascii | −ansi | −html | −latex | −mif | −mime | −rtf

Use one of these formatters for output, the *default* is *ASCII.* Available formatters are:

ASCII

Text is printed to STDOUT using plain ASCII with no embellishments.

ANSI

As ASCII, but with keywords emboldened with the relevant ANSI escape sequences. For use if you have a terminal that supports ANSI bold codes under Unix, or under DOS if have included the ANSI.SYS driver in your DOS CONFIG.SYS file. This format is supported by Unix pagers like *more , less or most.*

HTML

Outputs the text including a minimal subset of HTML sequences. This format is recognized by WWW-browsers like Netscape etc. and a lot of HTML-editors. With this formatter you should be able to present typeset Eiffel classes via WWW.

LaTeX

Outputs the text including a minimal subset of LaTeX sequences. This format is recognized by the LaTeX Document Preparation System. With this formatter you should be able to present typeset Eiffel classes in LaTeX.

MIF

Outputs the text including a minimal subset of FrameMaker Maker-Interchange-Format sequences. This format is recognised by the FrameMaker publishing software. If you load the output you should be able to print typeset Eiffel classes. Notice that *espretty's* implementation of this formatter has some *restrictions* and *limitations in* the proper *Eiffel comment handling* , for more information about this see also the *CAVEATS section.*

MIME

Outputs the text including RFC 1341 Rich-Text sequences. If you have a MIME richtext reader or you wish to mail typeset Eiffel Classes to people, this formatter is for you.

RTF

Outputs the text including a minimal subset of Microsoft's Rich-Text-Format sequences. This format is recognised by a lot of WordProcessors. If you load the output you should be able to print typeset Eiffel classes. Notice that *espretty's* implementation of this formatter has some *restrictions* and *limitations in* the proper *Eiffel comment handling* , for more information about this see also the *CAVEATS section.*

–l When this option is set, Eiffel identifiers (not types) are transformed into lower chars, this conforms to the ETL Appendix A layout guidelines.

–n Print output with line numbers and statement level, empty lines are ignored. This is not a bug, it's a feature. If combined together with the -x option, the output of empty lines without line numbers can be reduced.

–s Print output in a sort of short format, removes feature bodies.

–u When this option is set, (most) Eiffel types are transformed into upper chars, this might reduce the amount of shift or caps-lock presses under your keyboard.

–x Start new line after 'redefine, until ...', removes empty lines in routine bodies.

EXAMPLE USAGE

The output of an input class file appears by default at STDOUT! To write the *espretty* output into a file, you have to use STDOUT redirection. For example:

```
espretty class1.e  > myclass.e
```

or to append to myclasses.e:

```
espretty class7.e >> myclasses.e
```

or for documentation purposes the short form in RTF format:

```
espretty class1.e -s -rtf  > myclass.e
espretty class7.e -s -rtf >> myclasses.e
```

Of course there are much more possibilities, try them out.

SEE ALSO

espp(1),

[ETL92] **Bertrand Meyer, Eiffel: The language, Prentice Hall**

CAVEATS

The *RTF* , *HTML* , *LaTeX* , and *MIF* output formatters have some *limitations* in the *Eiffel comment* handling. So the user is forced to take care about the following *restrictions* when these formatters are used:

Eiffel names of ‘features’ or other ‘entity’ appearing in a comment *must* be enclosed in *exactly* these single quotes (one *opening quote* -> ‘ <- and one *closing quote* -> ’ <-), as shown here with *‘feature’* and *‘entity’* , to be printed correctly in *italics.*

Additional you have to *avoid generally* using this sort of single quotes for other purposes in Eiffel comments. This is *important* for the *RTF* , *HTML* , *LaTeX* , and *MIF* formatters, to produce a correct output format, otherwise the output of their format sequences might not be correct.

Ending comments of *routines* and *classes* can not be printed in *italics* until now with this version for *SmallEiffel.* So this doesn’t conforms to the ETL Appendix A layout guidelines. This might be corrected in a future release.

A correct example:

```
feature

  is_break (ch: CHARACTER): BOOLEAN is
                         -- Check if break-character is ‘ch’ <--- (correct
    do                                     quotes)
      Result := true
    end -- is_break
```

A wrong example:

```
feature
                                      (wrong quotes)
  is_break (ch: CHARACTER): BOOLEAN is          |
      -- Check if break-character is ’ch’ <----
      -- Control characters aren’t allowed for ‘ch‘
    do                      ˆ              ˆ
      Result := false       |              |
    end -- is_break         |______ (wrong quotes)
```

BUGS

espretty’s argument handler only recorgnizes options if they are listed, after the supplied file argument. For example:

```
espretty -l class.e      <--- (wrong)
```

won’t work, it has to be called in the following order:

```
espretty class.e -l      <--- (correct)
```

AUTHOR

Valentino Kyriakides