NAME

tgrind − typeset nice program listings using TeX

SYNOPSIS

tgrind [**−?**] [**−d** *description-file*] [**−f**] [**−fn** *fontfamily*] [**−h** *header*] [**−i** *increment*] [**−l** *language*] [**−n**] [**−o** *basename*] [**−p**] [**−q**] [**−r**] [**−v**] [**−x**] [**−**] filename ...

DESCRIPTION

tgrind formats program sources in a nice style using **tex**(1).

Comments are placed in italics, keywords in bold face and strings in typewriter font.

Source file line numbers normally appear in the left margin every 10 lines. but the numbering frequency, and side of the page, are under user control.

The start of a function is indicated by the function name in large type in the right margin.

tgrind is *not* a prettyprinter: all line breaks and horizontal spacing in the input source files are preserved. Prettyprinters require more sophistication, and language-specific knowledge, than is possessed by **tgrind**. Consequently, you may find it useful to apply a prettyprinter to your source code before giving it to **tgrind**. Some of the available ones are: **bibclean**(1) for BIBTEX, **bstpretty**(1) for BIBTEX style language files, **cb**(1) and **indent**(1) for C and C++, **pretty**(1) for Fortran, **sf3pretty**(1) for Fortran and Sftran3, **indent-sexp** for GNU Emacs Lisp, **mft**(1) for Metafont, and **pform**(1) and **pindent**(1) for Pascal.

In regular mode, **tgrind** processes its input file(s) and passes them to **tex**(1) for formatting and output, and then sends the output to a DVI driver for conversion to POSTSCRIPT.

For privacy, all output files are normally left in the current directory with numeric base names of the form *tg_12345.tex*; such names conform to the ISO 9660 standard for filenames on CD-ROMs (*Volume and File Structure of CD-ROM for Information Interchange, ISO 9660:1988(E)*), for maximal portability across current operating systems. The **−o** option (see below) can change this behavior. **tgrind** will *not* overwrite existing files.

In format-only mode (i.e., when the **−f** flag is used), **tgrind** processes its input file(s) and writes the result to standard output. This output can be saved for later editing, inclusion in a larger document, etc.

With version 3.00, **tgrind** has been redesigned so that its output is suitable for use with plain TEX, as before, and also with major macro packages, such as AMSTEX and LATEX 2.09 and 2e.

The output file can be typeset directly by plain TEX as a stand-alone document, but it can also be `\input` into an AMSTEX, LATEX, or plain TEX document and typeset as a fragment of the surrounding document.

To do this with AMSTEX and plain TEX, just use

```
\let \EndTgrind = \endgroup
```

before including **tgrind** output; this replaces the default definition of that macro, which would otherwise end the job.

For LATEX documents, the style file *tgrind.sty* provides suitable definitions, and can be used like this:

LATEX 2.09:

```
\documentstyle[tgrind]{...}
```

LATEX 2e:

```
\documentclass{...}
\usepackage{tgrind}
```

You may prefer to make a private modification of this style file and save it under a different name. The distributed version suppresses the output of the `\TgrindFile` macro, but a comment in the style file shows how to make it invoke a sectional heading command instead. Since the heading level chosen depends on the major document style or class chosen, and also on the style of the particular document, there is no reasonable heading level that could be used in a generic *tgrind.sty* file.

OPTIONS

On IBM PC DOS and DEC (Open)VMS, the leading ‘‘−’’ on option names may be replaced by a slash, ‘‘/’’; however, the ‘‘−’’ option prefix is always recognized.

Option values are always provided as *separate* arguments following the option name.

Letter case in option names is *not* significant, although it may be in option values.

Any argument that begins with a hyphen is expected to be an option, and will raise an error if it is not recognized. If a filename begins with a hyphen, you therefore need to disguise it by supplying a leading directory path. For example, *./−foo* represents the file named *−foo* in the current directory in UNIX.

The options are:

−? Display a brief usage summary on *stderr*, and exit immediately with a success return code.

This same usage summary is given when an unrecognized option is encountered.

−d *description-file*
Specify the language definitions file (default is */usr/local/lib/tex/inputs/vgrindefs*). This option is useful for testing new language definitions.

−f Force format-only mode. This simply skips the TEX and DVI driver steps and optional printer step, copies the temporary output TEX file to *stdout*, and deletes the temporary output TEX file.

−fn *fontfamily*
Define the font family to be used in the output. The default font family if this option is not specified is Computer Modern. Otherwise, all fonts are virtual fonts that map to POSTSCRIPT Type 1 outline fonts, making the POSTSCRIPT output by a DVI driver more compact, and completely resolution-independent (i.e., no character bitmaps). The *fontfamily* values recognized may be given as a long font family name, or as a short two- or three-letter font file prefix. The family names currently recognized are:

- *AvantGarde* (*pag*),
- *Bookman* (*pbk*),
- *Charter* (*bch*),
- *ComputerModern* (*cm*),
- *Courier* (*pcr*),
- *Helvetica* (*phv*),
- *HelveticaNarrow* (*phn*),
- *NewCentury* or *NewCenturySchoolbook* (*pnc*),
- *Palatino* (*ppl*),
- *Times* (*ptm*), and
- *Utopia* (*put*).

All of these fonts, except *Charter* and *Utopia*, and sometimes, *HelveticaNarrow*, are resident in standard POSTSCRIPT laser printers. The exceptional fonts will be included in the output POSTSCRIPT by the DVI driver program, if the driver’s *psfonts.map* file correctly identifies them as non-resident fonts.

When this option is provided, the POSTSCRIPT output will also use *Courier* for a typewriter font, and *Symbol* for certain special characters; both of these fonts are printer resident.

−h *text* Specify text to go on the left top margin of every output page (default is none).

−i *increment*
Specify an alternate line number increment, overriding the default of 10.

−l *language*

Specify the language of the input file(s). The *language* name often has two or three acceptable forms, one of which is the standard source file extension in UNIX; see the bar-separated names beginning each language entry in the *vgrindefs* file; these synonyms are excluded from the list below. Here are the *language* names currently recognized:

a68 Motorola 68xxx assembly language.

ada Ada.

asm68 Another Motorola 68xxx assembly language.

awk **awk**(1), **gawk**(1), and **nawk**(1).

bash GNU Bourne-Again shell (**bash**(1)).

bibtex BIBTEX (**bibtex**(1)).

bst BIBTEX (**bibtex**(1)) style file language.

c C (the default language).

caml CAML.

c++ C++ and Objective C.

csh C shell (**csh**(1)).

ELisp GNU Emacs Lisp. Keywords are considered to be all of those low-level Lisp functions that are implemented in the Lisp interpreter itself (in the C programming language); higher-level Lisp functions written in Lisp are not keywords.

f Fortran.

i ISP.

I Icon.

ISP Unknown language on NeXT systems.

ksh Korn shell (**ksh**(1)).

LDL Unknown language.

latex LATEX 2.09. Keywords are considered to be all of the control sequences named in the index of the first edition of Leslie Lamport, *LATEX User's Guide and Reference Manual*, Addison-Wesley (1985), ISBN 0-201-15790-X. As with Emacs Lisp and other extensible languages, it seems reasonable to distinguish built-in 'system' commands from 'user' commands.

m MODEL.

m2 Modula-2.

maple Maple V. Keywords include the language keywords, operators, constants, and standard global variables.

maplex Extended Maple V. The keywords also include all of the initially-loaded library functions.

matlab Matlab.

Miranda Miranda.

ml MLisp and Emacs Lisp.

objc Objective C.

p Pascal.

prolog Prolog.

ps PostScript.

r Ratfor.

russell Russell.

sh Bourne shell (**sh**(1)).

sf3 Sftran3.

src Unknown source code (no keywords, comments, or strings are recognized).

tcl Tool Command Language (**tcl**(1)).

tcsh Extended C shell (**tcsh**(1)).

tex TEX.

y yacc.

−n Do not display keywords in boldface.

As a side effect, this option suppresses the marginal procedure headers and thus also the index. This may be changed in a future version.

−o *basename*

Specify an alternate basename for the output files. The default is *tg_nnnnn*, where *nnnnn* is obtained from the current time. This option allows you to direct the output files anywhere you like. However, **tgrind** will still refuse to run if they already exist, in order to avoid overwriting a possibly-important file.

−p Send the output POSTSCRIPT to the default printer.

−q Quiet mode: suppress informational messages normally sent to *stderr* or *stdout*. Error messages to *stderr* are never suppressed.

−r Print line numbers on the *right* instead of on the left. Older versions of **tgrind** always numbered on the right, but this proved inconvenient because numbers were sometimes overwritten by program text, and other times, were so far from the text that it was difficult to match them visually. The new default of line numbering on the left eliminates both of these problems.

−v Display the program version number and date on *stderr*, and exit immediately with a success return code.

−x Suppress the multicolumn index at the end of the program listing.

− Take input from standard input.

BUGS

The marginal−function−name mechanism depends on the quality of the language description in *vgrindefs*. The distributed *vgrindefs* file fails to recognize many legal programming-language function declarations.

The **−f**, **−o**, and **−x** flags mean different things to **tgrind** and **vgrind**(1).

PORTABILITY ISSUES

The table-driven preprocessor program, **tfontedpr**, is written in highly-portable C, and can be compiled with old (K&R) style C compilers, as well as with ANSI/ISO Standard C and C++ compilers, and requires only a small number of universally-available header files. It compiles and runs on numerous UNIX systems, and should be readily portable to other operating systems. Its memory requirements are modest: less than 100KB on typical UNIX systems.

tgrind is a simple front-end to **tfontedpr**, which is rarely invoked directly by users. **tgrind** handles argument parsing, and invocation of the preprocessor, indexer, TEX, DVI driver, and print spooler.

For portability, **tgrind** too is written in C, although the distribution also includes implementations as UNIX **sh**(1) and **csh**(1) scripts.

Installation of both programs uses the GNU **autoconf**(1) system for very simple one-line installation on any UNIX system.

The indexing program, *tgrindex.awk*, is written in **nawk**(1), and can be readily handled by GNU **gawk**(1) as well. Commercial and freely-distributable implementations of these languages are available for several personal computer operating systems, and for DEC (Open)VMS.

Volunteers for ports of **tgrind** to other operating systems will be most welcome!

ADDING SUPPORT FOR NEW LANGUAGES

The language translations implemented by **tgrind** are entirely table driven, using language descriptions given in the *vgrindefs* file, which is modeled after the colon-delimited *key=value* format of UNIX **printcap**(4) and **termcap**(4) capability files. Adding support to **tgrind** for a new language requires only additions to this file. Most language entries average about 9 lines, of which 6 or 7 are usually just enumerations of the language keywords.

Keys are either Boolean flags, in which case they take no *=value* string (the flag is set *true* if the key is present, and *false* if it is absent), or else string variables whose values are specialized patterns, jokingly referred to as *irregular expressions*, vaguely similar to the regular expressions recognized by the UNIX **ex**(1) editor and **lex**(1) lexical-analyzer generator.

In **tgrind** patterns, the characters '$', '(', ')', ':', '?', '^', '|', and '\' are reserved characters: they must be quoted with a preceding \ if they are to be interpreted as normal characters. Otherwise, they have these meanings:

^ Beginning of line.

$ End of line.

: Key-value capability pair delimiter.

\ Escape character. Two such characters, \\, represent a single backslash.

The extended patterns are:

\a Matches any number of characters (like '.*' in **lex**(1)).

\d Matches any number of whitespace delimiters (space, tab, newline, start of line).

\p Matches any number of alphanumeric characters. In a procedure definition (the **pb** key), the string that matches this symbol is used as the procedure name.

| Alternation.

() Grouping, used mostly for alternation and optionality.

? Last item is optional (i.e., occurs zero times, or one time).

\ Preceding any string means that the string will not match an input string if the input string is preceded by an escape character (\). This is typically used for languages (like C) that can include the string delimiter in a string by escaping it.

Unlike other implementations of regular expressions, these patterns match *words* and not characters. Hence something like *(foo|bar)mumble?* would match *foo*, *bar*, *foomumble*, or *barmumble*. In **tgrind** patterns, alternation binds very tightly, so grouping parentheses are likely to be necessary in expressions involving alternation.

Here are the capability keys that are currently used in the *vgrindefs* file, and in the source code file, *tfontedpr.c*:

ab Alternate comment begin.

ae Alternate comment end.

bb Begin statement block.

be End statement block.

cb Comment begin.

ce Comment end.

ic Define extra characters that may appear as initial characters of procedure names (those that match *p*) and keywords, beyond the hard-wired defaults of letters, digits, and underscore. This supports languages that place restrictions on the initial characters of identifiers. If this key is not provided, then initial characters are treated the same as non-initial characters. This key does not exist in **vgrind**(1) implementations.

id Define extra characters that may appear in procedure names (those that match *p*) and keywords, beyond the hard-wired defaults of letters, digits, and underscore. This supports languages, like Lisp and TEX, that have a more extensive character set for identifiers. This key does not exist in older **vgrind**(1) implementations; it may have been introduced first by Sun Microsystems in the Solaris 2.x operating system release.

kw Language keywords (a space-separated list, usually in alphabetical order for readability, though that is not a requirement).

lb Literal string begin.

le Literal string end.

nc Define characters that may *not* appear as initial characters of procedure names (those that match *p*) and keywords. This provides a way to remove initial identifier characters from the hard-wired defaults of letters, digits, and underscore. Its value is examined *after* any **ic** value. This key is *not* available in **vgrind**(1).

ni Define characters that may *not* appear in procedure names (those that match *p*) and keywords. This provides a way to remove identifier characters from the hard-wired defaults of letters, digits, and underscore. Its value is examined *after* any **id** value. This key is unique to **tgrind**; it is *not* available in **vgrind**(1).

oc (Boolean) one case flag: letter case is not significant.

pb Procedure (function, subroutine) begin.

sb Character string begin.

se Character string end.

tc If this key appears, it *must* be last. Its value is the name of another *vgrindefs* entry that is looked up and appended to the end of the current entry, minus the initial entry names. That entry in turn may end with a **tc** key that refers to yet another entry, and so on, up to a limit of 32 (to catch unterminating loops). If the same key appears more than once in the constructed entry, only the first value is used. Thus, **tc** can be used to prepare minor variations on a basic language definition.

tl (Boolean) top lex flag: procedures may be defined only at top level, that is, nested procedures are *not* permitted.

The string value of **id** and **kw** is treated as an ordinary string, rather than a pattern: backslash has significance only at end-of-line, or before a colon.

Keys are always exactly two characters long, and the equals sign that separates them from their values must follow immediately, without intervening whitespace.

If you need a single backslash in a string, represent it like this: **:id=\\:**. **vgrind**(1), and older versions of **tgrind**, do not permit this, because their simplistic scan assumes that backslash-colon does not terminate the string. Alternatively, since backslash is significant only before colon and newline in **id** and **ni** strings, you could also write **:id=\a:**, since ‘a’ is already in the identifier character set.

Let's dissect a typical entry to see how this works:

```
modula2|mod2|m2:\
        :pb=(^\d?(procedure|function|module)\d\p\d|\(|;|\:):\
        :bb=\d(begin|case|for|if|loop|record|repeat|while|with)\d:\
        :be=\dend|;:\
        :cb={:\
        :ce=}:\
        :ab=\(*:\
        :ae=*\):\
        :sb=":\
        :se=":\
        :oc:\
        :kw=and array begin by case const definition div \
        do else elsif end exit export for from if \
        implementation import in loop mod module not of \
        or pointer procedure qualified record repeat \
        return set then to type until var while with:
```

Each line after the first conventionally begins with a tab, although this is not required, and if the next character is a colon, a key name follows. Terminal backslashes indicate line continuation.

Multiple *key=value* pairs can be given on one line, as long as they are separated by colons, so at the loss of readability, we could compact seven lines of this entry into just one, like this:

```
:cb={:ce=}:ab=\(*:ae=*\):sb=":se=":oc:\
```

The first line in our sample entry says that this language may be named `modula2`, `mod2`, or `m2` in the **tgrind −l** option.

The `pb` line says that a procedure definition begins a line with optional whitespace, followed by one of the keywords `procedure`, `function`, or `module`, followed by optional whitespace, followed by an alphanumeric procedure name. That name in turn may be followed by whitespace, an open parenthesis, a semicolon, or a colon, thanks to the tight binding of alternation. It would have been clearer to include grouping parentheses, writing `(\d|\(|;|\:)`.

The `bb` line says that a statement block starts with optional whitespace, and one of the keywords `begin ... with`, and the `be` line says a statement block ends with optional whitespace, followed by either the keyword `end`, or a semicolon.

The `cb` and `ce` lines say that comments are delimited by braces, and the `ab` and `ae` lines say that comments may also be delimited by `(* *)`.

The `sb` and `se` lines say that strings are delimited by quotation marks.

The `oc` flag says that letter case is not significant in names; this seems to be in error: Niklaus Wirth's *Programming in Modula-2*, Springer-Verlag (1983), ISBN 0-387-12206-0, says that upper and lower case letters are distinct.

Finally, the `kw` lines enumerate all of the Modula-2 language keywords, from `and` to `with`.

ADDING SUPPORT FOR NEW FONTS

Any **−fn** font name specified on the **tgrind** command line is written directly to the beginning of the output TEX file in the form

```
\def \TgrindFontFamily {NewCenturySchoolbook}
```

followed by the lines

```
\ifx \BeginTgrind \undefined
  \input tgrindmac
\fi
```

The interpretation of the font family name is handled entirely in the TEX file, *tgrindmac.tex*. For this example, a line in that file says

```
\ifstreq{\TgrindFontFamily}{NewCenturySchoolbook}
\def \TgrindFontFamily {pnc}\fi
```

This replaces the definition of `\TgrindFontFamily` with *pnc*. A few lines later, we find

```
\ifstreq{\TgrindFontFamily}{pnc} \setfonts pnc r ri b.\fi
```

When `\TgrindFontFamily` has the value *pnc*, `\setfonts` is executed with four arguments: the basename of the virtual font, and the suffixes to be added to it to name upright, italic, and bold fonts.

Thus, TEX will expect to find in its *TEXFONTS* search path the TEX font metric files *pncr.tfm*, *pncri.tfm*, and *pncb.tfm*, and the DVI driver will expect to find in the same search path the virtual font files *pncr.vf*, *pncri.vf*, and *pncb.vf*.

Besides these files, `\setfonts` will generate references to fonts *pcrro* and *psyr* for typewriter text and special symbols, so TEX will also need *pcrro.tfm* and *psyr.tfm*, and the DVI driver will need *pcrro.vf* and *psyr.vf*.

If all of the referenced fonts exist on the system, TEX and the DVI driver will handle the rest of the job automatically, and if you added two lines similar to the ones above to a private copy of *tgrindmac.tex* to define a new font family, you can stop reading this section now.

However, here's what goes on behind the scenes. The virtual font files contain references to the so-called 'raw' TEX font metric files, prefixed by a letter 'r', in this case, *rpcrro.tfm* and *rpsyr.tfm*. The correspondence between these raw font metric files and the actual long POSTSCRIPT font names, such as *Courier-Oblique* and *Symbol*, is made in the *psfonts.map* file, with lines like these:

```
rpcrro  Courier-Oblique
rpsyr   Symbol
...
rptmro  Times-Roman ".167 SlantFont"
...
putb0   Utopia-Bold <putb0.pfb
putbo0  Utopia-Bold ".167 SlantFont " <putb0.pfb
```

The first two simply identify the mapping between a file name and a font name. The third additionally specifies that the Times-Roman font is to be slanted to the right by one-sixth, to synthesize an oblique Times-Roman. The fourth tells the DVI driver that the font definition must be downloaded from the *putb0.pfb* Type 1 POSTSCRIPT binary font file, and the fifth specifies both a slant and a source file to be downloaded.

FONTS REQUIRED

In the absence of the **−fn** option, **tgrind** produces output that requires only fonts that are standardly preloaded in plain TEX, so that the output files can be typeset by any TEX installation. Specifically, the default fonts used are:

```
cmr7                        % margin line numbers
cmb10
cmti10
cmr10
cmsy10
cmtt10
cmbx10 scaled \magstep1     % page headers
cmr10 scaled \magstep2      % right margin proc names
```

Larger font sizes than the default 10pt sizes are usually undesirable for program listings, because their lines are already often rather long for normal typesetting conventions.

When a POSTSCRIPT font is specified, the sizes called for are

```
roman       at  7pt     % margin line numbers
bold        at 10pt
italic      at 10pt
roman       at 10pt
```

Symbol	at 10pt	
Courier-Oblique	at 10pt	
bold	at 12pt	% page headers
roman	at 14pt	% right margin proc names

The exact names for the *bold*, *italic*, and *roman* variants depend on the font chosen; the calls to the `\setfont` macro in *tgrindmac.tex* encode this information.

FILES

NB: The names of the default installation directories shown below can be changed at installation time, and may be different at your site.

tg_nnnnn.∗ default temporary files, where *nnnnn* is a generated number

/usr/local/lib/tex/inputs/multicol.tex
multi-column plain TEX macro package

/usr/local/lib/tex/inputs/psfonts.map
DVI driver POSTSCRIPT font mapping file

/usr/local/lib/tex/inputs/tgrindex.awk
indexing program

/usr/local/lib/tex/inputs/tgrindex.tex
indexing macro package

/usr/local/lib/tex/inputs/tgrindmac.tex
tgrind macro package

/usr/local/bin/tgrind
user-callable **tgrind** program

/usr/local/bin/tfontedpr
tgrind preprocessor program

/usr/local/lib/tex/inputs/vgrindefs
language descriptions

AUTHOR

Van Jacobson, Lawrence Berkeley Laboratory (based on **vgrind**(1) by Dave Presotto and William Joy of UC Berkeley).

Extensions for POSTSCRIPT fonts, procedure indexing, space after the **−l** option, the **−o** option, the **ic**, **id**, **nc**, and **ni** keywords, language support for Ada, awk, bash, BIBTEX, BIBTEX style file language, ANSI/ISO Standard C, ANSI/ISO Standard C++, CAML, ELisp, Fortran, ksh, LATEX, Maple, Matlab, MLisp, Miranda, Objective C, PostScript, Russell, Sftran3, and tcsh, plus major revisions of documentation and source distribution, by

Nelson H. F. Beebe
University of Utah
Department of Mathematics, 110 LCB
155 S 1400 E RM 233
Salt Lake City, UT 84112-0090
USA
Email: `beebe@math.utah.edu`, `beebe@acm.org`, `beebe@computer.org` (Internet)
WWW URL: `http://www.math.utah.edu/~beebe`
FAX: +1 801 581 4148
Tel: +1 801 581 5254

AVAILABILITY

tgrind is freely distributable. The definitive master archive at the maintainer's site is available at

`http://www.math.utah.edu/pub/tex/bib/index-table-t.html#tgrind`
`ftp://ftp.math.utah.edu/pub/tex/bib/tgrind-`*x.y*`.tar.gz`

where *x.y* is a version number (3.01 for this release)

You should also be able to find the most recent version in the Comprehensive TEX Archive Network (CTAN) collections; for a list of CTAN hosts, do

```
finger ctan@tug.org
```

The three main CTAN master sites at the time of writing are:

`ftp.dante.de` (Germany)
`ftp.tex.ac.uk` (England)
`ctan.tug.org` (Massachusetts, USA)

SEE ALSO

autoconf(1) **awk**(1), **bash**(1), **bibclean**(1), **bibtex**(1), **bstpretty**(1), **cb**(1), **csh**(1), **dpsexec**(1), **ex**(1), **gawk**(1), **gs**(1), **indent**(1), **ksh**(1), **lex**(1), **maple**(1), **matlab**(1), **mft**(1), **nawk**(1), **pageview**(1), **pform**(1), **pindent**(1), **postscript**(1), **pretty**(1), **printcap**(4), **sf3pretty**(1), **sh**(1), **tcsh**(1), **termcap**(4), **tex**(1), **vgrind**(1), **vgrindefs**(5), **xmaple**(1), **xsf3**(1).