README: Installation instruction for tgrind-3.00

Last update: **Sat Jun 20 08:08:51 1998**

Table of contents

Jump start

As with most GNUware, you can build, test, and install this program on most UNIX systems by these simple steps

```
csh et amici:
        setenv CC ...your favorite C or C++ compiler...
        ./configure && make all check install TEXINPUTS=...

sh et amici:
        CC=...your favorite C or C++ compiler...
        export CC
        ./configure && make all check install TEXINPUTS=...
```

Or in *one* line, if you have `env` (most modern UNIX systems do):

```
        env CC=... ./configure && make all check install TEXINPUTS=...
```

There is wide variation in TeX directory trees, and `configure` cannot figure that out for you, so you will almost certainly have to supply a suitable value for `TEXINPUTS`, the name of a directory that TeX and `tgrind` will search for needed files.

If you don't set the `CC` environment variable, then `gcc` (or `cc`, if `gcc` is not available) will be assumed.

If you wish to undo a *make install*, just do *make uninstall*; this will remove any files in system directories put there by *make install*.

See below for further details, and for instructions for non-UNIX systems.

Introduction

Please report all problems, suggestions, and comments to the maintainer and co-author:

Nelson H. F. Beebe
University of Utah
Department of Mathematics, 322 INSCC

155 S 1400 E RM 233
Salt Lake City, UT 84112-0090
USA
Tel: +1 801 581 5254
FAX: +1 801 585 1640, +1 801 581 4148
Email: beebe@math.utah.edu , beebe@acm.org , beebe@computer.org (Internet)
WWW URL: http://www.math.utah.edu/~beebe/

The principal author is no longer able to maintain this program.

Installation

`tgrind` 3.00 has been updated to use the GNU `autoconf` automatic configuration system for UNIX installations.

GNU `autoconf` is run at the maintainer's site to produce the `configure` script from `configure.in`.

The `configure` script is run at each installer's UNIX site to produce `Makefile` from `Makefile.in`, and `config.h` from `config.hin`. The `configure` script is a large (3000+ lines) Bourne shell program that investigates various aspects of the local C implementation, and records its conclusions in `config.h`.

For convenience and safety, the distribution includes a subdirectory named `save` that contains read-only copies of the files `Makefile`, `config.h`, and `configure` created by `autoconf` and *make configure*. This will allow recovery from a lost or damaged `configure` file.

Should you do a *make maintainer-clean* [**not** recommended, except at the maintainer's site], the configure script will be deleted, and you will need recent versions of both GNU `m4` and `autoconf` correctly installed to reconstruct things, which can be done this way:

```
make -f save/Makefile reconfigure
```

Suitable hand-crafted `config.h` files are provided for non-UNIX systems, and in the unlikely event of a failure of the `configure` script on a UNIX system, `config.h` can be manually produced from a copy of `config.hin` with a few seconds of editing work. If you do this, remember to save a copy of your `config.h` under a different name, because running `configure` will destroy it. If you have GNU `autoconf` installed (the installation is very simple and source code is available from `ftp://prep.ai.mit.edu/pub/gnu/autoconf-`*`x.y`*`.tar.gz`), you might try augmenting `configure.in` instead, then run `autoconf`, `autoheader`, and `configure`.

Thus, on UNIX, installation normally consists of just two steps (assuming a `csh` -compatible shell):

```
setenv CC ...your favorite C or C++ compiler...
./configure && make all check install TEXINPUTS=...
```

There is wide variation in TeX directory trees, and `configure` cannot figure that out for you, so you will almost certainly have to supply a suitable value for `TEXINPUTS`, the name of a directory that TeX and `tgrind` will search for needed files.

If you like, add `OPT='your favorite optimization flags'` to the `make` command; by default, no optimization flags are set.

The GNU standard installation directories `/usr/local/bin` for binaries, and `/usr/local/man/man1` for manual pages are assumed. The prefix `/usr/local` can be overridden by providing an alternate definition on the command line:

```
make prefix=/some/other/path install TEXINPUTS=...
```

After installation, you can do

```
make distclean
```

to restore the directories to their distribution state. You should also do this between builds for different architectures from the same source tree; *neglecting to do so will almost certainly lead to failure*, because the `config.cache` file created by `configure` will lead to an incorrect `config.h` for the next build.

UNIX Systems

The code can be compiled with either C (K&R or ISO/ANSI Standard C) or C++ compilers. With some C++ compilers, it may be necessary to supply additional switches for force the compiler to stay in C++ mode, rather than reverting to C mode (e.g., on DEC Alpha OSF/1, you must do `setenv CC "cxx -x cxx"`).

On UNIX systems, the only changes that you are likely to need in the `Makefile` are the settings of `CC` and `CFLAGS`, and possibly, `DEFINES`, and if you wish to do *make install*, the settings of `bindir`, `MANDIR`, and `MANEXT`.

If you are installing `tgrind` on a new system, you should definitely run *make check* before installing it on your system. Sample output of *make check* from a UNIX system is given below.

These programs have been successfully built and tested with C and C++ compilers and tested on these systems for the 3.00 release (76 builds):

Machine and model	**O/S**	**Compilers**
DEC Alpha 2100-5/250	OSF/1 3.2	/bin/c89 /bin/cc /bin/cxx /usr/bin/c89 /usr/bin/cc /usr/ccs/bin/c89 /usr/ccs/bin/cc /usr/local/bin/g++ /usr/local/bin/gcc /usr/local/bin/lcc /usr/ucb/cc
DECstation 5000/200	ULTRIX 4.3	/bin/cc /usr/bin/cc /usr/local/bin/g++ /usr/local/bin/gcc /usr/local/bin/lcc
HP 9000/735	HP-UX 10.01	/bin/CC /bin/c89 /bin/cc /usr/bin/CC /usr/bin/c89 /usr/bin/cc /usr/ccs/bin/cc /usr/local/bin/g++ /usr/local/bin/gcc
IBM PowerPC 43P	AIX 4.1	/bin/c89 /bin/cc /bin/xlC /usr/bin/c89 /usr/bin/cc /usr/local/bin/g++ /usr/local/bin/gcc
IBM PowerPC 43P	AIX 4.2	/bin/c89 /bin/cc /bin/xlC /usr/bin/c89 /usr/bin/cc
Intel Pentium (200MHz MMX)	Linux 2.0.30	/usr/bin/cc /usr/bin/g++ /usr/bin/gcc
NeXT Turbostation	Mach 3.3	/bin/cc /usr/local/bin/g++ /usr/local/bin/gcc
SGI Challenge L	IRIX 5.3	/bin/CC /bin/cc /usr/bin/CC /usr/bin/cc /usr/local/bin/g++ /usr/local/bin/gcc
SGI O2 R10000-SC	IRIX 6.3	/bin/CC /bin/c89 /bin/cc /usr/bin/CC /usr/bin/c89 /usr/bin/cc
SGI Origin/200-4	IRIX 6.4	/bin/CC /bin/c89 /bin/cc /usr/bin/CC /usr/bin/c89 /usr/bin/cc /usr/local/bin/g++ /usr/local/bin/gcc
Sun SPARC 20/512	Solaris 2.6	/opt/SUNWspro/bin/CC /opt/SUNWspro/bin/c89 /opt/SUNWspro/bin/cc /usr/local/bin/g++ /usr/local/bin/gcc /usr/local/bin/lcc /usr/ucb/cc
Sun SPARC 4/380	SunOS 4.1.3	/bin/cc /usr/bin/cc /usr/lang/CC /usr/lang/acc /usr/local/bin/gcc /usr/ucb/cc

IBM PC DOS

PostScript font support

With only a few exceptions, PostScript fonts are licensed proprietary products that cannot be freely redistributed with a software package like `tgrind`. Nevertheless, the few that *are* available are included in the `tgrind` distribution in the `fonts` subdirectory. They are provided in five forms each, identified by these file extensions:

- `.afm`: Adobe Font Metric files, containing character dimensions, and usually also kerning and ligature information, but no information about character shapes;
- `.pfa`: PostScript Font ASCII files, containing the character shapes, but neither dimensions, nor kerning, nor ligature information;
- `.pfb`: PostScript Font Binary files, a more compact form of `.pfa` files, but with identical contents after format conversion;
- `.tfm`: TeX Font Metric files, containing character dimensions, and kerning and ligature information;
- `.vf`: TeX Virtual Font files, containing remappings of characters from one or more physical fonts into a virtual font containing characters in the order expected with ordinary TeX fonts.

There is also a file named `psfonts.add`, containing suitable mappings that must be added to your local `psfonts.map` file to make these fonts accessible. However, check your own `psfonts.map` file to see whether it might not already have the required mappings. The fonts included here are already available in many TeX installations.

Although including both `.pfa` and `.pfb` files in the `tgrind` distribution is redundant, some PostScript software requires one of these, and cannot handle the other. Providing both formats therefore makes sense. The `t1ascii` and `tibinary` utilities in the `t1utils` distribution, available in the CTAN archives (see the penultimate section of the `tgrind` manual pages for CTAN locations), can convert between these formats.

Because TeX installations differ widely in their directory structures, there is no consistent place that the `make install` step could install fonts in. It is therefore up to *you, the installer,* to decide whether or not to install the provided fonts, and where.

Many UNIX systems have TeX DVI driver software that interfaces to the UNIX `MakeTeXPK` script. The drivers call that script when they cannot find a font, and the script then attempts to generate the font on-the-fly, either by running Metafont, or by using `gsftopk` and `gs` (Aladdin ghostscript), or `ps2pk`, to rasterize PostScript Type 1 fonts, and convert them to TeX PK (packed bitmap) format. In particular, the maintainer's own DVI drivers, Tom Rokicki's `dvips`, and Paul Vojta's `xdvi` drivers all do this. This makes PostScript Type 1 fonts usable on a wide variety of output devices, including non-PostScript printers, and workstation screens.

Test suite

The `tgrind` distribution includes an extensive test suite. It is run by

```
make check
```

Each test consists of up to 4 files, named `check` *nnn* (*nnn* = 001, 002, ...), with these extensions:

`.in` input data

`.out` correct `tgrind` *stdout* data produced at the maintainer's site

`.err` correct `tgrind` *stderr* data produced at the maintainer's site

`.opt` command line arguments [optional]

The input files are stored in the `test` subdirectory, and the correct `.out` and `.err` files are stored in the `test/okay` subdirectory.

The *make check* job loops over the list of test files, and executes either

```
../tfontedpr checknnn.in > checknnn.out 2> checknnn.err
```

or

```
../tfontedpr 'cat checknnn.opt' checknnn.in > checknnn.out 2> checknnn.err
```

depending on whether a `.opt` file is available or not. The resulting `.out` and `.err` files are then compared with the expected output in the `.out` and `.err` files in the `test/okay` subdirectory.

All of the `.out` files are valid plain TeX files, and each can be typeset and viewed or printed to demonstrate `tgrind`'s formatting. This is not, however, done as part of the validation test suite.

For use on other operating systems, the `test/check.bat` file for PC DOS may serve as a useful starting point for preparing a script to run the tests.

Please *do* run the validation suite at your site before installing the program. Compilers are complex software systems that also have bugs, so the fact that the program runs correctly somewhere else does not mean that it will do so on a different system.

Sample build output for UNIX

Here is a log of a successful build on Sun Solaris 2.6 using the native C++ compiler, `CC`:

```
% env CC=CC ./configure && make all check
creating cache ./config.cache
checking whether make sets ${MAKE}... yes
checking for gcc... CC
checking whether the C compiler (CC  ) works... yes
checking whether the C compiler (CC  ) is a cross-compiler... no
checking whether we are using GNU C... no
checking for c++... c++
checking whether the C++ compiler (c++  ) works... yes
checking whether the C++ compiler (c++  ) is a cross-compiler... no
checking whether we are using GNU C++... yes
checking whether c++ accepts -g... yes
checking for a BSD compatible install... /usr/local/bin/install -c
checking whether ln -s works... yes
checking for col... col -x -b
checking for gawk... gawk
checking for chmod... chmod
checking for checksum... checksum
checking for rcp... rcp
checking for cmp... cmp
checking for deroff... deroff
checking for diff... diff
checking for distill... distill
checking for dvips... dvips
checking for dvialw... (cached) dvips
checking for dw... dw
checking for geqn... geqn
checking for gzip... gzip
checking for ispell... ispell
checking for lpr... lpr
checking for ls... ls
checking for man2html... man2html
```

```
checkıng for mkdır... mkdır
checking for mv... mv
checking for groff... groff
checking for rmdir... rmdir
checking for sed... sed
checking for sort... sort
checking for spell... spell
checking for strip... strip
checking for gnutar... no
checking for gtar... no
checking for tar... tar
checking for gtbl... gtbl
checking for tex... /usr/local/lib/tex
checking for touch... touch
checking for unzip... unzip
checking for zip... zip
checking for zoo... zoo
checking for Standard C/C++ function declarations... yes
checking how to run the C preprocessor... CC -E
checking for ANSI C header files... no
checking for ctype.h... yes
checking for fcntl.h... yes
checking for libc.h... yes
checking for stdio.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
checking for time.h... yes
checking for unistd.h... yes
checking for Standard C/C++ prototype support... yes
checking for working const... no
checking for size_t... yes
checking for atoi... no
checking for atol... no
checking for remove... no
checking for rename... no
checking for strtol... no
checking for strtoul... no
checking for unlink... no
checking for argument type of ctime()... const time_t *
checking for struct stat... yes
updating cache ./config.cache
creating ./config.status
creating Makefile
creating config.h
CC -DHAVE_CONFIG_H -I.  -DDEFSFILE=\"/usr/local/lib/tex/inputs/vgrindefs\" -c tfontedpr.c
CC -DHAVE_CONFIG_H -I.    -c vgrindefs.c -o vgrindefs.o
CC -DHAVE_CONFIG_H -I.    -c regexp.c -o regexp.o
CC -DHAVE_CONFIG_H -I.  -o tfontedpr tfontedpr.o vgrindefs.o regexp.o
CC -DHAVE_CONFIG_H -I.    -c retest.c -o retest.o
CC -DHAVE_CONFIG_H -I.  -o retest retest.o regexp.o
CC -c -DHAVE_CONFIG_H -I.  \
        '-DAWK="gawk"' \
        '-DDIR="ls -l"' \
        '-DDVI="dvips"' \
        '-DDVI_OPTION="-o"' \
        '-DLIBDIR="/usr/local/lib/tex/inputs/"' \
        '-DPRINT="lpr"' \
        '-DTGRIND="tgrind"' \
        tgrind.c
CC -DHAVE_CONFIG_H -I.  tgrind.o -o tgrind
The following checks should produce no output other than their names...
check001
check002
check003
check004
check005
check006
check007
check008
check009
check010
check011
check012
```

```
check013
check014
check015
check016
check017
check018
check019
check020
```